New Optimization Algorithms in Physics

Edited by
Alexander K. Hartmann and Heiko Rieger

WILEY-VCH

WILEY-VCH Verlag GmbH & Co. KGaA

Editors

Alexander K. Hartmann
Universität Göttingen, Germany
hartmann@theorie.physik.uni-goettingen.de

Heiko Rieger
Universität des Saarlandes, Germany
h.rieger@mx-uni-saarland.de

Cover Picture
Short artificial peptide (U. H. E. Hansmann); Dense configuration of polydisperse hard discs (W. Krauth); Cut Polytope for the complete graph with N=3 (F. Liers et al); Factor graph for 3-SAT (R. Zecchina)

Library of Congress Card No.: applied for
British Library Cataloging-in-Publication Data:
A catalogue record for this book is available from the British Library

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

Printed in the Federal Republic of Germany
Printed on acid-free paper

Composition Uwe Krieg, Berlin
Printing Strauss GmbH, Mörlenbach
Bookbinding Litges & Dopf Buchbinderei GmbH, Heppenheim
ISBN 3-527-40406-6

Contents

List of Contributors

- *Jean-Christian Anglès d'Auriac,* Ch. 6

  Centre de Recherches sur les Tres Basses Temperatures
  BP 166, F-38042 Grenoble, France
  e-mail: dauriac@grenoble.cnrs.fr

- *Stefan Boettcher,* Ch. 11

  Stefan Boettcher
  Physics Department
  Emory University
  Atlanta, GA 30322; USA
  e-mail: sboettc@emory.edu
  www.physics.emory.edu/faculty/boettcher

- *Simona Cocco,* Ch. 8

  Laboratoire de Dynamique des Fluides Complexes
  3 rue de l'Université
  67000 Strasbourg, France
  e-mail: cocco@ldfc.u-strasbg.fr

- *Liat Ein-Dor,* Ch. 8

  Department of Physics of Complex Systems
  Wezmann Institute
  Rehovot 76100, Israel

  and

  Laboratoire de Physique Théorique de l'ENS
  24 rue Lhomond
  75005 Paris, France

- *Ulrich H. E. Hansmann,* Ch. 13

  Department of Physics
  Michigan Technological University
  Houghton
  MI 49931-1295, USA
  e-mail: hansmann@mtu.edu

- *Alexander K. Hartmann,* Ch. 1, 12

  Institut für Theoretische Physik
  Universität Göttingen
  Tammanstraße 1
  D-37077 Göttingen, Germany
  e-mail:
  hartmann@theorie.physik.uni-goettingen.de

- *Michael Jünger,* Ch. 4

  Institut für Informatik
  Universität zu Köln
  Pohligstraße 1
  D-50969 Köln, Germany
  e-mail: mjuenger@informatik.uni-koeln.de

- *Werner Krauth,* Ch. 2

  CNRS-Laboratoire de Physique Statistique
  Ecole Normale Supérieure
  24, rue Lhomond
  F-75231 Paris Cedex 05, France
  e-mail: Werner.Krauth@ens.fr

- *Frauke Liers,* Ch. 4

  Institut für Informatik
  Universität zu Köln
  Pohligstraße 1
  D-50969 Köln, Germany
  e-mail: liers@informatik.uni-koeln.de

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

• *Olivier C. Martin,* Ch. 3

LPTMS, Université Paris-Sud
Orsay Cedex 91405, France
e-mail: martino@ipno.in2p3.fr

• *A. Alan Middleton,* Ch. 5

Department of Physics
Syracuse University
Syracuse, NY, USA
e-mail: aam@syr.edu

• *Remi Monasson,* Ch. 8

Laboratoire de Physique Théorique de l'ENS
24 rue Lhomond
75005 Paris, France
e-mail: monasson@lpt.ens.fr

and

Laboratoire de Physique Théorique
3 rue de l'Université
67000 Strasbourg, France

• *Károly F. Pál,* Ch. 10

Department of Theoretical Physics
Institute of Nuclear Research of the Hungarian Academy of Sciences
Bem tér 18/c
H-4026 Debrecen, Hungary
e-mail: kfpal@hal.atomki.hu

• *Gerhard Reinelt,* Ch. 4

Institut für Informatik
Ruprecht-Karls-Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany
e-mail:
Gerhard.Reinelt@informatik.uni-heidelberg.de

• *Heiko Rieger,* Ch. 1

Theoretische Physik
Universität des Saarlandes
D-66041 Saarbrücken, Germany
e-mail: rieger@lusi.uni-sb.de

• *Giovanni Rinaldi,* Ch. 4

Istituto di Analisi dei Sistemi ed Informatica
'Antonio Ruberti' - CNR
Viale Manzoni, 30
00185 Roma, Italy
e-mail: rinaldi@iasi.rm.cnr.it

• *Martin Weigt,* Ch. 7

Institut für Theoretische Physik
Universität Göttingen
Tammanstraße 1
D-37077 Göttingen, Germany
e-mail: weigt@theorie.physik.uni-goettingen.de

• *Riccardo Zecchina,* Ch. 9

International Centre for Theoretical Physics (ICTP)
Strada Costiera, 11 P.O.Box 586
I-34100 Trieste, Italy
e-mail: zecchina@ictp.trieste.it

1 Introduction

Alexander K. Hartmann and Heiko Rieger

Optimization problems occur very frequently in physics. Some of them are easy to handle with conventional methods also used in other areas such as economy or operations research. But as soon as a huge number of degrees of freedom are involved, as is typically the case in statistical physics, condensed matter, astrophysics and biophysics, conventional methods fail to find the optimum in a reasonable time and new methods have to be invented. This book contains a representative collection of new optimization algorithms that have been devised by physicists from various fields, sometimes based on methods developed by computer scientists and mathematicians. However, it is not a mere collection of algorithms but tries to demonstrates their scope and efficiency by describing typical situations in physics where they are useful.

The individual articles of this collections are self-contained and should be understandable for scientists routinely using numerical tools. A more basic and pedagogical introduction into optimization algorithms is our book on *Optimization Algorithms in Physics*, which can serve as an appendix for the newcomer to this field of computational physics or for undergraduate students. The reason why we found it necessary to compose another book in this field with a greater focus is the fact that the application of optimization methods is one of the strongest growing fields in physics. The main reasons for these current developments are the following key factors:

First of all great progress has been made in the development of new combinatorial optimization methods in computer science. Using these sophisticated approaches, much larger system sizes of the corresponding physical systems can be treated. For many models the systems sizes which were accessible before, were too small to obtain reliable and significant data. However, this is now possible. In this way computer science has helped physics.

But knowledge transfer also works the other way round. Physics provides still new insights and methods of treating optimization problems, such as the earlier invention of the simulated annealing technique. Recent algorithmic developments in physics are, e.g., the extremal optimization method or the hysteric optimization approach, both covered in this book.

Moreover, phase transitions were recently found in "classical" optimization problems within theoretical computer science, during the study of suitably parameterized ensembles. These phase transitions very often coincide with peaks of the running time or with changes of the typical-case complexity from polynomial to exponential. As well as the gain from taking the physical viewpoint, by mapping the optimization problems to physical systems and applying methods from statistical physics, it is possible to obtain many results, which have not been found with traditional mathematical techniques. This is true also for the analysis of the typical-case complexity of (random) algorithms.

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

Finally: All benefit from the increasing power of computers. Despite all predictions, the speed of the hardware still seems to grow exponentially fast, making the application of optimizations methods more and more valuable.

Thus the aim of this book is to promote progress in the fields given above. Physicists will become familiar with the huge progress still taking place in the development of algorithmic techniques. On the other hand, the new developments of physically inspired algorithms can be very useful in computer science as well. In particular the application of physical methods in the field of phase transitions seems to be a very promising field for the next decade.

Currently, the interactions between different communities, namely mathematics, computer science, biology, economy and physics are still too weak. Only by gathering researchers from these different groups and trying to find a common language, can real progress be achieved. All problems, algorithms and results are presented here in a pedagogical way, which makes the information available to a broad audience. This is the main purpose of this collection of papers.

The book contains three main parts. In the first part, we focus on applications of optimization algorithms to problems from physics. The standard way of solving computational problems in statistical physics is to use a Monte Carlo simulation. In his contribution, Werner Krauth shows that by using modern cluster algorithms, many previously inaccessible models can be treated at low temperatures (obtaining low, i.e., minimum energies) or respectively, high densities. He studies as examples the phase separation in binary mixtures and the application of the algorithm to monomer-dimer models. Next, Olivier Martin surveys algorithms for Ising spin-glass ground-state calculations and he explains one new Monte Carlo algorithm in detail. It is a cluster method based on the real-space renormalization group.

Monte Carlo methods, like those shown in the first two contributions, are very efficient and have a wide range of applicability, but they do not guarantee to find a global optimum solution. In contrast, the Branch-and-Cut approach is an exact algorithm. It is presented by Frauke Liers, Michael Jünger, Gerhard Reinelt and Giovanni Rinaldi. They explain the method for an application to the max-cut problem, which is used here for the ground-state calculation of three-dimensional Ising spin glasses.

Another important class of problems in statistical physics is the random-field Ising model. Alan Middleton explains how one can calculate ground states using push/relabel algorithms in polynomial time, how these algorithms perform near phase transitions and how one can use it to characterize the ground-state landscape of the random-field model. In the last chapter of the first part, Jean-Christian Anglès d'Auriac describes a new method for calculating the partition function and other thermodynamic quantities of the infinite-state Potts model with random bonds using a combinatorial optimization algorithm. The latter is based on the concept of submodular functions, which might also prove useful in a number of other applications in the near future.

The second part is dedicated to the study of phase transitions in combinatorial optimization problems. First, Martin Weigt introduces the Satisfiability Problem (SAT), the most fundamental problem in computational complexity theory. He then shows how one can generate large SAT formulas which have a solution but where the solution is hard to find for local algorithms like Walksat. This behavior can be understood by solving the corresponding physical problem analytically by using techniques from statistical mechanics. Simona Cocco, Liat Ein-Dor and Remi Monasson show how one can calculate the typical running time of exact backtracking algorithms for SAT and for the coloring problem. The basic idea is to investigate the dynamics of the algorithm moving in the phase diagram of the problem. Finally, Riccardo Zecchina presents the currently fastest Algorithm for SAT, the Survey Propagation algorithm, which allows to solve SAT instances near the SAT-UNSAT phase transition of systems having 10^6 variables. The method is based on the cavity approach, an analytical technique used to study mean-field-like disordered systems in statistical physics. Nevertheless, his presentation is solely based on probability theory, making it also very accessible to non-physicists.

The third part of this book is on new heuristics and interdisciplinary applications. Károly Pál presents an optimization method which is inspired by a physical technique, the measurement of hysteresis in a magnetic system. The basic idea is to demagnetize a system by performing hysteresis loops with continuously decreasing magnitude. He presents the algorithm in a very general style, which in principle allows arbitrary applications. As examples, results for spin glasses and the traveling salesman problem are shown. Stefan Boettcher explains another very general algorithm, the extremal optimization algorithm. Its basic idea is very simple and similar to genetic algorithms. The latter ones usually have many free parameters, which must be tuned to obtain an efficient algorithm. Extremal optimization has the advantage that it is, in the simplest variant, absolutely parameter free. Another major difference in comparison with genetic algorithms is that fitness values are not assigned to different configurations but to different particles of one configuration. Application to graph coloring, spin glasses and image matching are given.

The last two contributions contain applications from Molecular Biology. After providing some biological background, Alexander Hartmann explains alignment algorithms, which are used to compare biological sequences by applying a shortest-path algorithm. As an application, a method to obtain the rare-event tail of the statistics of protein alignments is presented. Finally, Ulrich Hansmann reviews methods used to solve protein-folding problems via energy minimization and in particular explains energy-landscape paving. The basic idea is that one initially modifies the energy landscape such that the global minimum is easier to find. During the simulation, the energy landscape gradually approaches the ordinal one. Furthermore, the algorithm tries to avoid previously visited regions, if the energy is not low enough. Various results for the influence of the temperature on helix formation are also shown.

Compiling this book would not have been possible without the help of many people and various institutions. First of all, we would like to thank all authors for preparing their valuable contributions and also for their helpful cooperation. Furthermore, we are particularly indebted to Vera Palmer, Uwe Krieg, and Cornelia Wanka from Wiley-VCH for the excellent collaboration. Financial support was provided by the VolkswagenStiftung within the Program "Nachwuchsgruppen an Universitäten", by the Deutsche Forschungsgemeinschaft (DFG), by the International Conference and Research Center for Computer Science Schloss Dagstuhl and by the Institute for Scientific Interchange (ISI) Foundation in Turin. The European Community supported this book financially via the Human Potential Program under contract number HPRN-CT-2002-00307 (DYGLAGEMEM), via the High-Level Scientific Conferences (HLSC) program, and via the Complex Systems Network of Excellence "Exystence". This book was prepared in connection with the Dagstuhl Seminar No. 03381 "New Optimization Algorithms in Physics".

Part I: Applications in Physics

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

2 Cluster Monte Carlo Algorithms

Werner Krauth

In recent years, a better understanding of the Monte Carlo method has provided us with many new techniques in different areas of statistical physics. Of particular interest are the so called cluster methods, which exploit the considerable algorithmic freedom given by the detailed balance condition. Cluster algorithms appear, among other systems, in classical spin models, such as the Ising model [14], in lattice quantum models (bosons, quantum spins and related systems) [5] and in hard spheres and other "entropic" systems for which the configurational energy is either zero or infinite [4].

In this chapter, we discuss the basic idea of cluster algorithms with special emphasis on the pivot cluster method for hard spheres and related systems, for which several recent applications are presented. We provide less technical detail but more context than in the original papers. The best implementations of the pivot cluster algorithm, the "pocket" algorithm [10], can be programmed in a few lines. We start with a short exposition of the detailed balance condition, and of "*a priori*" probabilities, which are needed to understand how optimized Monte Carlo algorithms may be developed. A more detailed discussion of these subjects will appear in a forthcoming book [9].

2.1 Detailed Balance and a priori Probabilities

In contrast with the combinatorial optimization methods discussed elsewhere in this book, the Monte Carlo approach does not construct a well-defined state of the system – minimizing the energy, or maximizing flow, etc – but attempts to generate a number of statistically independent representative configurations a, with probability $\pi(a)$. In classical equilibrium statistical physics, $\pi(a)$ is given by the Boltzmann distribution, whereas, in quantum statistics, the weight is the diagonal many-body density matrix.

In order to generate these configurations with the appropriate weight (and optimal speed), the Monte Carlo algorithm moves (in one iteration) from configuration a to configuration b with probability $P(a \to b)$. This transition probability is chosen to satisfy the fundamental condition of detailed balance

$$\pi(a)P(a \to b) = \pi(b)P(b \to a) \tag{2.1}$$

which is implemented using the Metropolis algorithm

$$P(a \to b) = \min\left(1, \frac{\pi(b)}{\pi(a)}\right) \tag{2.2}$$

or one of its variants.

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

For the prototypical Ising model, the stationary probability distribution (the statistical weight) of a configuration is the Boltzmann distribution with an energy given by

$$E = -J \sum_{\langle i,j \rangle} S_i S_j \quad L > 0 \tag{2.3}$$

as used and modified in many other places in this book. A common move consists of a spin flip on a particular site i, transforming configuration a into another configuration b. This is shown in Figure 2.1 (left). In a hard sphere gas, also shown in Figure 2.1 (right), one typically displaces a single particle i from $\mathbf{x}$ to $\mathbf{x} + \boldsymbol{\delta}$. There is a slight difference between these two simple algorithms: by flipping the same spin twice, one goes back to the initial configuration: a spin flip is its own inverse. In contrast, in the case of the hard-sphere system, displacing a particle twice by the same vector $\boldsymbol{\delta}$ does not usually bring one back to the original configuration.

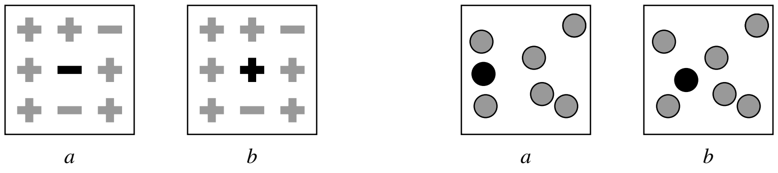


Figure 2.1: Two examples of local Monte Carlo algorithms: the two-dimensional Ising model with single-spin flip dynamics (left) and two-dimensional hard disks with a single-particle move (right).

An essential concept is the one of an *a priori* probability: it accounts for the fact that the probability $P(a \to b)$ is a composite object, constructed from the probability of *considering* the move from a to b, and the probability of *accepting* it.

$$P(a \to b) = \underbrace{\mathcal{A}(a \to b)}_{\text{consider } a \to b} \times \underbrace{\tilde{P}(a \to b)}_{\text{accept } a \to b}$$

In usual Monte Carlo terminology, if $a \to b$ is rejected (after having been considered), then the "move" $a \to a$ is chosen instead and the system remains where it is.

With these definitions, the detailed balance condition Eq. (2.1) can be written as

$$\frac{\tilde{P}(a \to b)}{\tilde{P}(b \to a)} = \frac{\pi(b)}{\mathcal{A}(a \to b)} \frac{\mathcal{A}(b \to a)}{\pi(a)}$$

and implemented by a Metropolis algorithm generalized from Eq. (2.2):

$$\tilde{P}(a \to b) = \min \left\{ 1, \frac{\pi(b)}{\mathcal{A}(a \to b)} \frac{\mathcal{A}(b \to a)}{\pi(a)} \right\} \tag{2.4}$$

It is very important to realize that the expression "*a priori* probability $\mathcal{A}(a \to b)$" is synonymous to "Monte Carlo algorithm". A Monte Carlo algorithm $\mathcal{A}(a \to b)$ of our own conception must satisfy three conditions:

1. It must lead the state of the system from a configuration a to a configuration b, in such a way that, eventually, all configurations in phase space can be reached (ergodicity).

2. It must allow to compute the ratio $\pi(a)/\pi(b)$. This is trivially satisfied, at least for classical systems, as the statistical weight is simply a function of the energy.

3. It must allow, for any possible transition $a \to b$, to compute both the probabilities $\mathcal{A}(a \to b)$ and $\mathcal{A}(b \to a)$. Again, it is the ratio of probabilities which is important.

A trivial application of *a priori* probabilities for hard spheres is given in Figure 2.2. (Suppose that the points a and b are embedded in a large two-dimensional plane.) On the left side of the figure, we see one of the standard choices for the trial moves $\mathbf{x} \to \mathbf{x} + \boldsymbol{\delta}$ of a particle in Figure 2.1: The vector $\boldsymbol{\delta}$ is uniformly sampled from a square centered around the current position. If, however, we decide for some obscure reason to sample $\boldsymbol{\delta}$ from a triangle, we realize that in cases such as the one shown in Figure 2.2 (right), the *a priori* probability for the return move vanishes. It is easy to see from Eq. (2.4) that, in this case, both $P(a \to b)$ and $P(b \to a)$ are zero.

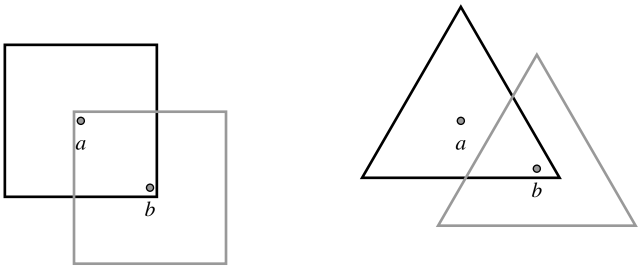


Figure 2.2: A priori probabilities for the hard-sphere system. Left: "square" – $\mathcal{A}(a \to b)$ is constant within the square boundary, and zero outside. By construction, $\mathcal{A}(a \to b) = \mathcal{A}(b \to a)$. Right: "triangle" – for the analogous (if hypothetical) case of a triangle, there are pairs a, b, where $\mathcal{A}(a \to b)$ is finite, but $\mathcal{A}(b \to a) = 0$. Both rates $P(a \to b)$ and $P(b \to a)$ vanish.

Notwithstanding its simplicity, the triangle "algorithm" illustrates that *any* Monte Carlo method $\mathcal{A}(a \to b)$ can be made to comply with detailed balance, if we feed it through Eq. (2.4). The usefulness of the algorithm is uniquely determined by the speed with which it moves through configuration space, and is highest if no rejections at all appear. It is to be noted, however, that, even if $\widetilde{P}(a \to b)$ is always 1 (no rejections), the simulation *can* remain rather difficult. This happens, for example, in the two-dimensional XY-model and in several examples treated below.

Local algorithms are satisfactory for many problems but fail whenever the typical differences between relevant configurations are much larger than the change that can be achieved by one iteration of the Monte Carlo algorithm. In the Ising model at the critical point, for example, the distribution of magnetizations is wide, but the local Monte Carlo algorithm implements a change of magnetization of only ± 2. This mismatch lies at the core of critical slowing down in experimental systems and on the computer.

In liquids, modeled e.g. by the hard-sphere system, another well-known limiting factor is that density fluctuations can relax only through local diffusion. This process generates slow hydrodynamic modes, if the overall diffusion constants are small.

Besides these slow *dense* systems, there is also the class of highly *constrained* models, of which binary mixtures will be treated later. In these systems, the motion of some degrees of freedom naturally couple to many others. In a binary mixture, e. g., a big colloidal particle is surrounded by a large number of small particles, which are influenced by its motion. This is extremely difficult to deal with in Monte Carlo simulations, where the local moves $\mathbf{x} \to \mathbf{x} + \boldsymbol{\delta}$ are essentially the unconstrained motion of an isolated particle.

2.2 The Wolff Cluster Algorithm for the Ising Model

The local spin-flip Monte Carlo algorithm not being satisfactory, it would be much better to move large parts of the system, so called clusters. This cannot be done by a blind flip of one or many spins (with $\mathcal{A}(a \to b) = \mathcal{A}(b \to a)$), which allows unit acceptance rate both for the move $a \to b$ and its reverse $b \to a$ only if the energies of both configurations are the same. One needs an algorithm whose *a priori* probabilities $\mathcal{A}(a \to b)$ and $\mathcal{A}(b \to a)$ soak up any differences in statistical weight $\pi(a)$ and $\pi(b)$.

This can be done by starting the construction of a cluster with a randomly sampled spin and by iteratively adding neighboring sites of the same magnetization with a probability p. To be precise, one should speak about "links": if site i is in the cluster and a neighboring site j is not, and if, furthermore, $S_i = S_j$, then one should add link $\langle i, j \rangle$ with probability p. A site is added to the cluster if it is connected by at least one link. In configuration a of Figure 2.3, the cluster construction has stopped in the presence of 9 links "$--$" across the boundary. Each of these links could have been accepted with probability p, but has been rejected. This gives a term $(1-p)^9$ in the *a priori* probability. Flipping the cluster brings us to configuration b. The construction of the cluster for configuration b would stop in the presence of 19 links "++" across the boundary (*a priori* probability $\propto (1-p)^{19}$)).

This allows us to compute the *a priori* probabilities

$$\begin{aligned}
\mathcal{A}(a \to b) &= \mathcal{A}_{\text{interior}} \times (1-p)^9 \\
\mathcal{A}(b \to a) &= \mathcal{A}_{\text{interior}} \times (1-p)^{19} \\
E_a &= E_{\text{interior}} + E_{\text{exterior}} - 9 \times J + 19 \times J \qquad (\pi_a = \exp[-\beta E_a]) \\
E_b &= E_{\text{interior}} + E_{\text{exterior}} - 19 \times J + 9 \times J \qquad (\pi_b = \exp[-\beta E_b])
\end{aligned}$$

In these equations, the "interior" refers to the part of the cluster which does not touch the boundary. By construction, the "interior" and "exterior" energies and *a priori* probabilities are the same for any pair of configurations a and b which are connected through a single cluster flip.

We thus dispose of all the information needed to evaluate the acceptance probability $\tilde{P}$ in Eq. (2.4), which we write more generally in terms of the number of "same" and of "different" links in the configuration a. These notions are interchanged for configuration b (in Figure 2.3,

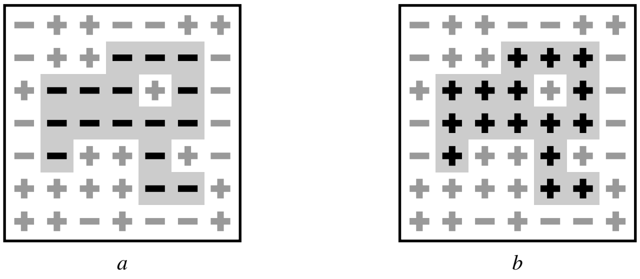


Figure 2.3: The Wolff cluster algorithm for the Ising model adds, with probability p, a link connecting a site outside the cluster to a site already in the cluster (thereby adding the site). In the configuration a, construction of the cluster (as shown) stopped with 9 links "$--$", corresponding to an *a priori* probability $\mathcal{A}(a \to b) = \mathcal{A}_{\text{interior}} \times (1-p)^9$. The return move stops with probability $\mathcal{A}(b \to a) = \mathcal{A}_{\text{interior}} \times (1-p)^{19}$, as there are 19 links "++" across the boundary in configuration b.

we have $n_{\text{same}} = 9$, $n_{\text{diff}} = 19$). With the energy scale J set to 1, we find

$$\begin{aligned}\tilde{P}(a \to b) &= \min\left\{1, \frac{e^{\beta n_{\text{diff}}} e^{-\beta n_{\text{same}}}}{(1-p)^{n_{\text{same}}}} \frac{(1-p)^{n_{\text{diff}}}}{e^{-\beta n_{\text{diff}}} e^{\beta n_{\text{same}}}}\right\} \\ &= \min\left\{1, \left[\frac{e^{-2\beta}}{1-p}\right]^{n_{\text{same}}} \left[\frac{1-p}{e^{-2\beta}}\right]^{n_{\text{diff}}}\right\} \qquad (2.5)\end{aligned}$$

Once the cluster construction stops, we know the configuration b, may count n_{same} and n_{diff}, and evaluate $\tilde{P}(a \to b)$. Of course, a lucky coincidence[1] occurs for $p = 1 - \exp[-2J\beta]$. This special choice yields a rejection-free algorithm whose acceptance probability is unity for all possible moves and is implemented in the celebrated Wolff cluster algorithm [14], the fastest currently known simulation method for the Ising model. The Wolff algorithm can be programmed in a few lines, by keeping a vector of cluster spins, and an active frontier, as shown below. The algorithm below presents a single iteration $a \to b$. The function $\text{ran}[0,1]$ denotes a uniformly distributed random number between 0 and 1, and p is set to the magic value $p = 1 - \exp[-2J\beta]$. The implementation uses the fact that a cluster can grow only at its frontier (called the "old" frontier $\mathcal{F}_{\text{old}}$, and generating the new one $\mathcal{F}_{\text{new}}$). It goes without saying that for the magic value of p we do not have to evaluate $\tilde{P}(a \to b)$ in Eq. (2.5), as it is always 1. Any proposed move is accepted.

[1] This accident explains the deep connection between the Ising model and percolation.

```
**algorithm** wolff-cluster
**begin**
    $i :=$ random particle;
    $\mathcal{C} := \{i\}$;
    $\mathcal{F}_{\text{old}} := \{i\}$;
    **while** $\mathcal{F}_{\text{old}} \neq \{\}$ **do**
    **begin**
        $\mathcal{F}_{\text{new}} := \{\}$;
        **for** $\forall\, i \in \mathcal{F}_{\text{old}}$ **do**
        **begin**
            **for** $\forall\, j$ neighbor of $i$ with $S_i = S_j,\ j \notin \mathcal{C}$ **do**
            **begin**
                **if** ran$[0,1] < p$ **then**
                **begin**
                    $\mathcal{F}_{\text{new}} := \mathcal{F}_{\text{new}} \cup \{j\}$;
                    $\mathcal{C} := \mathcal{C} \cup \{j\}$;
                **end**
            **end**
        **end**
        $\mathcal{F}_{\text{old}} := \mathcal{F}_{\text{new}}$;
    **end**
    **for** $\forall\, i \in \mathcal{C}$ **do**
    $S_i := -S_i$;
**end**
```

2.3 Cluster Algorithm for Hard Spheres and Related Systems

We want to further exploit the analogy between the spin model and the hard-sphere system. As the spin-cluster algorithm constructs a cluster of spins which *flip* together, one might think that a cluster algorithm for hard spheres should identify "blobs" of spheres that *move* together. Such a macroscopic ballistic motion would replace slow diffusion.

To see that this strategy cannot be successful, it suffices to look at the generalized detailed balance condition in the example shown in Figure 2.4: any reasonable algorithm $\mathcal{A}$ would have less trouble spotting the cluster of dark disks in configuration a than in b. This means that $\mathcal{A}(a \to b) \gg \mathcal{A}(b \to a)$ and that the acceptance rate $\tilde{P}(a \to b)$ would be very small.

The imbalance between $\mathcal{A}(a \to b)$ and $\mathcal{A}(b \to a)$ can, however, be avoided if the two transition probabilities are protected by a symmetry principle: the transformation T producing b from a must be the same as the one producing a from b. Thus, T should be its own inverse.

In Figure 2.5, this program is applied to a hard disk configuration using, as transformation T, a rotation by an angle π around an arbitrarily sampled pivot (denoted by $\oplus$, for each iteration a new pivot is used). Notice that for a symmetric particle, the rotation by an angle π is identical to the reflection around the pivot. It is useful to transform not just a single particle,

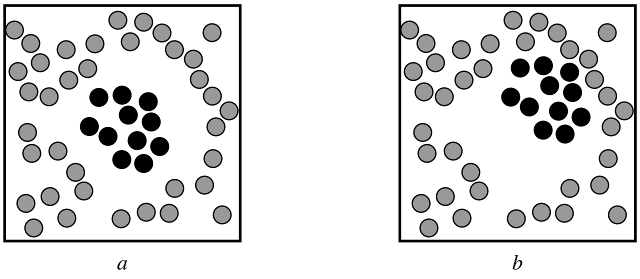


Figure 2.4: The dark disks are easier to identify as a cluster in configuration a than in b, where they are fused into the background. This means that, for the configurations a and b shown in this figure, $\mathcal{A}(a \to b) \gg \mathcal{A}(b \to a)$ for any generic Monte Carlo algorithm. As $\pi(a) = \pi(b)$, the acceptance probability $\tilde{\mathcal{P}}(a \to b)$ in Eq. (2.4) will be extremely small. The problem can be avoided [4] if the transformation $a \to b$ is protected by a symmetry principle: it must be its own inverse.

but the whole original configuration a yielding the "copy". By overlaying the original with its rotated copy, we may identify the invariant sub-ensembles (clusters) which transform independently under T. For example, in Figure 2.5, we may rotate the disks numbered 6, 8, and 9, which form a cluster of overlapping disks in the ensemble of overlayed original and copy.

In Figure 2.5, there are the following three invariant clusters:

$$\{6, 8, 9\}, \{2, 3, 4, 7\}, \{1, 5\} \tag{2.6}$$

The configuration b in Figure 2.5 shows the final positions after rotation of the first of these clusters. By construction, $\mathcal{A}(a \to b) = \mathcal{A}(b \to a)$ and $\pi(a) = \pi(b)$. This perfect symmetry ensures that detailed balance is satisfied for the non-local move. Notice that moving the cluster $\{1, 5\}$ is equivalent to exchanging the labels of the two particles and performing two local moves. Ergodicity of the algorithm follows from ergodicity of the local algorithm, as a local move $\mathbf{x} \to \mathbf{x} + \boldsymbol{\delta}$ can always be disguised as a cluster rotation around the pivot $\mathbf{x} + \boldsymbol{\delta}/2$.

Figure 2.5 indicates the basic limitation of the pivot cluster approach: if the density of particles becomes too large, almost all particles will be in the same cluster, and flipping it will essentially rotate the whole system. Nevertheless, even though above the percolation threshold in the thermodynamic limit there exists a large cluster containing a finite fraction of all particles, the remaining particles are distributed among a distribution of small clusters. This means that finite clusters of various sizes will be produced. These may give rise to useful moves, for example in the case of dense polydisperse disks discussed below. Even small clusters provide non-diffusive mass transport if they contain an odd number of particles (cf. the example in Figure 2.5) or particles of different type.

It is also useful to discuss what will happen if the "copy" does not stem from a symmetry operation, for example, if the copy is obtained from the original through a simple translation with a vector $\boldsymbol{\delta}$. In this case, there would still be clusters, but they no longer appear in pairs. It would still be possible to flip individual clusters, but not to conserve the number of particles

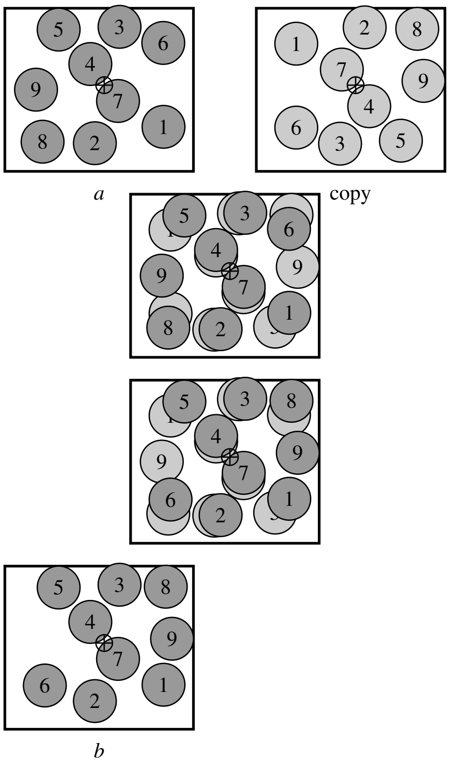


Figure 2.5: The pivot cluster algorithm performs a symmetry operation which is its own inverse. In this system of hard disks (with periodic boundary conditions), a rotation by an angle π around an arbitrarily sampled pivot ($\oplus$) is shown: a is the original configuration, b the rotated copy. The intermediate pictures show the superposed system of original and copy before and after the flip. The final configuration, b, is also shown. Notice that the transformation maps the simulation box (with periodic boundary conditions) onto itself. If this is not the case, the treatment of boundary conditions becomes more involved, and generates rejections.

on each plate. This setting can also have important applications, it is very closely related to Gibbs ensemble simulations and provides an optimal way of exchanging particles between two plates. The two plates would no longer describe the same system but would be part of a larger system of coupled plates.

```
algorithm pocket-cluster
begin
  $\mathbf{r}_{\text{pivot}}$ := random point in box;
  $i$ := random particle;
  $\mathcal{P} := \{i\}$;
  $\mathcal{O} :=$ {all particles} $\setminus \{i\}$;
  while $\mathcal{P} \neq \{\}$ do
  begin
    $i$ := any element of $\mathcal{P}$;
    $\mathcal{P} := \mathcal{P} \setminus \{i\}$;
    $\mathbf{r}(i)$ := reflection of $\mathbf{r}(i)$ around $\mathbf{r}_{\text{pivot}}$;
    for $\forall\, j \in \mathcal{O}$ do
    if $j \cap i$ then
    begin
      $\mathcal{O} := \mathcal{O} \setminus \{j\}$;
      $\mathcal{P} := \mathcal{P} \cup \{j\}$;
    end
  end
end
```

Having discussed the conceptual underpinnings of the pivot cluster algorithm, it is interesting to understand how it can be made into a working program. Figure 2.5 suggests that one should use a representation with two plates, and perform cluster analyses, very similar to what is done in the Wolff algorithm.

However, it is not necessary to work with two plates. The transformation can be done on the system itself and does not even have to consider a cluster at all. This ultimately simple solution is achieved in the “pocket” algorithm [10]: it merely keeps track of particles which eventually have to be moved in order to satisfy all the hard-core constraints. After sampling the pivot (or another symmetry operation), one chooses a first particle, which is put into the pocket. At each stage of the iteration, one particle is taken from the pocket, and the transformation is applied to it. At the particle’s new position, the hard-core constraint will probably be violated for other particles. These have simply to be marked as “belonging to the pocket”. One single “move” of the cluster algorithm consists of all the stages until the pocket is empty or, equivalently, of all the steps leading from frame a to frame e in Figure 2.6. The inherent symmetry guarantees that the process will end with an empty pocket, and detailed balance will again be satisfied as the output is the same as in the two-plate version.

In the printed algorithm, $\mathcal{P}$ stands for the “pocket”, and $\mathcal{O}$ is the set of “other” particles that currently do not have to be moved to satisfy the hard-core constraints. The expression $j \cap i$ is “true” if the pair i, j violates the hard-core constraint.

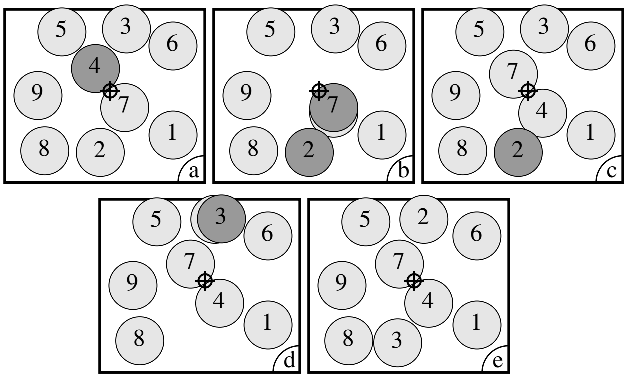


Figure 2.6: One iteration of the pocket algorithm ("pocket" $\equiv$ "dark disks"). Initially (frame a), a pivot is chosen and a starting disk (here disk 4) is put into the pocket. At each subsequent step, a disk is removed from the pocket and transformed with respect to the pivot. Any overlapping disks are added to the pocket. For example, in frame b, overlaps exist between disk 4 (which has just been moved) and disks 2 and 7. Only one of these disks is transformed in frame c. The pocket algorithm is guaranteed to move from a valid hard-disk configuration to another one, and to respect detailed balance. It can be implemented in a few lines of code, as shown in the algorithm on page 15.

2.4 Applications

2.4.1 Phase Separation in Binary Mixtures

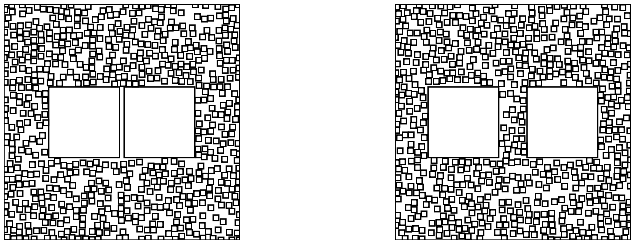


Figure 2.7: Entropic interaction between two colloids (squares of edge length d_{large}) in a sea of small particles (of size d_{small}). Left: Small particles cannot penetrate into the slit between the large particles. The concentration difference leads to an effective entropic interaction between colloids, which is attractive at small separation. Right: At large distances between colloids, the effective interaction vanishes.

The depletion force – one of the basic interactions between colloidal particles – is of purely entropic origin. It is easily understood for a system of large and small squares (or cubes): In the left picture of Figure 2.7, the two large squares are very close together so that no small particles can penetrate into the slit between the large ones. The finite concentration of small squares close to the large squares constitutes a concentration (pressure) difference between the outside and the inside, and generates an osmotic force which pulls the large squares together. The model of hard oriented squares (or cubes) serves as an "Ising model of binary liquids" [3], for which the force is very strong because of the large contact area between them. Besides this, the situation is qualitatively similar to the one for hard spheres.

For a long time, there was a dispute as to whether the depletion interaction (which is oscillatory – repulsive and attractive, starting with an attractive piece at small distances) was sufficiently strong to induce phase transitions. The situation has been cleared up recently due to experimental, analytical and numerical work.

We want to perform Monte Carlo simulation on this system [1, 2]. But as one can see immediately, this is not simple: While the small squares may move with a local algorithm of Figure 2.1, the large particles suffer from a serious "pope in the crowd" effect: The large square is surrounded by so many small particles in its immediate neighborhood that any trial move will somewhere lead to the violation of the hard-core constraint, i.e., it will be rejected. A local Monte Carlo algorithm has vanishing acceptance rate for the motion of the large particles in the limit of $d_{\text{small}}/d_{\text{large}} \to 0$, expressing the increasing number of constraints in this limit.

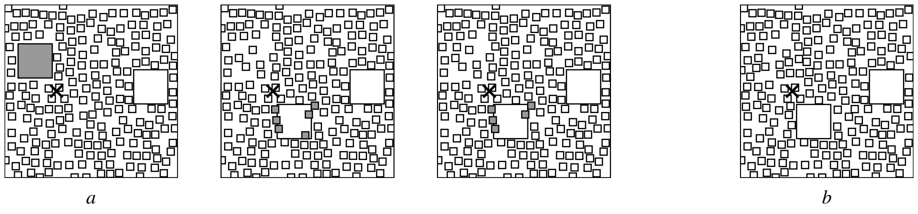


Figure 2.8: Pocket algorithm applied to the binary mixture of squares. The first three stages, and the final configuration of one step are shown. Note that the squares which are fully covered by the moved large square can be transformed immediately, without passing through the pocket, as they will not induce further overlaps.

The pivot cluster method provides a straightforward solution to this problem. Randomly pick a square (large or small), and transform it by applying a symmetry operation of the whole system (rotation around a random pivot, reflection about a symmetry axis of the lattice). At each stage of the algorithm, pick an arbitrary particle from the pocket, transform it and add to the pocket any particles it may overlap with.

As can be seen in Figure 2.8, there is a nice simplification: particles which are completely covered by a "big" particle (as in the second frame of Figure 2.8) will never generate new constraint violations. These particles can be transformed directly, without passing through the pocket.

Using this algorithm, it has become possible to show directly that binary mixtures undergo a phase separation transition, where the large particles crystallize. The transition takes place at smaller and smaller densities as the size mismatch $d_{\text{large}}/d_{\text{small}}$ increases at, say, constant ratio of densities. At the same time, the percolation threshold of the combined two-plate system is sensitive only to the total density of particles.

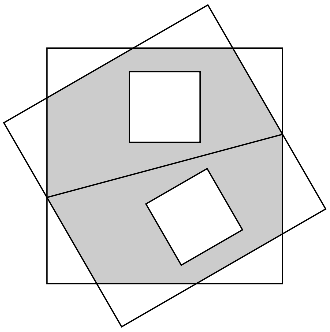


Figure 2.9: The figure shows a transformation (reflection about a straight line) which is not a symmetry transformation of the whole system (with periodic boundary conditions). In this case, cluster transformations involving squares outside the gray area have to be rejected. Transformations, as the ones shown, allow arbitrary orientations of the squares.

It is also possible to relax the "orientation" constraint. This can be done with transformations T which satisfy $T^2 = 1$, but are not symmetries of the simulation box. An example is shown in Figure 2.9.

2.4.2 Polydisperse Mixtures

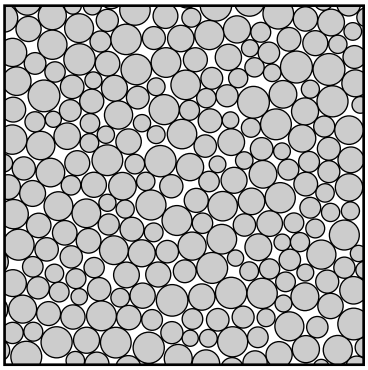


Figure 2.10: Dense configuration of polydisperse hard disks, for which the time evolution of a local Monte Carlo algorithm is immeasurably slow. The cluster algorithm remains ergodic at this density and even higher ones.

At several places in this chapter, the juxtaposition of spin systems with hard spheres has lead to fruitful analogies. One further analogy concerns the very origin of the slowdown of the local algorithm. In the Ising model, the critical slowing down is clearly rooted in the thermodynamics of the system close to a second-order phase transition: the distribution of the magnetization becomes wide, and the random walk of the local Monte Carlo algorithm acquires a long auto-correlation time.

The situation is less clear, even extremely controversial, for the case of hard-sphere systems. It is best discussed for polydisperse mixtures, which avoid crystallization at high densities. In Figure 2.10, a typical configuration of polydisperse hard disks is shown at high density, where the time evolution of the local Monte Carlo algorithm is already immeasurably slow. This system behaves like a glass, and it is again of fundamental interest to study whether there is a thermodynamic explanation for this, or whether the system slows down for purely dynamic reasons.

In the spin problem, the cluster algorithms virtually eliminate critical slowing down. These algorithms are the first to allow precision measurements of thermodynamic properties close to the critical point. The same has been found to apply for polydisperse hard disks, where the pivot cluster algorithm and its variants allow perfect thermalization of the system up to extremely high densities, even much higher than those shown in Figure 2.10. As is evident from the figure, the two-plate system is way beyond the percolation threshold, and one iteration of the cluster algorithm probably involves a finite fraction of all particles. The small clusters which are left behind lead to very useful moves and exchanges of inequivalent particles.

Extensive simulations of this system have given no indications of a thermodynamic transition. For further discussion, see [12, 13].

2.4.3 Monomer-Dimer Problem

Monomer-dimer models are purely entropic lattice systems packed with hard dimers (dominoes) which each cover two neighboring sites. The geometric cluster algorithm provides an extremely straightforward simulation method for this system, for various lattices, and in two and higher dimensions [10]. In this case, the "clusters" have no branches. For the completely covered dimer system (in the two-plate representation), the clusters form closed loops, which are symmetric under the transformation. These loops can be trivially generated with the pocket algorithm and are special cases of transition graph loops used in other methods.

Care is needed to define the correct symmetry transformations, see Figure 2.11. For example, a pure rotation by an angle π would leave the orientation (horizontal, vertical) of each dimer unchanged, and conserve their numbers separately. On a square lattice of size $L \times L$, the diagonals are symmetries of the whole system. It has been found that reflections about all symmetry axes on the square or triangular lattice lead to an ergodic algorithm. The reasoning can be extended to higher dimensions [8]. It is very interesting to observe that, in any dimension, the cluster can touch the symmetry axis (or symmetry hyperplane) twice at most. This implies that symmetry axes (or their higher dimensional generalizations) will not allow the cluster to fill up the whole system. For a detailed discussion, see [10].

2.5 Limitations and Extensions

As with other powerful methods, the pivot cluster algorithm allows to solve basic computational problems for some systems, but fails abysmally for the vast majority. The main reason

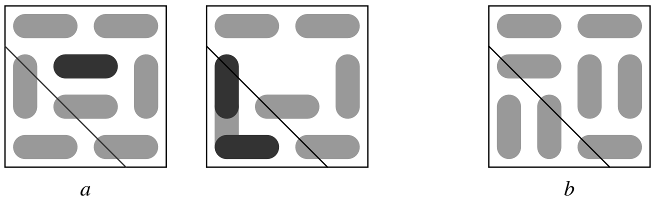


Figure 2.11: Application of the pocket algorithm to a dimer-configuration on the two-dimensional square lattice. In this problem, the maximum size of the pocket is 2. The initial configuration a, the configuration after the first transformation, and the final configuration b are shown.

for failure is the presence of clusters which are too large, in applications where they leave only "uninteresting" small clusters.

This phenomenon is familiar from spin-cluster algorithms, which, for example, fail for frustrated or random spin models, thus providing strong motivation for many of the combinatorial techniques presented elsewhere in this book. Clearly, a single method cannot be highly optimized for a general application.

In the first place, the cluster pivot algorithm has not improved the notoriously difficult simulations for monodisperse hard disks at the liquid–solid transition density. This density is higher than the percolation threshold of the combined two-plate system comprising the original and the copy. Nevertheless, one might suppose that the presence of small clusters would generate fast non-local density fluctuations. Unfortunately, this has not been found to have much impact on the overall convergence times. A clear explanation of this finding is missing.

Another frustrating example is the Onsager problem of liquid crystals: hard cylindrical rods with diameter D, and length L, which undergo an isotropic–nematic transition at a volume fraction which goes to zero as the rods become more and more elongated [6].

$$\rho_{\text{iso}} \sim 3.3\frac{D}{L} \quad \text{for } D/L \to 0 \tag{2.7}$$

This is analogous to what we found for binary mixtures, where the transition densities also go to zero with the ratio of the relevant length scales, and one might think that the cluster algorithm should work just as well as it does for binary mixtures.

Consider, however, a cube with edges of length L, filled with density ρ_{iso} of rods, see Figure 2.12. The question of the percolation threshold translates into asking what is the probability of another, identical, rod hitting one of the rods in the system.

$$\begin{aligned}
\text{Volume of rods in cube of size } L^3 &\sim 3.3\ DL^2 \\
\text{Number of rods } &\sim \frac{13.2}{\pi}\ L/D \\
\text{Surface } &\sim \frac{13.2}{\pi}\ L^2
\end{aligned}$$

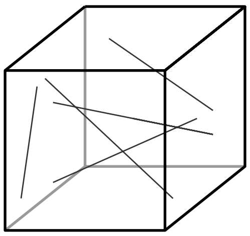


Figure 2.12: Hard rods of length L and diameter D in a test box of dimensions L^3. At the critical density for the isotropic–nematic transition, the volume fraction occupied by the rods goes to zero, but the cube is still opaque. This is due to the fact that the surface of a very thin object ($\sim LD$) is much larger than its volume ($\sim LD^2$).

During the performance of the cluster algorithm, an external rod will be moved into the test cube from elsewhere in the system. It is important that it does not generate a large number n_{rods} of violations of the hard-core constraint with rods in the cube. We can orient the test cube such that the new rod comes in "straight" and find that the number is given as

$$n_{\text{rods}} \sim \frac{\text{surface of rods in test cube}}{\text{surface of test cube}} \sim 4.2 \tag{2.8}$$

This is what was indeed observed: the exterior rod will hit around 4 other rods, so this means that this system is far above the percolation threshold $n_{\text{rods}} = 1$, and the cluster will contain essentially all the rods in the system.

The pivot cluster algorithm has been used in a series of studies of more realistic colloids, and has been extended to include a finite potential, in addition to the hard-sphere interaction [11].

Finally, the pivot cluster algorithm has been very successfully applied to the Ising model with fixed magnetization, where the number of "+" and "−" spins are separately conserved. This is important in the context of lattice gases, which can be mapped onto the Ising model [7].

Acknowledgments

I would like to thank C. Dress, S. Bagnier, A. Buhot, L. Santen, and R. Moessner for stimulating collaborations over the last few years.

References

[1] A. Buhot, W. Krauth, *Numerical Solution of Hard-Core Mixtures*, Phys. Rev. Lett. **80**, 3787 (1998).

[2] A. Buhot, W. Krauth, *Phase Separation in Two-Dimensional Additive Mixtures*, Phys. Rev. E **59**, 2939 (1999).

[3] J. A. Cuesta, *Fluid Mixtures of Parallel Hard Cubes*, Phys. Rev. Lett. **76**, 3742 (1996).

[4] C. Dress, W. Krauth, *Cluster Algorithm for Hard Spheres and Related Systems*, J. Phys. A: Math Gen. **28**, L597 (1995).

[5] H. G. Evertz, *The Loop Algorithm*, Adv. Phys. **52**, 1 (2003).

[6] P. G. de Gennes, *The Physics of Liquid Crystals*, (Oxford University Press, 1974).

[7] J. R. Heringa and H. W. J. Blöte, *The Simple-cubic Lattice Gas with Nearest-neighbour Exclusion: Ising Universality*, Physica **232A**, 369 (1996).

[8] D. A. Huse, W. Krauth, R. Moessner, S. L. Sondhi, *Coulomb and Liquid Dimer Models in Three Dimensions*, Phys. Rev. Lett. **91**, 167004 (2003).

[9] W. Krauth, *Statistical Mechanics: Algorithms and Computations*, (Oxford University Press, 2004).

[10] W. Krauth, R. Moessner, *Pocket Monte Carlo Algorithm for Classical Doped Dimer Models*, Phys. Rev. B **67**, 064503 (2003).

[11] J. G. Malherbe, S. Amokrane, *Asymmetric Mixture of Hard Particles with Yukawa Attraction Between Unlike Ones: a cluster algorithm simulation study*, Mol. Phys. **97**, 677 (1999).

[12] L. Santen, W. Krauth, *Absence of Thermodynamic Phase Transition in a Model Glass Former*, Nature **405**, 550 (2000).

[13] L. Santen, W. Krauth, *Liquid, Glass and Crystal in Two-dimensional Hard disks*, cond-mat/0107459.

[14] U. Wolff, *Collective Monte Carlo Updating for Spin Systems*, Phys. Rev. Lett. **62**, 361 (1989).

3 Probing Spin Glasses with Heuristic Optimization Algorithms

Olivier C. Martin

Finding the ground state of an Ising spin glass corresponds to determining the maximum cut in a graph, a prominent problem in combinatorial optimization. Understanding the physical properties of this kind of system is a long-standing challenge, and developing better ground-state solvers should have a large impact. After providing a general introduction to spin glasses, we cover some heuristic algorithms that allow one to tackle these systems. We also stress some of the open problems that one can hope to resolve in the next few years.

3.1 Spin Glasses

3.1.1 Motivations

Spin glasses were discovered in the early 1970s. For these magnetic materials, the response to an oscillating external field is singular: in the limit of small frequencies, the susceptibility has a cusp at a critical temperature T_c. This behavior is qualitatively different from ordinary magnetic materials, and can be interpreted as being due to a "freezing" of the magnetic dipoles in the samples. To really understand the physics of spin glasses, one needs to master magnetism, phase transitions, and numerous peculiarities specific to spin glasses. The interested reader is referred to the book "Spin Glasses" by Fischer and Hertz [11] which gives a nice overview, covering, in particular, the history of the subject.

The exotic properties of a spin glass are believed to be controlled by its low-energy configurations, a configuration being the specification of the orientation of each of the magnetic dipoles. The physical origin of these dipoles is the "spin" of the electrons as in ordinary magnetism, but it is the "irregular" nature of the couplings between these spins that gives these materials their striking properties. Although the *source* of these couplings is quantum mechanical, the spins can be treated classically using an ordinary classical Hamiltonian that gives the energy of each configuration. Thus to understand most of the phenomenology of spin glasses, it is enough to find this Hamiltonian's ground state and low-energy excitations: that is why much effort has focused recently on applying optimization algorithms to spin glasses. Furthermore, spin glasses are considered to be the archetypes of complex systems so that what is learned here is expected to have a strong impact on our understanding of many other systems with competing effects.

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

3.1.2 The Ising Model

Let us start with the Ising model which historically was introduced to describe ferromagnetism. When the spin of an electron is treated classically, it becomes a vector of constant length, only its direction is variable. Generally, this vector can point anywhere in space, so if we represent it in Cartesian coordinates, we have a three-component vector $\vec{S}$ of fixed size that is specified by its three projections S_x, S_y and S_z. However, in many materials, the crystalline arrangement of the atoms induces preferential directions; for instance if the atoms form a cubic lattice, the axes of the lattice become special directions. In such cases, it is energetically favorable to have the spin align with these directions, and so at low temperature the other possible orientations of the spins become irrelevant. Now in fact there are materials where just *one* axis dominates the others; then a spin will be oriented along that axis, either positively or negatively. Without loss of generality, we shall label this sign S, i.e., $S = \pm 1$. In the 1920s, Wilhelm Lenz and his Ph.D. student Ernst Ising introduced a statistical physics model[1] for this case: they placed an "Ising" spin $S_i = \pm 1$ on each lattice site i and introduced spin-spin interactions through nearest-neighbor ferromagnetic couplings; the corresponding Hamiltonian is

$$H_{\text{Ising}} = -J \sum_{\langle ij \rangle} S_i S_j \quad \text{where} \quad \text{J} > 0\,. \tag{3.1}$$

Here J is the coupling strength and the summation over $\langle ij \rangle$ means sum over all nearest-neighbor pairs of sites. H_{Ising} thus gives the energy of an arbitrary configuration and is a sum of terms involving only two spins at a time. Since $J > 0$, the interaction energy of two spins is lowest when the two spins are of the same sign.

At high temperature the spins fluctuate independently, while at low temperature, thermal fluctuations are suppressed and the system is dominated by the configurations of lowest energy. Clearly the absolute lowest of these have $S_i S_j = 1$ for all i and j, i.e., all spins are of the same sign. In these configurations one has a net magnetization per site: $m = \sum_i S_i / \sum_i 1$, a quantity that plays the role of an order parameter in statistical physics. At high temperature, $m = 0$, while at low temperature, $m \approx \pm 1$. This is indeed what happens for any lattice of dimension $d \geq 2$. The irony of history is that Ising studied [28] the one-dimensional case; he showed that $m = 0$ as soon as any thermal fluctuations are allowed (i.e., whenever the temperature is non-zero), from which he concluded that understanding ferromagnetism required a different approach.

3.1.3 Models of Spin Glasses

When we consider spin-glass materials, two major differences arise, compared to the simple ferromagnetic case just described. First, the couplings J_{ij} among spins are disordered, i.e., they have many values and no underlying periodicity is present. (The lack of periodicity is due to the fact that the spins are associated with just some of the atomic species forming the lattice, and the arrangement of these particular atoms is random.) One says that the system has *quenched* disorder; indeed, the spins are fixed in position (but not in orientation), and thus

[1] The reader may benefit from reading Chapter 5 which also describes this model.

the J_{ij} are also fixed or "quenched". Second, the J_{ij} may take on negative values; then at low temperatures, two neighboring spins S_i and S_j either prefer to align (if $J_{ij} > 0$) or to anti-align (if $J_{ij} < 0$). This generally translates into what is called *frustration* [54]: it is not possible to simultaneously minimize all the $-J_{ij}S_iS_j$ terms, the interactions of a given spin with its neighbors being antagonistic. A simple example of this arises when considering three spins coupled anti-ferromagnetically to one another: as illustrated in Figure 3.1, the top spin is subjected to contradictory forces.

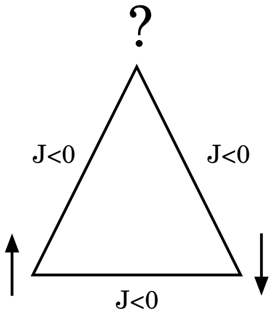


Figure 3.1: Three spins coupled antiferromagnetically. If the bottom spins are set according to their mutual interaction, then the top spin is subject to antagonistic forces.

When this kind of competition arises with *many* spins, it is not clear what are the lowest energy configurations, and thus we no longer know how the system should behave at low temperature. Finding these low-energy configurations algorithmically is then a good starting point for understanding the low-temperature properties of spin glasses.

To add quenched disorder and frustration to the Ising model, Edwards and Anderson [9] kept the spins on the regular square or cubic lattice, and simply made the couplings J_{ij} random independent variables:

$$H_{\mathrm{EA}} = -\sum_{\langle ij \rangle} J_{ij} S_i S_j \; . \tag{3.2}$$

They argued that the orientations of the spins should "order" at low temperatures, but not as in the Ising model; instead, the orientations would seem random and would follow from minimizing H_{EA}. Furthermore, the magnetization per site should be zero, while the magnetization at each site should be non-zero. Much of the theoretical research on spin glasses since [1] has confirmed these ideas, but it has been difficult to make firm claims accepted by all, to a large extent because numerical computations are restricted to small sizes and theoretical approximations are of limited reliability.

There is, however, one model of spin glasses where there is a consensus. Sherrington and Kirkpatrick [53] proposed to take infinite range rather than nearest-neighbor interactions in H_{EA}; one calls this limit[2] the SK model In 1979, Parisi proposed a variational solution [51] to this model, and it is now widely believed that his solution is exact. This solution predicts

[2] The couplings have to be scaled so that thermodynamic quantities such as energy and free energy remain extensive.

very striking properties for the set of low-energy configurations as will be detailed in the next section. It is of real interest to know whether these exotic features are artifacts of the infinite-range interactions or remain valid in the original Edwards–Anderson (EA) model. Indeed, one of the main controversies amongst spin-glass theorists is whether the Parisi or "mean-field" picture is appropriate for finite-range interactions. There are other competing pictures, the main one being based on a real-space renormalization group approximation called the droplet/scaling picture [5, 12] as will be discussed later.

More recently, some theoretical progress [40] has been made in understanding models intermediate between the SK and the EA models, namely spin glasses on random graphs of finite connectivity. Since the interactions remain at infinite range, it is not so surprising that these models have properties rather close to those of the SK model.

3.1.4 Some Challenges

Let us focus on the EA model in dimension 3 as the stakes there are the highest. The definition of the model (cf. Eq. (3.2)) is very compact; furthermore, the spin variables have an elementary nature since they are Ising, $S_i = \pm 1$. It thus seems unbelievable that even the *qualitative* features of this system are still subject to debate. Some of the corresponding issues such as dynamics or critical properties at T_c fall at the heart of statistical physics. Of interest to us here are the unsettled issues directly related to optimization, namely the nature of the ground state and its low-energy excitations. Let us thus look at these two issues more closely.

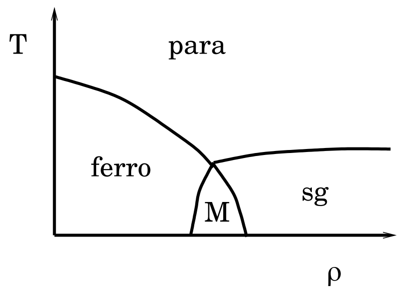
3.1.4.1 Phase Diagram

In most studies, the J_{ij} in Eq. (3.2) are taken from a distribution symmetric about 0: there are as many ferro as anti-ferromagnetic interactions. If instead $J_{ij} < 0$ only for a small fraction of the couplings, the system is ferromagnetic and resembles the ordinary Ising model. As the fraction of anti-ferromagnetic couplings is increased beyond a critical value, the ferromagnetism disappears. Does the spontaneous magnetization vanish just as the spin-glass ordering appears? Can there be a "mixed" phase where the two types of ordering co-exist? Such a phenomenon is expected within the mean-field picture [41] as is illustrated in Figure 3.2. On the contrary, the droplet/scaling picture says no such coexistence can arise [42]. Only recently has this question been considered numerically [33] and not surprisingly the issue is not settled.

An analogous situation appears when an external perturbation in the form of a magnetic field is applied to the spins. The modified Hamiltonian becomes

$$H_{\text{EA}} = -\sum_{\langle ij \rangle} J_{ij} S_i S_j - B \sum_i S_i \, . \tag{3.3}$$

The coupling to B biases the sign of the spins. In the droplet/scaling picture, the spin-glass ordering is destroyed as soon as $B \neq 0$, leading to a paramagnetic phase [5, 12]. On the contrary, in the mean-field picture, spin-glass ordering *co-exists* with the net magnetization induced by B as long as B is not too large. (This can be generalized to any temperature, the transition curve being the so-called de Almeida–Thouless [7] or "AT" line as illustrated in Figure 3.2.) Numerical studies [24, 31, 35] suggest that no co-existence occurs in three dimensions, but further work is necessary.



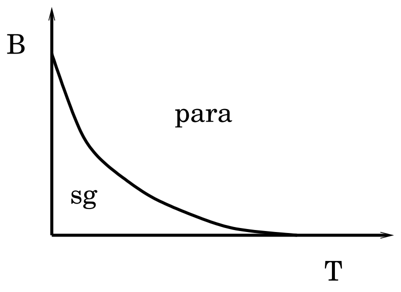


Figure 3.2: The phase diagrams expected within the mean-field picture. Left: mixed phase (M) where ferromagnetism and spin glass ordering co-exist. Right: the AT line separates a paramagnetic (disordered) phase and the spin-glass phase. T is the temperature, ρ the concentration of $J_{ij} < 0$, and B the intensity of the magnetic field.

3.1.4.2 Energy Landscape

Now consider the organization of the low-energy configurations in the EA model. Can one give a statistical description whereby some of their properties "scale"? By this we mean for instance that a characteristic energy has a power-law scaling with N as $N \to \infty$. (N is the number of spins in the system, which can be identified with the "volume" of the system; $N \to \infty$ is the "thermodynamic" limit.) To bring out some possibilities, let us first outline the predictions from the mean-field picture. For that, we list some of the properties arising in the SK model:

- *Clustering*
  Given a low-energy configuration, it is possible to further flip 1, 2, and more spins if they are carefully chosen without changing substantially the value of the excitation energy. In the Parisi solution, the set of low-energy configurations form families or "clusters"; two configurations belong to the same cluster if their Hamming distance is very small compared to N. (The Hamming distance of two configurations is the number of spins that are set differently.)

- *Replica symmetry breaking*
  Among the clusters formed by the low-energy configurations, consider the ones in which a finite fraction x of the spins are flipped compared to the ground state. (By finite we mean that x is fixed, $0 < x < 1$, and then one focuses on the large-N limit with that given x.) In the SK model, the corresponding *lowest* excitation energy is $O(1)$; note that this energy scale is the same as that for flipping a single spin chosen at random! Furthermore, the SK has *continuous* replica symmetry breaking, meaning that x can take on values that span at least some sub-interval of $[0, 1]$.

- *Ultrametricity*
  One defines the distance between two configurations as their Hamming distance divided by N. One can also define the distance $d(\alpha, \beta)$ between two *clusters* of configurations from the mean distance of their respective members. In the SK model, it

turns out that the low-energy clusters are organized hierarchically: each cluster is divided into sub-clusters which are themselves sub-divided... Furthermore, if we think of clusters as being points in an abstract space, that space is *ultrametric*, i.e., all triangles are isosceles. Mathematically, this means that for any three "points" (α, β, γ), we have $d(\alpha, \gamma) \leq \max[d(\alpha, \beta), d(\beta, \gamma)]$. Such a space can be mapped to the leaves of a tree and then the distance between two leaves is just the height of the smallest sub-tree containing both of them.

It is a major challenge to understand whether these properties also arise in the three-dimensional EA model. Of course they may not; for instance the droplet/scaling picture may hold instead. In that picture, scaling laws play a central role but so do "position-space" properties. At the heart of these is the notion of *droplet* excitations above the ground state; a droplet is defined on a given scale ℓ and around a given spin S_0 as follows. One considers all connected clusters of spins that are enclosed in the cube of side 2ℓ centered on S_0 but which are not contained in that of side ℓ; among all these potential excitations, the droplet is the one of lowest energy.

In the droplet/scaling picture, one postulates the following properties:

- Droplet energies grow as a positive power of their volume; thus if one flips $O(\ell^3)$ spins in a given region, the corresponding *lowest* energy grows as ℓ^θ. (An early numerical estimate [5] suggested that $\theta \approx 0.2$.)
- There is no replica symmetry breaking.
- The organization of the low-energy clusters is not hierarchical; nevertheless, the energy landscape is self-similar, i.e., it is a fractal in which the exponent θ determines the scaling laws.

There are also a number of other issues that can be considered. For instance, one can ask what is the "geometry" of the low-energy excitations; in particular, are they compact, rough, topologically complicated...? Another class of problem relates to how the energy landscape is modified under perturbations of the Hamiltonian. For instance, there is now a consensus that the ground state and also excited states are very sensitive to changes in the couplings J_{ij}: if one changes all the couplings even by a very small amount, the ground state will change dramatically, a phenomenon referred to as "chaos". At present, however, there is no consensus regarding chaos in the presence of a magnetic field.

3.2 Some Heuristic Algorithms

3.2.1 General Issues

It is convenient to divide algorithms for finding ground states into two classes: "exact" or complete, and "heuristic" or incomplete. In the first class, the algorithms will terminate and will definitely provide the minimum energy solution. For NP-hard problems [13] such as spin glasses (Max-Cut), exact algorithms [50] include branch and bound, branch and cut, and of course, exhaustive search. In all cases though, the worst-case computation times grow exponentially with the problem size N. Much progress has been made in designing effective exact algorithms and most probably further improvements will come. The status of these approaches for spin glasses is given in the Chapter 4.

Our focus here is on *heuristic* algorithms: such algorithms provide good but not necessarily optimal solutions. They should be used to find ground states only if one can measure their reliability, i.e., if can one convince oneself that the failures to find the true optima arise so rarely that they make no difference for the study. Given this drawback, heuristics have the advantage of being easy to program and also of being very fast; this often allows researchers to quickly tackle large systems with little pain. We begin here with some general remarks that go beyond the use of heuristics for spin glasses. The reader may also consult several of the chapters of the book "Optimization Algorithms in Physics" [21] for additional material.

3.2.1.1 Reliability Tests

Given an heuristic algorithm, we want to find out how often it fails to find the true ground state, and when it does, what the size of the error in the energy will be.

First suppose that you also have an exact algorithm; then you can compare directly, measuring the heuristic's failure frequency and the associated error in the ground-state energy. Unfortunately, you can do this only on the samples for which the exact algorithm terminates! The whole point of resorting to an heuristic is to use it on samples where you cannot use the exact algorithm. The way out here is to extrapolate. Assume that you have measured the failure rate on spin glasses having up to $N = 100$ spins; then guess an appropriate law for the N dependence of this failure rate and apply this law to all N. If the law works well for $N \leq 100$, it is "reasonable" to perform this extrapolation.

Second, suppose instead that you have no exact algorithm, or that the one you have allows you only to go up to $N = 30$ spins; then extrapolating to $N = 1000$ seems a bit dangerous. We thus seek a self-consistent test of the heuristic algorithm on its own. All the heuristics we consider operate by improving configurations (feasible solutions), and they require one or more configurations as input. Both this input and the search for improving configurations can be randomized. It is then possible to perform *independent multiple starts* of the algorithm. As an illustration, assume one has performed 10 independent runs of the algorithm. If all 10 outputs are identical, there is good reason to expect that the true ground state has been found. If, on the contrary, the 10 outputs do not all agree, the heuristic is not so reliable. Generally, it seems unavoidable for heuristics to become unreliable at large N; the challenge is to postpone this bad behavior for as long as possible.

Let us illustrate these points using a specific example. We ran an heuristic called Kernighan–Lin [30] (hereafter referred to as KL) on Ising spin-glass samples of increasing sizes N. Historically, KL is the first "variable depth search" heuristic (see Section 3.2.2 for what this means; also, for the kinds of samples used in this illustration, see [52].) Figure 3.3 shows the distributions of the energies per spin found when using random starts: for each size ($N = 20$, 50, 100 and 170), the histogram is for a single sample, i.e., a single choice of the quenched disorder.

In the first case ($N = 20$), the distribution is completely dominated by the probability of finding the lowest energy E_0 (which presumably is that of the ground state), $P(E_0) \approx 0.98$. This problem size is thus extremely simple to "solve". Increasing the size to $N = 50$, we find that the distribution still peaks at E_0 but much less than before: $P(E_0) \approx 0.35$. Going on to

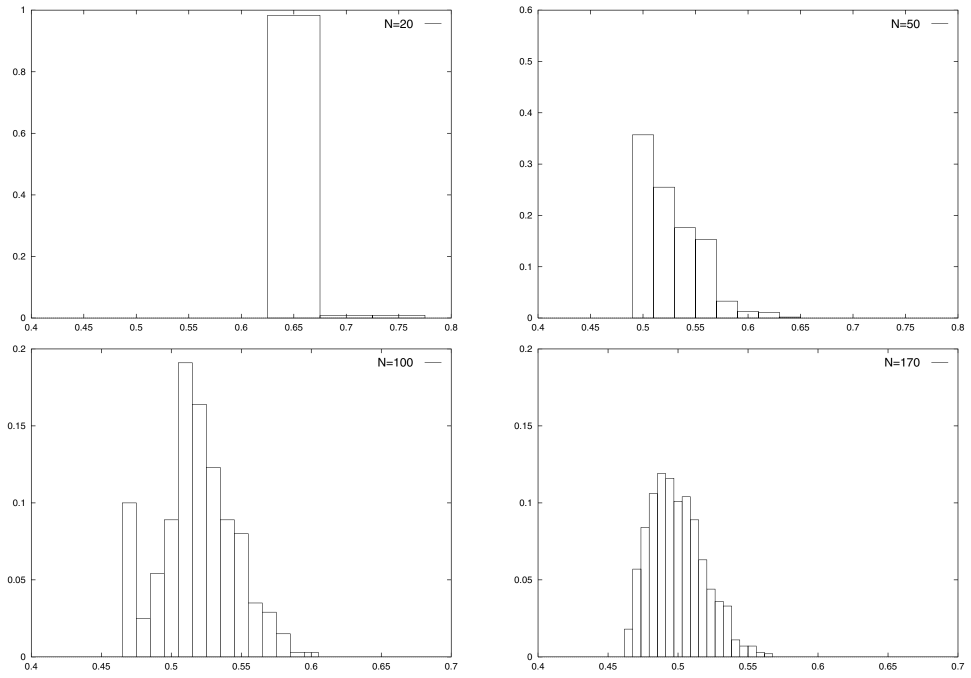


Figure 3.3: Distribution of energies per spin for the KL algorithm applied to four samples with $N = 20, 50, 100$, and 170 spins.

$N = 100$, the distribution becomes bimodal, the main peak occurring significantly above the ground-state energy. Finally, for $N = 170$, the peak at the minimum energy has disappeared and the distribution has become bell shaped. All in all, as N increases, the quality of the algorithm, as measured for instance by $P(E_0)$, deteriorates very clearly.

None of this is specific to our combinatorial optimization problem nor to the heuristic for treating it. Most generally, a smart heuristic finds the optimum quickly and with a probability close to 1 when N is "small"; in fact, often the error rate is unmeasurable. Increasing N, one finds some samples for which errors appear. When N grows still more, samples typically lead to errors but by repeating the runs enough times we can still hope to extract the ground state. Finally, for very large N, we *never* find the ground state in practice. (A signal that this is the case is when the outputs are all different when using multiple starts.) Any heuristic algorithm will follow this pattern of "simple" to "difficult" as N grows. Such behavior is generic; the main difference from one heuristic algorithm to another is the typical value of N where this cross-over from simple to difficult arises. The more powerful the algorithm, the later in N the cross-over occurs and the greater the range in which the heuristic can be used to find the ground state with a high level of reliability.

3.2.1.2 Large-N Scaling

When N becomes very large, scaling laws set in. First, the *distribution* of the energies generated from multiple starts becomes Gaussian [52]. Obviously when this happens, the heuristic is of no value if we want to extract the true ground-state energy; indeed, E_0 is then far in the tail, and the probability that the heuristic finds E_0 goes to zero! This brings us to the second scaling law, namely how $P(E_0) \to 0$ as N increases. To make this a well defined question, we first average this probability over different samples to make the sample dependence disappear.

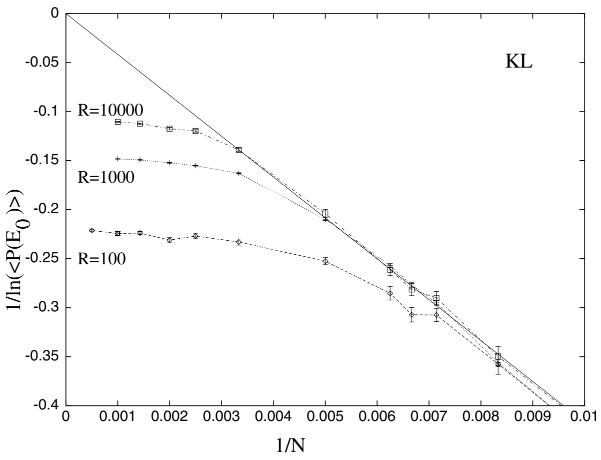


Figure 3.4: The mean probability of finding the "optimum" using the KL algorithm as a function of N and R. The straight line is the exponential fit.

We show, in Figure 3.4, results when using the same KL algorithm as before. For each value of N, we have averaged over 100 to 1000 disorder samples. For each sample, we used R independent random starts and defined E_0 to be the lowest energy found in any of the R restarts; we then estimated $P(E_0)$ from the number of runs finding E_0. Because R is finite, our estimator for $\langle P(E_0)\rangle$ is biased, and can only be reliable when $R\langle P(E_0)\rangle \gg 1$. When this condition is not satisfied, the sampling is inadequate, and we will necessarily find that "best-found" has a frequency near $1/R$. Figure 3.4 shows the resulting estimators of $\langle P(E_0)\rangle$ for $R = 100, 1000, 10\,000$. The correct (unbiased) $\langle P(E_0)\rangle$ function is the $R \to \infty$ envelope. The curves show both the cross-over to the $\mathcal{O}(1/R)$ behavior and also that there is a large-R limit at fixed N. We also see that the algorithm is useful for finding ground states at $N \leq 100$ (using multiple starts) but becomes very unreliable at $N \geq 200$. This result agrees with the conclusions drawn from Figure 3.3. Since we expect $\langle P(E_0)\rangle \sim e^{-\gamma N}$, we have plotted $1/\ln\langle P(E_0)\rangle$ as a function of $1/N$. Although it is difficult to take the large-N limit, we do see that the behavior of the envelope agrees with the expected exponential law. (Note that $-1/\ln[R] = -0.217, -0.145$ and -0.109 for the different R; the onset of an $\mathcal{O}(1/R)$ behavior indicates that E_0 as defined is probably not the true ground-state energy.)

3.2.1.3 Rules of Thumb

Much has been written about developing good heuristics in combinatorial optimization problems. In practice, the more one incorporates problem-specific knowledge, the more effective the heuristic can be. Another question is whether it is better to use a sophisticated heuristic or to use multiple independent starts of a fast but less sophisticated heuristic. If the ratio of the times used by these algorithms does not grow exponentially fast in N, then at large N the more sophisticated algorithm will almost always be more effective. Indeed, as was mentioned before, the distribution of energies becomes Gaussian but more importantly the *relative* fluctuations of those energies go to zero as $1/\sqrt{N}$. (When the relative fluctuations of a quantity tend to zero, one says that it is "self-averaging".) Asymptotically, the distribution of energies found by the two heuristics will have zero overlap, and the one with the peak at the lowest value, namely the more sophisticated one, will win. Furthermore, it is expected, and simulations confirm [52], that each heuristic leads to energies that are a fixed percentage above the true ground-state value when $N \to \infty$. Getting the ground state in that limit is then impossible because it requires obtaining events that are far in the tails of the distributions.

3.2.2 Variable Depth Search

We now move on to describe algorithmic aspects. Although much of what follows applies to general combinatorial optimization problems, the presentation will be given in the framework of Ising spin glasses. Our space of feasible solutions consists of all configurations. (Recall that a configuration here is the specification of each of the spin orientations, $S_i = \pm 1$, $i = 1, \ldots, N$ in a system of N Ising spins.) It is useful to define a neighborhood structure in this space as follows: a configuration $\mathcal{C}'$ is a neighbor of a configuration $\mathcal{C}$ if and only if $\mathcal{C}'$ can be reached from $\mathcal{C}$ by flipping a single spin, or more generally by flipping, at most, k spins, where k is fixed and "small".

3.2.2.1 Local Search

The optimization problem is to find the lowest energy configuration; there are 2^N configurations, so searching for the best one is generally like searching for a needle in a hay stack. *Local Search* (LS) consists of exploring the search space by repeatedly hopping from one configuration to the next, restricting each hop to be between *neighboring* configurations. The first such algorithm is due to Lin [36] who proposed to perform a hop if and only if it lowered the energy; one then has an iterative improvement scheme. With this restriction, after a finite number of hops, one reaches a configuration whose energy is lower or equal to that of all of its neighbors; the hopping process then stops and we have found a "local minimum". This process is sometimes called k-Opt as it uses hops that change, at most, k variables at a time, hereafter referred to as k-changes. For our spin-glass problem, we can go a bit further and notice that the energy change when flipping k spins is the sum of the changes for flipping each spin on its own *unless* a coupling J_{ij} connects two of them. This means that when performing k-Opt, we can stop if we have reached a local minimum under *cluster* changes of, at most, k spins. By a cluster we mean a set of spins that forms a connected set, i.e., cannot be broken into sub-clusters between which there are no J_{ij} couplings. k is referred to the "depth" of the search because only k-changes are allowed in each hop. Since one must consider all possible k-changes, it is natural to organize the k-Opt algorithm as follows:

```
algorithm $k$-Opt
begin
    Initialize $\mathcal{C}_0$    Starting configuration
    at_local_min := FALSE;
    while (at_local_min = FALSE) do
        at_local_min:=TRUE;
        for site $= 1, \ldots, N$ do
            if (Find_Favorable_k_Change(site) $< 0$) at_local_min:=FALSE;
        end do
    end while
end
```

Nearly all the CPU time is spent in the search for favorable k-changes. Usually, one simply takes the *first* favorable change found:

```
algorithm Find_Favorable_k_Change(site_0)
begin
    for all connected clusters of $k' \leq k$ sites containing site_0 do
        compute $\delta E$    the energy change when flipping the cluster
        if $\delta E < 0$ then
            flip this cluster;
            return ($\delta E$) ;
    end do
    return (0);
end
```

One can also use instead the *best* k-change rather than the first favorable one found; however, for most problems that strategy does not pay off. Historically, Lin tested k-Opt on the Traveling Salesman Problem and found it to be quite effective; furthermore he found that the quality of the solutions produced improved when increasing k.

These results are rather general, applying to most combinatorial optimization problems and to spin glasses in particular. Finally, note that when $k = 1$, k-Opt corresponds to a zero-temperature simulated annealing algorithm using single spin flips. That algorithm is not effective, giving rise to energies that are typically 20% above the ground state; to do better, one must increase k.

3.2.2.2 The Kernighan–Lin VDS Extension

A few years later, Kernighan and Lin [30] realized that fixing k ahead of time was overly restrictive: it is better to let k increase if the changes seem sufficiently promising. The difficulty is that considering all potential k-changes becomes very time consuming when k grows; in fact the growth is typically exponential in k. Thus one must be "selective" and consider only a few of the many possible k-changes. Kernighan and Lin's (KL) strategy is to string together a sequence of 1 or 2-changes to build changes with variable k. A crucial point is that these building blocks (1 or 2-changes in their case, single spin flips in ours) are *not* imposed to decrease the energy, but are chosen for their “promise”. To reduce the possible choices to a minimum, Kernighan and Lin proposed to apply the greedy criterion of using at each step only the “best” building block, i.e., the one leading to the lowest energy at that step. After a (possibly long) sequence of changes is generated, one sees at what stage it gave rise to its lowest energy configuration $\mathcal{C}^*$; let p^* be the corresponding number of spin flips involved. If the energy of $\mathcal{C}^*$ is lower than the energy at the beginning of the sequence (the configuration when $p = 0$), then one hops to this new improved configuration. The value of p^* is not given *a priori* so this kind of LS is called Variable Depth Search (VDS) or “KL” in honor of its inventors. They implemented it for the graph bi-partitioning problem which is equivalent to finding the ground state of a spin glass with the constraint of zero magnetization.

In our implementation of VDS, we impose the flipped spins to form a connected cluster; thus we start with a seed site, flip it ($p = 1$), and then flip its most favorable neighbor ($p = 2$), and continue like this, always flipping the most promising spin at the *boundary* of the current cluster. Once a spin joins the cluster, it cannot be removed. (Kernighan and Lin call this the taboo condition.) The growth of the cluster continues up to some maximum size (which can be set to optimize the over-all performance of the heuristic). Then one finds p^* and flips the associated cluster of spins if indeed this lowers the energy. Our construction is deterministic, but randomization can be introduced just as in Find_Favorable_k_Change.

Schematically the algorithm works as follows. Given the current configuration $\mathcal{C}$, we pass a seed site to the routine which then searches for a favorable cluster containing that seed:

```
algorithm Greedy_Cluster_Search(site_0)
begin
    best_gain := 0;
    initialize cluster to contain only site_0
    initialize cluster's gain
    while search looks promising do
        add to the cluster that site at its surface with
            the largest gain     (greedy criterion)
        compute the gain of the cluster
        if (gain $>$ best_gain) then
            memorize this cluster;
            best_gain := gain;
    end do
    if (best_gain $>$ 0) flip the memorized cluster;
    return ( - best_gain ) ;
end
```

The gain refers to the *decrease* in the energy, which may be positive or negative. Intuitively, we accept that we must do a bit of climbing out of a valley at the beginning, taking the direction of least resistance, hoping that this will lead us into another valley.

Finally, to be complete, the overall algorithm for VDS is very close to the k-Opt routine previously described, the main change being that Greedy_Cluster_Search replaces Find_Favorable_k_Change. The second important change is that in practice we do not consider all sites as seeds because that would be too costly in CPU time; instead, our VDS stops when (1) no 1-change can lead to improvement; and (2) Greedy_Cluster_Search has found no improvement for the last five calls, each seed site having been chosen randomly.

3.2.2.3 Data Structures and Computational Complexity

We estimate the computational complexity of the VDS algorithm in terms of the time it takes to perform the greedy search for a cluster (Greedy_Cluster_Search). We also assume that the search in that routine proceeds until all N sites are captured. For sparse graphs, each such search requires $\mathcal{O}(N \ln N)$ operations if one uses heaps in which the candidate sites (at the boundary of the current cluster) are stored. Heaps allow one to obtain the site with the largest gain in a time growing only logarithmically with the number of sites in this heap. (In the case where the energies are integers rather than real numbers, one can appeal to a faster sorting strategy such as radix sort to get a logarithmic gain in speed [10].)

In our practice, the overall algorithm requires just a few calls to Greedy_Cluster_Search before no further improvements are possible; in particular the number of calls grows rather slowly with the problem size N. Neglecting this small effect, we obtain a complexity of $\mathcal{O}(N \ln N)$ for our VDS.

3.2.2.4 Benchmarks

To stay concise, we present here benchmarks only for the three-dimensional EA model with periodic boundary conditions, using Gaussian J_{ij} of zero mean and unit variance. We denote by L the lattice "size"; the number of spins is then $N = L^3$.

First, consider the mean error of the output of the VDS; call E the estimated ground-state energy and E_0 the true ground-state energy for a given realization of the quenched disorder. (We compute E_0 for each sample using the most powerful algorithm we have in conjunction with multiple trials: for the sizes we consider here, the true ground state is most probably correctly determined for *all* our samples.) We want to know what is the size of the error made by VDS on each lattice, and so we consider

$$\delta e \equiv \frac{E_0 - E}{E_0} \tag{3.4}$$

where the choice of sign is due to the fact that the energies are always negative. δe fluctuates with the disorder sample and even for a given sample, as it depends on the input configuration (which we obtain by setting the spins randomly). To characterize its distribution, we show in Figure 3.5 both the mean and the standard deviation of δe as a function of the lattice size L. We see that the mean error grows with L but saturates reasonably rapidly just above 6%. The standard deviation is also displayed in that figure in the form of error bars. (Note that these "error bars" are thus *not* the uncertainty on the mean; in fact the statistical uncertainties on the mean of δe are smaller than the symbols.) As expected from the claim that E is self-averaging, the standard deviation decreases at large L.

Another quantity of importance is the probability that VDS finds the exact ground state. We thus measure the probability that $\delta e = 0$ and average it over disorder samples; for a given lattice size L, call this average $P_L(0)$. We find $P_3(0) = 0.513$, $P_4(0) = 0.124$, $P_5(0) =$

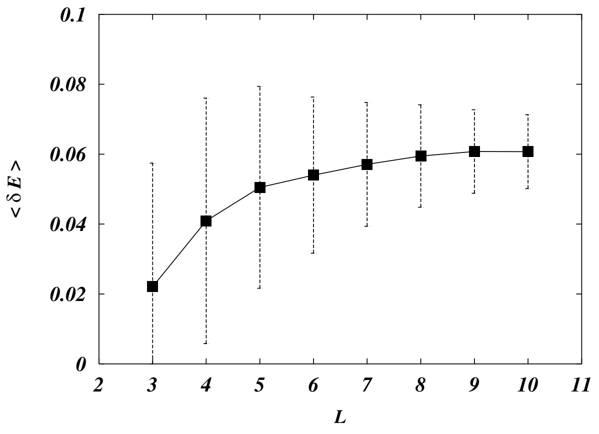


Figure 3.5: The mean relative error in the ground-state energy and the standard deviation of that error as a function of L in the $3-d$ EA model when using VDS.

$0.009, \ldots$ This decrease is very fast, and so we see that our VDS is useful for finding ground states (using multiple trials) at $L \leq 4$ but unreliable for $L \geq 5$.

Finally, what about the time VDS takes to terminate? As mentioned before, we have approximately an $N \ln N$ behavior. The constant in front of this factor depends on the speed of the machine, of course. On a 180 MHz PC, the time to do 10 000 VDS is approximately 80 seconds when $L = 6$ and 400 seconds when $L = 10$.

3.2.3 Genetic Renormalization Algorithm

Generalizations of local searches such as VDS are typically fast but they are also "near-sighted": they can get out of a local minimum only if the neighborhood search climbs out of the current valley. It is believed that in most hard problems such as spin glasses, escaping from a local minimum requires changing a large number of variables, in fact a number that diverges with N (the number of variables defining the combinatorial optimization problem). It seems plausible that smarter approaches could be devised if one could exploit *several* configurations at local minima rather than just one (a limitation inherent to LS and VDS). This brings us to population-based searches, Genetic Algorithms (GA) being the most widely used of these (see, for instance, the article by C. Reeves in [14]). Using a biologically motivated vocabulary, the goal is to use several good parents to generate even better children. Biologically, this works well, but for combinatorial optimization problems it is a non-trivial task especially when there are constraints: if one is not very careful, most "combinations" of parents will give children that are *less* fit than the parents! Interestingly, this difficulty was to a large extent overcome for spin glasses by Pál's introduction of the "triadic" crossover [44] where a child is produced from *three* parents. Since that early work, enhancements as well as other approaches have been developed [15,16,25,38,45,47], all using GA with more than two parents. In fact, all of today's most competitive (meta)heuristics for spin glasses fall into this class. The algorithm presented below [27] combines the multi-parent GA approach with "renormalization", thereby allowing optimization on multiple scales; we call it GRA for "Genetic Renormalization Algorithm". The other most powerful heuristic algorithm for spin glasses is called "Cluster-exact Approximation" and has been explained in detail in Chapter 9 of the book "Optimization Algorithms in Physics" [21].

3.2.3.1 Renormalization

Renormalization is a powerful concept, which began to flourish in statistical physics in the 1970s [55]; today it is commonly used in Monte Carlo simulations as well as in analytical work. However, there are major stumbling blocks when trying to apply this formalism to *disordered* systems. In fact, there is an ongoing debate about whether renormalization can make sense in strongly disordered systems like spin glasses. We take a weaker formulation, considering renormalization only as a way to access multiple energy scales; in our optimization context, this means that renormalization will be used to construct Hamiltonians of "block" spins, but without the need to make these blocks, nor the associated Hamiltonian, unique. Because of this restriction, we are merely providing a procedure to realize multi-scale optimization.

The first use of renormalization for optimization in spin glasses goes back to 1992 [29]. The lack of follow-up on the work of those authors is due to the disappointing performance of their over-all algorithm; some further ingredients are necessary to tackle lattice sizes of relevance for interesting physical questions. Nevertheless, for the renormalization part, those authors included the essential ideas. Suppose we are given a spin-glass Hamiltonian specified in terms of a weighted graph G with N sites (vertices) and edges of weights J_{ij}. We are also given k configurations. (These configurations are not specified by G; it is in that sense that the renormalization of G is not unique.) We denote these configurations by $\{S_i^{(1)}\}$, $\{S_i^{(2)}\}$, $\ldots\{S_i^{(k)}\}$, with $1 \leq i \leq N$. For each site i, define the list of $(k-1)$ site "overlaps"

$$\vec{q}_i = (S_i^{(1)}S_i^{(2)}, S_i^{(1)}S_i^{(3)}, \ldots, S_i^{(1)}S_i^{(k)}) \tag{3.5}$$

$\vec{q}_i$ can be thought of as a $(k-1)$-dimensional vector of ± 1 entries, and we refer to it as the "signature" of the k configurations. Given the signature at each site, we introduce an equivalence class over the sites: two sites i and j belong to the same class I if their signatures are equal; this means that their *relative* orientation (parallel or anti-parallel) is the same in all k configurations. We specify the orientations of these classes (sets of spins) by Ising block spins $S_I = \pm 1$; note that the k configurations differ only by the orientations of these block spins, no other flips need be considered. Finally, one can see that it is possible to work with the connected clusters of spins composing a given class, so that the block spins are now associated with the orientations of *clusters*. An example of this construction is given in Figure 3.6.

The energy of any configuration built out of these block spins is easily computed: it is given (up to an additive constant) by a renormalized spin-glass Hamiltonian H_R:

$$H_R = -\sum_{\langle IJ \rangle} J_{IJ} S_I S_J \quad \text{where} \quad J_{IJ} = \sum_{i\in I}\sum_{j\in J} J_{ij} S_i^{(1)} S_j^{(1)} \tag{3.6}$$

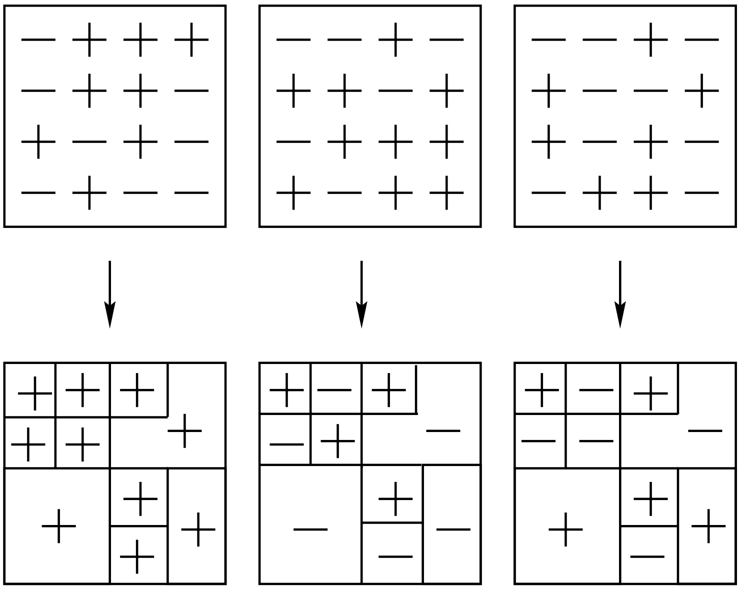


Figure 3.6: Renormalization and associated block spins of a 4×4 lattice using three arbitrary configurations.

are the renormalized couplings between the block spins. This renormalization is useful in practice because H_R involves fewer spins than H and thus may be tackled more effectively. Suppose, for example, that we are able to find the ground-state configuration $\mathcal{C}_R$ of H_R; then we can construct from $\mathcal{C}_R$ a configuration $\mathcal{C}$ of the non-renormalized spins S_i by recalling what the block spins mean in terms these spins. In many cases, the energy of $\mathcal{C}$ will be lower than that of any of the k parents used in the renormalization.

3.2.3.2 The Next_Generation Routine

Now we proceed with the "population" part of the algorithm, i.e., the GA part. We assume we have a population of, say, M +configurations that have been optimized by a local search procedure, hereafter referred to as Local_Opt. Just as in biological evolution, we want to take two or more parents and produce children in the hope that they will be better than their parents. We shall do this generation by generation. Roughly, we take k new parents from the population, have them produce children that are at least as good as themselves, and then put these children into the next generation. When all M configurations have been used in this process, we replace the old generation by the new one. (Note that to avoid wastage of CPU time and memory, we remove any repetitions of a child in the new generation.)

The heart of the evolution is the passage from parents to children. It is here that we appeal to renormalization: given say k parents, we renormalize the spin-glass problem as previously explained, obtaining H_R, a smaller graph, and "renormalized" parents. First of all, because of the reduction in the graph's size, Local_Opt now flips "block" spins at each elementary step; optimization then occurs on larger length scales than when working with the original graph. Secondly, we can bring in recursivity: indeed, H_R is again a spin-glass Hamiltonian and we want to find its ground state. In our first algorithmic approach [25], we solved this new problem from scratch, generating a population of random initial configurations. However, this is quite time consuming, and we have found that it is more effective to run Next_Generation recursively using only the configurations passed; this renders the algorithm much faster, and the final result is that larger spin-glass problems can be treated.

The schematic outline of our Next_Generation procedure is then as follows:

```
algorithm Next_Generation(G,configurations)
begin
  while configurations have not been used as parents do
    choose $k$ unused parents at random
    Renormalize using these Parents, generating a
          renormalized graph $G'$ and renormalized configurations
    Local_Opt these renormalized configurations on $G'$
    Next_Generation($G'$,configurations');    recursivity!
    Lift children (the configurations returned)
    Local_Optimize each of these configurations
    Add these to the next generation
  end do
  Remove redundant configurations of the new generation
  return
end
```

By Lift, we mean that a renormalized configuration associated with the block spins on G' is brought back to its representation using the spins on G, G' being a renormalization of G.

The number of parents used is very important. For the simplest choice, that is $k = 2$, most of the time the renormalization will lead to two huge block spins, each of which contains about half of the spins of the total system; then the children will be identical to the parents and nothing will be gained. Suppose now we increase k, leading to decreasing block spin sizes. If k is too large, the renormalized graph G' is identical to the original graph G and nothing is gained. Finally, if k is neither too small nor too large, G' is significantly smaller than G, yet finding its ground state can still be a challenge; this is the regime of interest. In view of this, Next_Generation adjusts k dynamically so that the size of G' is about 40% that of G. This modification works well, but for the *last* renormalization we generally cannot find enough unused parents to get a reasonable G'; we then decide to take previously used parents to perform that last renormalization.

It is easy to see that Next_Generation produces children that are at least as good as their parents. In order to prevent the algorithm from getting "stuck", returning children that are *identical* to their parents, we implement the additional constraint that the largest energy in the base population decreases strictly. In particular, if Next_Generation does not modify the population, then we satisfy this extra constraint by removing the worst configuration from the new generation. Ref. [27] details some further ways to make the algorithm more efficient.

3.2.3.3 Putting it all Together

All in all, the genetic renormalization algorithm (GRA) has two core routines, Local_Opt and Next_Generation that are essentially independent. For Local_Opt we have used the variable depth search described previously, but any other single configuration optimizer can do. Given these two routines, the way to use them is straightforward:

```
algorithm Genetic_Renormalization_Algorithm(G,M)
begin
    comment    $G$ is the graph
    Initialize $M$ configurations
    Local_Opt these $M$ configurations
    while $(M > 1)$ Next_Generation($G$,configurations);
    return the single remaining configuration
end
```

In the initialization of the M configurations, our code assigns the spin variables randomly, but one could use other approaches such as constructive heuristics. Note that, at this level of description, M is the algorithm's only parameter accessible to the user.

The quality of the solutions found by Genetic_Renormalization_Algorithm does not depend too much on the choice of Local_Opt so if Genetic_Renormalization_Algorithm is to be improved, it is probably best to concentrate on new strategies within the Next_Generation routine. However, most of the CPU time is spent in Local_Opt via calls from Next_Generation, so optimizing this routine for speed is important. Another point is that the number of generations needed for termination of the program depends on the power of Local_Opt: using VDS rather than k-Opt allows for earlier termination. Thus both speed and quality of Local_Opt are in fact important.

3.2.3.4 Benchmarks

First, what is the *qualitative* behavior of GRA? When the M configurations have been initialized and run through the local search, the mean excess energy is around 6% (cf. the VDS benchmarks). Then, as the first few generations are produced, the mean excess energy drops rapidly to a fraction of a percent. After that, the improvement slows down at the same time as the population shrinks. After 10 to 20 generations, M reaches 1 at which point the algorithm halts. (The total number of generations is not very sensitive to the sample nor to M.)

As for all heuristic algorithms, GRA finds the ground state easily for small N and fails at large N, the important issue being the cross-over point. Just as for the VDS benchmarks, we focus on $L \times L \times L$ lattices with periodic boundary conditions and Gaussian J_{ij}. At each value of L, we can determine the value of M for which the mean probability (averaged over the disorder) of finding the ground state $P_{L,M}(0)$ is a given constant. We have observed [27] the following scaling law

$$P_{L,M}(0) = 1 - \exp\left[-\frac{M}{M^*(L)}\right] \tag{3.7}$$

where $M^*(L)$ is a function we determine numerically. Note that with this notation one has $P_{L,M^*(L)}(0) = 1 - e^{-1}$. Our measurements give, for instance, $M^*(6) \approx 14$, $M^*(8) \approx 53$,

$M^*(10) \approx 200$, $M^*(12) \approx 650$, ... For still larger sizes, $M^*(L)$ becomes so big that we are not sure (even in a probabilistic sense) that we ever reach the true ground state.

Another issue concerns the CPU time. Our usage reveals that it grows roughly linearly with M and $N = L^3$. In fact, running GRA at $M^*(L)$ on a 180 MHz PC takes about 24 seconds for $L = 10$ and about 135 seconds for $L = 12$. We thus see that ground-state studies for $L \leq 12$ lattices are quite feasible; unfortunately, the growth of $M^*(L)$ is so steep that GRA becomes unreliable for larger lattice sizes.

3.3 A Survey of Physics Results

Past and current studies have not led to clear-cut answers to the main challenges in spin glasses. Nevertheless, the field has benefited tremendously from the use of ground-state solvers. In particular, at present the droplet/scaling picture seems less compatible with the numerical studies than before. We cannot cover here all past work on spin-glass ground states, so we shall focus exclusively on the $d = 3$ EA model. In the last few years, several teams [16, 25, 38, 46, 47], each with their own codes, have tried to understand better the properties of this system. Let us now give a broad overview of these efforts.

3.3.1 Convergence of the Ground-state Energy Density

In 1996, Pál [45] obtained accurate estimates of ground-state energies as a function of L. From these he guessed that the mean energy scaled as $e_0 L^3$ with corrections to scaling that were either $O(1)$ or grew as a small power of L. Computations by Hartmann [15, 16] also lead to similar results. Later it was realized [3] that the droplet/scaling picture in fact predicts

$$\overline{E_0} \approx e_0 L^3 + e_1 L^\theta \tag{3.8}$$

where $\theta \approx 0.20$ is the domain wall exponent (see the next paragraph). Such a law is certainly compatible with the best current data, but the uncertainty of that correction to the scaling exponent remains large. Interestingly, the mean-field prediction is also Eq. (3.8) but with $\theta = 1.0$; that is completely excluded by all the studies.

3.3.2 Domain Walls

In the droplet/scaling picture, spin-glass ordering is characterized by a stiffness of the ground state to changes in its boundary conditions. In practice, this stiffness is measured by the change ΔE in the ground-state energy when going from periodic to anti-periodic boundary conditions. If there is spin-glass ordering, the droplet/scaling picture predicts that the typical values of ΔE grow as L^θ for $L \times L \times L$ lattices. Furthermore, one expects the large-L scaling

$$P(L, \Delta E) \underset{L \to \infty}{\longrightarrow} \frac{P(\Delta E / L^\theta)}{L^\theta} \,. \tag{3.9}$$

Numerous studies [2, 4, 6, 18, 39, 47] have confirmed this and now give values for θ in $d = 3$ that are between 0.19 and 0.24.

Another property of interest is the nature of the domain walls themselves. Although clearly they must be very rough, in fact they span the whole system [37]. Furthermore, they may be space *filling*, but additional studies are necessary to control the associated finite-size effects.

3.3.3 Clustering of Ground States

In studies of the EA model, the J_{ij} are sometimes taken to be Gaussian, sometimes "binary", i.e., $J_{ij} = \pm 1$. For the second choice, the ground state is highly degenerate, having an extensive entropy. Given all the ground states for a given disorder sample, it is natural to ask how they are organized. A first way to tackle this is to consider the distribution of overlaps among ground states; the overlap q of two configurations $\mathcal{C}^{(k)} = \{S_1^{(k)}, S_2^{(k)}, \ldots, S_N^{(k)}\}$ ($k = 1, 2$), is given by

$$q(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}) = \frac{1}{N} \sum_{i=1}^{N} S_i^{(1)} S_i^{(2)} \tag{3.10}$$

and thus $q = 1 - 2Nd_H$ where d_H is the usual Hamming distance of $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$. Replica symmetry breaking (RSB) among the ground states means that the probability distribution $P(q)$ of these overlaps is non-trivial in the thermodynamic limit. A trivial $P(q)$, namely two delta functions at $\pm q_{\mathrm{EA}}$, is the prediction of the droplet/scaling picture. The first estimates of $P(q)$ came from Hartmann [16, 19], but unfortunately the large-L extrapolation was difficult. The most recent investigations [22, 49] give contradictory conclusions so this question deserves further study.

A second approach for understanding the organization of these ground states is to cluster them into families. In the mean-field picture, one expects the clusters to exhibit ultrametricity. Although initial computations [16] saw signs of ultrametricity, the most recent results [8, 23] do not go in that direction. One thus has the rather paradoxical situation where there are multiple clusters of ground states (RSB-like) yet no ultrametricity.

3.3.4 Low-energy Excitations

Given the ground state(s), we want to explore the system's energy landscape. The droplet/scaling picture has definite predictions for this which have begun to be tested, though mainly in $d = 2$. For $d = 3$, it has been found [34] that energies do not grow with the droplet's size unless these are forced to be compact; without such a constraint, the droplets resemble lattice animals as expected from the mean-field picture.

It is also possible to look at system-size excitations, i.e., excitations that span the whole system. In the droplet/scaling picture, these are expected to have energies that grow indefinitely as $L \to \infty$. On the contrary, in the mean-field picture, one expects them to have $O(1)$ energies and to be topologically random like sponges [26], while their overlap with the ground state should exhibit continuous RSB. All these mean-field features transpire from the different ways to search for system-size excitations [20, 32, 38, 46, 48], seemingly invalidating the droplet/scaling picture. However, other aspects are not so clear. One essential prediction of the mean-field picture is that these system-size excitations should be both space spanning and space *filling*. This means that the surface area of these excitations should grow linearly with the lattice volume; interestingly, measurements indicate that the typical surface grows instead as L^{d_s} with $d_s \approx 2.7$ [32] or $d_s \approx 2.6$ [48], both estimates of d_s being significantly below 3. If such a scaling were to hold for all L, the distribution of "link" overlaps would be trivial at large L. (The link overlap is obtained from Eq. (3.10) by replacing each spin variable S_i by the link variable S_iS_j, one for each nearest-neighbor pair on the lattice.) Given that the standard (spin) overlaps are non-trivial (exhibit RSB), such a pattern is referred to as TNT. Of course the current values of L are still relatively modest; the mean field would be confirmed if measurements at larger L gave an increased d_s. However, if that did not happen, other interpretations such as in [32, 43, 48] would have to be considered in greater depth.

3.3.5 Phase Diagram

One of the challenges mentioned at the beginning of this chapter was the identification of the phase diagram of the EA model in $d = 3$. When the fraction of ferromagnetic bonds goes beyond a threshold concentration, ferromagnetism appears [17] as can easily be detected by the spontaneous magnetization of the ground state. To test for the end of the spin-glass ordering, one has to look at the excitations. In [33], the signal for the spin-glass ordering was taken to be the presence of sponge-like system-size excitations. Since these seemed to disappear just at the onset of ferromagnetism, it was argued that no mixed ferromagnetic spin-glass phase exists, in contrast to what is expected in the mean field.

In the same vein, several optimization studies have been performed to test whether spin-glass ordering survives in the presence of a magnetic field. Different signatures of spin-glass ordering have been used: chaos in the field [24], spongy system-size excitations [31], or the absence of magnetization for droplet-like excitations [35]. Interestingly, all these studies suggest that even a small field destroys the spin-glass ordering, in contrast to the mean-field expectation.

3.4 Outlook

If spin glasses are so controversial, it is largely because finite-size effects in these systems are not well understood. Without such an understanding, it is very difficult to extrapolate to the thermodynamic limit, i.e., $N \to \infty$ and to have researchers in the field reach a concensus. In many standard models, the large-N scaling arises rather early, that is once $N \gg 1$. In spin glasses though, this does not happen, and corrections to the leading scaling behavior seem to be controlled by *small* exponents. This means that algorithmic improvements have to be important, allowing us to go to significantly further in N. Currently, we can perform studies involving about 2000 spins. It would probably be necessary to at least double that to begin to have good enough control over the finite-size corrections. For instance, if the recent energy landscape studies could be extended to $L = 16$, most probably the data would no longer be even roughly compatible with all of the theoretical frameworks.

In summary, a necessary condition for progress in the field is better algorithmics; clearly both heuristic and exact approaches should be pursued. But such progress will have a much larger impact if smarter "theoretical" approaches are also developed. By that we mean that it is important to find: (1) improved probes for looking at spin-glass physics; (2) properly motivated parameterizations of finite-size effects. In the first class, the use of sponges or other geometrical properties of excitations has had some successes [32] but much more needs to be done. In the second class, either more sophisticated field theory methods or solvable models must serve as guides. Our feeling is that neither theory nor algorithmics on their own will settle the main challenges in spin glasses, but if put together, many longstanding controversies can most likely be finally resolved.

References

[1] K. Binder and A. P. Young, *Spin-glasses: Experimental facts, theoretical concepts and open questions*, Rev. Mod. Phys., **58**, 801 (1986).

[2] S. Boettcher, *Stiffness Exponents for Lattice Spin Glasses in Dimensions* $d = 3,\ldots,6$, cond-mat/0310698.

[3] J.-P. Bouchaud, F. Krzakala, and O. C. Martin, *Energy exponents and corrections to scaling in Ising spin glasses*, Phys. Rev. B **68**, 224404 (2003).

[4] A. J. Bray and M. A. Moore, *Lower critical dimension of Ising spin-glasses: A numerical study*, J. Phys. C Lett. **17**, L463 (1984).

[5] A. J. Bray and M. A. Moore, *Scaling theory of the ordered phase of spin-glasses*, In J. L. van Hemmen and I. Morgenstern, editors, *Heidelberg Colloquium on Glassy Dynamics*, volume 275 of *Lecture Notes in Physics*, pages 121–153, (Springer, Berlin, 1986).

[6] M. Cieplak and J.R. Banavar, *Scaling of stiffness in Ising spin glasses*, J. Phys. A **23**, 4385 (1990).

[7] J. R. L. de Almeida and D. J. Thouless, *Stability of the Sherrington-Kirkpatrick solution of a spin-glass model*, J. Phys. A **11**, 983 (1978).

[8] E. Domany, G. Hed, M. Palassini, and A. P. Young, *State hierarchy induced by correlated spin domains in short range spin glasses*, Phys. Rev. B **64**, 224406 (2001).

[9] S. F. Edwards and P. W. Anderson, *Theory of spin-glasses*, J. Phys. F **5**, 965 (1975).

[10] C. M. Fiduccia and R. M. Mattheyses, *A linear-time heuristic for improving network partitions*, In *Proceedings of the 19th Design Automation Workshop (Las Vegas)*, pages 175–181 (1982).

[11] K. H. Fischer and J. A. Hertz, *Spin-Glasses*, volume 1 of *Cambridge Studies in Magnetism.* (Cambridge University Press, Cambridge, 1991).

[12] D. S. Fisher and D. A. Huse, *Ordered phase of short-range Ising spin-glasses*, Phys. Rev. Lett. **56**, 1601 (1986).

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (Freeman, New York, 1979).

[14] F. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*. (Kluwer, Boston, 2002).

[15] A. K. Hartmann, *Cluster-exact approximation of spin glass groundstates*, Physica A **224**, 480 (1996).

[16] A. K. Hartmann, *Evidence for existence of many pure ground states in 3d* $\pm J$ *spin glasses*, Europhys. Lett. **40**, 429 (1997).

[17] A. K. Hartmann, *Ground-state behavior of the 3d $\pm J$ random-bond Ising model*, Phys. Rev. B **59**, 3617 (1999).

[18] A. K. Hartmann, *Scaling of stiffness energy for 3d $\pm J$ Ising spin glasses*, Phys. Rev. E **59**, 84 (1999).

[19] A. K. Hartmann, *How to evaluate ground-state landscapes of spin glasses thermodynamical correctly*, Eur. Phys. J. B **13**, 539 (2000).

[20] A. K. Hartmann and F. Ricci-Tersenghi, *Direct sampling of complex landscapes at low temperatures: the three-dimensional $\pm J$ Ising spin glass*, Phys. Rev. B **66**, 224419 (2002).

[21] A. K. Hartmann and H. Rieger, editors, *Optimization Algorithms in Physics*. (Wiley-VCH, Berlin, 2002).

[22] G. Hed, A.K. Hartmann, and E. Domany, *Correct extrapolation of overlap distribution in spin glasses*, Europhys. Lett. **55**, 112 (2001).

[23] G. Hed, A.K. Hartmann, D. Stauffer, and E. Domany, *Spin domains generate hierarchical ground state structure in $J = \pm 1$ spin glasses*, Phys. Rev. Lett. **86**, 3148 (2001).

[24] J. Houdayer and O. C. Martin, *Ising spin glasses in a magnetic field*, Phys. Rev. Lett. **82**, 4934 (1999).

[25] J. Houdayer and O. C. Martin, *Renormalization for discrete optimization*, Phys. Rev. Lett. **83**, 1030 (1999).

[26] J. Houdayer and O. C. Martin, *A geometrical picture for finite dimensional spin glasses*, Europhys. Lett. **49**, 794 (2000).

[27] J. Houdayer and O. C. Martin, *A hierarchical approach for computing spin glass ground states*, Phys. Rev. E **64**, 056704 (2001).

[28] E. Ising, *Beitrag zur theorie des ferromagnetismus*, *Zeitschr. f. Physik* **31**,253 (1925).

[29] N. Kawashima and M. Suzuki, *Replica optimization method for ground-state search of random spin systems*, J. Phys. A **25**, 1055 (1992).

[30] B. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Technical Journal **49**, 291 (1970).

[31] F. Krzakala, J. Houdayer, E. Marinari, O. C. Martin, and G. Parisi, *Zero-temperature responses of a 3d spin glass in a field*, Phys. Rev. Lett. **87**, 197204 (2001).

[32] F. Krzakala and O. C. Martin, *Spin and link overlaps in three-dimensional spin glasses*, Phys. Rev. Lett. **85**, 3013 (2000).

[33] F. Krzakala and O. C. Martin, *Absence of an equilibrium ferromagnetic spin-glass phase in 3d*, Phys. Rev. Lett. **89**, 267202 (2002).

[34] J. Lamarcq, J.-P. Bouchaud, O. C. Martin, and M. Mézard, *Non-compact local excitations in spin-glasses*, Europhys. Lett. **58**, 321 (2002).

[35] J. Lamarcq, J.-P. Bouchaud, and O. C. Martin, *Local excitations of a spin glass in a magnetic field*, Phys. Rev. B **68**, 012404 (2003).

[36] S. Lin, *Computer solutions of the traveling salesman problem*, Bell System Technical Journal **44**, 2245 (1965).

[37] E. Marinari and G. Parisi, *Effects of changing the boundary conditions on the ground state of Ising spin glasses*, Phys. Rev. B **62**, 11677 (2000).

[38] E. Marinari and G. Parisi, *On the effects of a bulk perturbation on the ground state of 3d Ising spin glasses*, Phys. Rev. Lett. **86**, 3887 (2001).

[39] W. L. McMillan, *Domain-wall renormalization-group study of the three-dimensional random Ising model*, Phys. Rev. B **30**, 476 (1984).

[40] M. Mézard and G. Parisi, *The Bethe lattice spin glass revisited*, Eur. Phys. J. B **20**, 217, (2001).

[41] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin-Glass Theory and Beyond*, volume 9 of Lecture Notes in Physics. (World Scientific, Singapore, 1987).

[42] G. Migliorini and A. N. Berker, *Global random-field spin-glass phase diagrams in two and three dimensions*, Phys. Rev. B. **57**, 426 (1998).

[43] C. M. Newman and D. L. Stein, *The state(s) of replica symmetry breaking: Mean field theories vs. short-range spin glasses*, J. Stat. Phys. **106**, 213 (2002).

[44] K. F. Pál, *Genetic algorithm with local optimization*, Biol. Cybern. **73**, 335 (1995).

[45] K. F. Pál, *The ground state energy of the Edwards-Anderson Ising spin glass with a hybrid genetic algorithm*, Physica A **223**, 283 (1996).

[46] M. Palassini, F. Liers, M. Juenger, and A. P. Young, *Low-energy excitations in spin glasses from exact ground states*, Phys. Rev. B **68**, 064413 (2003).

[47] M. Palassini and A. P. Young, *Triviality of the ground state structure in Ising spin glasses*, Phys. Rev. Lett. **83**, 5126 (1999).

[48] M. Palassini and A. P. Young, *Nature of the spin-glass state*, Phys. Rev. Lett. **85**, 3017 (2000).

[49] M. Palassini and A. P. Young, *The $\pm J$ spin glass: Effects of ground state degeneracy*, Phys. Rev. B **63**, 140408(R) (2001).

[50] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. (Prentice Hall, Englewood Cliffs, NJ, 1982).

[51] G. Parisi, *Infinite number of order parameter for spin-glasses*, Phys. Rev. Lett. **43**, 1754 (1979).

[52] G. R. Schreiber and O. C. Martin, *Cut size statistics of graph bisection heuristics*, SIAM Journal on Optimization **10**, 231 (1999).

[53] D. Sherrington and S. Kirkpatrick, *Solvable model of a spin-glass*, Phys. Rev. Lett. **35**, 1792 (1975).

[54] G. Toulouse, *Theory of frustration effect in spin-glasses: I*, Comm. Phys. **2**, 115 (1977).

[55] K. Wilson and M. E. Fisher, *Critical exponents in 3.99 dimensions*, Phys. Rev. Lett. **28**, 240 (1972).

4 Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-cut

Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi

4.1 Introduction

When studying ground-state properties of Ising spin glasses, the computation of exact ground states, i.e., spin configurations with exact rather than approximate minimum energy, is often desirable. The Ising model is one of the most commonly used, both for its simplicity and its accuracy in describing many real world systems. When we refer to a "spin glass" in this chapter, we always have the Ising model in mind. For a short introduction into the physics of spin glasses, see Chapter 3.

The configurations that are mostly considered in the literature are the two- and three-dimensional Ising spin glasses on a grid with nearest-neighbor interactions and free or periodic boundary interactions, with or without an external magnetic field. The periodic boundary conditions are a standard way of approximating an infinite spin glass with a structure that contains only a finite number of spins.

Unfortunately, there are no functions in closed form that, given all interactions between the spins and the external field, yield a ground state. Therefore, the only way to compute a ground state is using a numerical algorithm. Since the total number of states for a structure with n spins is 2^n, as soon as n exceeds, say, 35, it is impossible, from a computational point of view, to find a ground state by *brute force*, i.e., by enumerating all possible states and computing the energy for each of them.

A fundamental question from both a practical and a theoretical point of view in computer science, is to determine whether it is possible to design an algorithm that, for any possible choice of the spin interactions and of the magnetic field, finds a ground state in a number of elementary operations bounded by a polynomial function of n, or, more precisely, of the number of bits needed to store the problem data. A problem for which this is possible is called *polynomially solvable* and the procedure used is called a *polynomial algorithm*.

The theory of computational complexity has given quite negative results for the possibility of treating certain Ising spin-glass problems. Most cases are NP-hard. They are therefore generally considered as notoriously hard, and it is believed that there can be no polynomial algorithm for their solution.

Due to the difficulty of the problem, popular methods described in the literature compute an approximation of the value of the minimum energy of the spin glass. Such methods usually use Monte Carlo simulation including simulated annealing and evolutionary and genetic algorithms. More recent algorithmic approaches are described in [19]. Despite the practical

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

usefulness and efficiency of the heuristic methods used in the literature, two main drawbacks arise:

- It is not possible to estimate how far away from a real ground state is the produced solution, when the algorithm stops. Therefore it is not possible to determine reliably the degree of accuracy of the experimental results produced with these methods. For example, if we want to compute the expected ground-state energy as a function of the grid size, using heuristic methods, the values will always have a positive bias, no matter how large the grid size is, how many experiments are done or how much computation time is invested.

- Two different states with almost the same energy may be very different. Therefore, it is not clear whether the state produced by these algorithms yields useful information on the structure of the ground state; one has to keep this in mind when *real* ground states have to be analyzed.

In the seventies and early eighties, the NP-hardness of the ground state problem was believed to make its exact solution practically impossible, but the introduction of branch-and-cut techniques for NP-hard combinatorial optimization problems has made such computations practical. Nowadays, we can treat system sizes that allow for meaningful statistics in theoretical physics.

In this chapter we explain the branch-and-cut technique for exact ground-state computation and show its potential for the study of ground-state properties of Ising spin glasses.

In Section 4.2 we show that the ground-state problem for Ising spin glasses is equivalent to the max-cut problem that is well known in combinatorial optimization and has a number of further applications. From then on, we concentrate on properties of and algorithmic approaches to max-cut. We start by introducing a general scheme for solving hard max-cut problems in Section 4.3 which consists of specializing the well known branch-and-bound method to max-cut. In branch-and-bound, relaxations play a crucial role, and as a prerequisite for the branch-and-cut algorithm we are aiming at, we study linear programming relaxations of max-cut in some detail in Section 4.4. After these preparations, we outline in Section 4.5, the main components of our branch-and-cut method. In Section 4.6 we present some recent results obtained with exact ground-state computations in the context of studying the nature of the spin-glass phase. Then we comment on advantages of branch-and-cut for exact ground-state computations in Section 4.7. We close in Section 4.8 with some challenges which we see for the future.

4.2 Ground States and Maximum Cuts

Finding the ground state of a spin glass is closely related to a well known problem in combinatorial optimization: the maximum cut problem in a weighted graph (max-cut problem for short).

The max-cut problem is the following. We are given a graph $G = (V, E)$ with weights $c_{ij} \in \mathbb{R}$ for all edges $ij \in E$. For each (possibly empty) subset W of the node set V, the *cut* $\delta(W)$ in G is the set of all its edges with one endpoint in W and the other in $V \setminus W := \{i \in V \mid i \notin W\}$. W and $V \setminus W$ are called the *shores* of the cut $\delta(W) = \delta(V \setminus W)$. The *weight* *of a cut* is given by the sum of the weights of all its edges. The *max-cut problem* is to find a cut of G with maximum weight.

Assume that we have a spin glass consisting of n spins $S_1, S_2, \ldots, S_n$ where the variables S_i take values $+1$ or -1 and an external magnetic field of strength h.

Given a spin configuration ω, the Hamiltonian of this system is

$$H(\omega) = -\sum_{(ij)} J_{ij} S_i S_j - h \sum_{i=1}^{n} S_i, \tag{4.1}$$

where the sum $\sum_{(ij)}$ is over the coupled spins.

We identify the spins with the node set $V = \{1, \ldots, n\}$ of a graph $G = (V, E)$, the interaction graph associated with the system. For a pair i, j of nodes, G contains an edge ij if the interaction J_{ij} between two magnetic impurities is non zero. We introduce an extra node "0" with "ghost spin" S_0 for the external magnetic field that is connected to all other nodes i by an edge $0i$ of weight h. We rewrite

$$H(\omega) = -\sum_{ij \in E} J_{ij} S_i S_j - \sum_{i=1}^{n} h S_0 S_i. \tag{4.2}$$

Observe that each spin configuration ω induces a partition of the node set V of the interaction graph G into node sets V^+ and V^-, where $V^+ := \{i \in V \mid S_i = +1\}$ and $V^- = \{i \in V \mid S_i = -1\}$. So the energy of the spin configuration ω can be translated to the form

$$H(\omega) = -2 \sum_{ij \in \delta(V^+)} c_{ij} - C, \tag{4.3}$$

where $c_{ij} := -J_{ij}$ for all $ij \in E$ and $C := \sum_{ij \in E} J_{ij}$. Hence, the problem of minimizing H is equivalent to maximizing

$$c(\delta(V^+)) := \sum_{ij \in \delta(V^+)} c_{ij} \tag{4.4}$$

over all $V^+ \subseteq V$.

This problem is a max-cut problem in the interaction graph G associated with the spin-glass system. Thus, finding a ground state in the Ising model of a spin glass is equivalent to finding an optimum solution of the corresponding max-cut problem. The structure of the underlying graph and the type of the objective function determine the complexity of the problem.

In Figure 4.1 we show an instance on a 3×3 grid with periodic boundary conditions, ± 1 interactions and no external field. Figure 4.1(a) shows the max-cut problem. The solid lines have edge weight 1 (i.e., the coupling strength in the spin-glass instance is -1), the dashed lines weight -1. Figure 4.1(b) shows an optimum solution. The dash-dotted lines correspond to the edges of the cut.

The max-cut problem is NP-hard in general, but is polynomially solvable for some classes of graphs like, e.g., planar graphs, graphs not contractable to K_5 (the complete graph on five

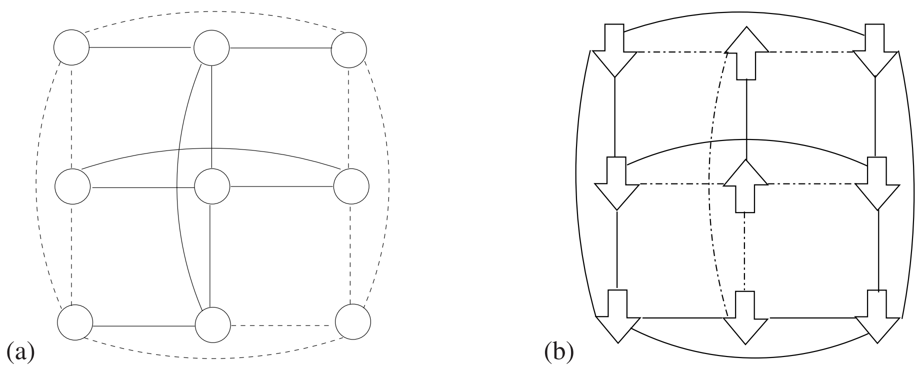


Figure 4.1: Example for a 3×3 instance. (a) A 3×3 instance. The solid lines have weight 1, the dashed lines weight -1. (b) An optimal solution. The dash-dotted lines correspond to the cut edges.

nodes), weakly bipartite graphs and graphs with non-negative edge weights. For example, the standard two-dimensional grid model with nearest-neighbor interactions, no periodic boundary conditions and no magnetic field, amounts to solving a max-cut problem in a planar graph, and is therefore polynomially solvable. Surprisingly, max-cut is already NP-hard for almost planar graphs [3], i.e., graphs where only one node has to be removed to obtain a planar graph. There are quite negative results for the possibility of solving the Ising spin glass with periodic boundary conditions. Polynomially solvable cases are the two-dimensional grid without a field and $\pm J$ interactions [30], and more generally, the case in which the genus of the graph is bounded by a constant and the sizes of the integral edge weights are bounded in absolute value by a polynomial in the size of the graph [16]. For the Gaussian case the question is still open. As soon as we have an external field, the problem becomes NP-hard for all kinds of spin interactions [2].

Goemans and Williamson [20] presented a 0.878-approximation algorithm for the maximum cut problem, i.e., an algorithm with running time bounded by a polynomial in the input size that provably delivers a solution of at least 0.878 times the optimum value of a maximum cut. However, the bad news is that under the assumption P$\neq$NP there is no polynomial algorithm that provably delivers a solution of at least 98% of the optimum value of a maximum cut [10]. In the following we only deal with optimum solutions of the maximum cut problem.

The algorithmic approach in this chapter is applicable to any instance of the max-cut problem and therefore to other combinatorial optimization problems that, like the ground-state problem, can quite easily be transformed to max-cut. Here are some examples:

- A generic optimization problem whose equivalence to max-cut has been shown by Hammer, see [11] is *quadratic 0-1 optimization*, in which we are given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $a \in \mathbb{R}^n$ and wish to solve $\min\{x^T Ax + a^T x \mid x \in \{0,1\}^n\}$, see [7] for details.

- In [6] an application to the layout of electronic circuits is described: when the wiring occurs on two layers, the task is to assign each wire segment to one of these layers so as to minimize the number of *vias*, i.e., layer changes along the wires.

- An application to the vertical or horizontal polarization assignment to television broadcast antennas is described in [1]: The optimization goal is to minimize interference between the transmitters.
- In a tournament schedule for a sports league, a *break* consists of two consecutive "home" or two consecutive "away" games for an individual team. Breaks are considered undesirable. A reduction of the break minimization problem to max-cut is described in [14] along with a computational study.

4.3 A General Scheme for Solving Hard Max-cut Problems

The most common algorithmic framework for solving NP-hard combinatorial optimization problems to optimality is the branch-and-bound scheme that has been introduced by [25] in the context of the general integer linear programming problem

$$\max\{c^T x \mid Ax \leq b,\, x \geq 0,\, x \text{ integer}\} \tag{4.5}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. We describe this framework here in the context of max-cut while keeping in mind that it is applicable in a much broader sense. Understanding branch-and-bound and some properties of the set of cuts in graphs is a prerequisite for the branch-and-cut algorithm at which we are aiming.

Recall that max-cut is defined on an undirected graph $G = (V, E)$ with edge weights c_e for all $e \in E$. For a subset $S \subseteq E$ and a vector $a \in \mathbb{R}^E$, let $a(S)$ be shorthand for $\sum_{e\in S} a_e$, where a_e is the eth component of the vector a. Denote by $\mathcal{F}$ the set of all cuts in G. Then the max-cut problem becomes

$$\max\{c(F) \mid F \in \mathcal{F}\}. \tag{4.6}$$

For the subsequent discussion it is convenient to replace the cuts $F \in \mathcal{F}$, i.e. the *feasible solutions* by their characteristic vectors in $\{0,1\}^E$: With any subset $S \subseteq E$ we associate its characteristic vector $\chi^S \in \{0,1\}^E$ defined by

$$\chi_e^S = \begin{cases} 0 & \text{if } e \notin S, \\ 1 & \text{if } e \in S. \end{cases} \tag{4.7}$$

If we let $\mathcal{C} = \{x \in \{0,1\}^E \mid x = \chi^F \text{ for some } F \in \mathcal{F}\}$, then (4.6) can equivalently be written as

$$\max\{c^T x \mid x \in \mathcal{C}\}. \tag{4.8}$$

When $\mathcal{C}$ is replaced by a superset $\mathcal{C}_R \supseteq \mathcal{C}$, the solution value of

$$\max\{c^T x \mid x \in \mathcal{C}_R\} \tag{4.9}$$

is an upper bound for the solution value of (4.8), i.e., an upper bound for the weight of a maximum cut in G. Problem (4.9) is a *relaxation* of problem (4.4), and a crucial ingredient of branch-and-bound algorithms is to find relaxations that are computationally "easy" to solve yet give "good" bounds.

Before we look at suitable relaxations of max-cut, we outline the generic branch-and-bound scheme that is a simple variant of the generic divide-and-conquer principle. Namely, if the problem cannot be solved directly, it is split into (hopefully easier) subproblems. These subproblems are then either solved or split further until eventually very simple problems are generated. From the solution of all the generated subproblems a solution of the original problem can then be constructed.

```
algorithm branch-and-bound
begin
  1. Initialize the list of active subproblems with the original problem.
  2. while list of active subproblems is not empty do
     begin
  3. choose some subproblem from the list of active problems and "solve" it as follows:
       (a) find an optimal solution for the subproblem, or
       (b) prove that the subproblem has no feasible solution, or
       (c) prove using a relaxation that there is no feasible solution for the subproblem that
           has a higher objective function value than the best feasible solution that is already
           known, or
       (d) split the subproblem into further subproblems and add them to the list of active
           problems, if none of the above is possible.
     end
  4. return the best feasible solution found so far as an optimal solution.
end
```

The splitting of problems into subproblems can be represented by the so-called branch-and-bound tree, the root of which represents the original problem. It is crucial for the efficiency of a branch-and-bound algorithm that this tree does not grow too large. Therefore subproblems have to be solved if possible by alternatives (a), (b) or (c) of step 3. Alternative (a) rarely occurs, for (b) and (c) relaxations are important. Namely for (b), if a relaxation of the subproblem is already infeasible, then also the subproblem itself is infeasible. To be able to finish the subproblem in (c), good lower and upper bounds must be available. Lower bounds are obtained by finding feasible solutions. These are either obtained by solving some subproblem to optimality or usually by determining good feasible solutions using heuristics. Upper bounds can be computed by using relaxations where, in principle, any type of relaxation discussed above can be employed. It is clear that the tighter a relaxation, the better the performance of the algorithm will be. Without a suitable relaxation, branch-and-bound would tend to completely enumerate the set of feasible solutions and thus become computationally infeasible. Of prime interest are linear programming relaxations that we will explain in the next section.

The easiest way to implement the splitting step 3(d) is to select a variable x_e and replace the current subproblem by two, in one of which the additional restriction $x_e = 0$ and in the other the additional restriction $x_e = 1$ is appended.

Let the symmetric difference of two cuts $F \in \mathcal{F}$ and $F' \in \mathcal{F}$ in G, denoted by $F \triangle F'$, be the set of edges that belongs to one of the two cuts but not to both. In the special context of max-cut, we can exploit the fact that for cuts $F \in \mathcal{F}$ and $F' \in \mathcal{F}$ $F \triangle F'$ also is a cut. We say the set of all cuts in a graph G *is closed under taking symmetric differences*. This simple combinatorial concept can be translated into algebraic terms and applied to the characteristic vectors of cuts. Let χ^F and $\chi^{F'}$ be the characteristic vectors of F and F', respectively. Then the map

$$s^F : \mathbb{R}^E \longrightarrow \mathbb{R}^E \tag{4.10}$$

called *switching along the cut* F and defined by

$$s^F(\chi^{F'})|_e = \begin{cases} \chi_e^{F'} & \text{if } e \notin F, \\ 1 - \chi_e^{F'} & \text{if } e \in F. \end{cases} \tag{4.11}$$

maps the characteristic vector of any cut F' in G into the characteristic vector of another cut in G. It maps χ^F to the 0 vector (the characteristic vector of the empty cut) and it is an automorphism of $\mathcal{F}$.

Given an instance of the max-cut problem defined by a graph G and an objective function vector c, for a fixed cut F, the map $s^F(\cdot)$ provides a variable transformation $z = s^F(x)$ that translates the problem

$$\max\{c^T x \mid x \in \mathcal{C}\} \tag{4.12}$$

into the problem

$$\max\{c'^T z + d \mid z \in \mathcal{C}\} \tag{4.13}$$

where c' is defined by

$$c'_e = \begin{cases} c_e & \text{if } e \notin F \\ -c_e & \text{if } e \in F \end{cases} \tag{4.14}$$

and $d = c\chi^F$.

In addition to the switching operation, we need the concept of contraction of two nodes u and v in G. This operation consists of identifying u and v with a new node, deleting the edge $e = uv$, if $e \in E$, and making all edges incident with either u or v in G, incident with the new node. Finally, double edges arising in this process are replaced by a single edge whose weight is the sum of the weights of the two edges, see Figure 4.2 for an illustration.

In max-cut we can contract a pair of nodes u and v whenever we want to make sure that u and v belong to the same shore of the cut, thus reducing the number of edges and therefore the dimension of the space in which the optimization takes place.

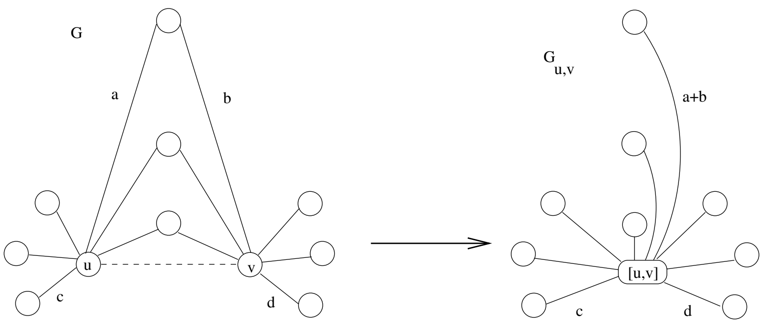


Figure 4.2: Contracting nodes u and v

Based on the switching and contraction operations, we can implement a branch-and-bound scheme for max-cut as follows:

Let us assume that we have a relaxation

$$\max\{c^T x \mid x \in X \subseteq [0,1]^E\} \tag{4.15}$$

of max-cut defined by the set X that contains all characteristic vectors of cuts in G and has the additional property that every integral vector of X is the characteristic vector of some cut in G, and that (4.15) is "easy" to solve. (We shall see later that such relaxations are readily available.)

Now let $\bar{x}$ be the characteristic vector of a cut in G. $\bar{x}$ serves as the incumbent solution: it corresponds to the best cut we know at any time during the execution of the branch-and-bound algorithm. Let x^* be an optimum solution of (4.15). If $c^T x^* < c^T \bar{x}$, we have case 3(c) and the subproblem is *fathomed*, i.e., needs no further consideration. If x^* is integral, it must be the characteristic vector of a cut, and we are again in case 3(c) after we have updated $\bar{x} \longleftarrow x^*$ in case $c^T x^* > c^T \bar{x}$, so we can consider the subproblem fathomed. Finally, we implement case 3(d) by choosing an edge $e = uv$ such that $0 < x^*_e < 1$, and replace the current subproblem by the following pair of subproblems:

(S1) The first subproblem is obtained by contracting u and v. This corresponds to forcing x_e to 0.

(S2) The second subproblem is obtained by switching x^* by $\delta(u)$, i.e., performing the variable transformation $s^{\delta(u)}(x^*)$ and recording this by multiplying all edge weights c_e for $e \in \delta(u)$ by -1, recording the additive constant $c\chi^{\delta(u)}$ for the objective function value, and finally contracting u and v. Contracting u and v amounts to forcing x_e to be equal to 0 in the transformed subproblem (after switching), which by switching corresponds to forcing x_e to 1 in the original subproblem (before switching).

Subproblems (S1) and (S2) are two new instances of max-cut. Therefore, any algorithm used to solve the original problem can be recursively applied to each subproblem.

There are various relaxations of max-cut that can be used in either implementation of the above branch-and-bound scheme, for an overview, see [22]. Here we want to concentrate on linear programming relaxations because their application has proven to be by far superior in the context of ground-state computations for spin glasses with short-range interactions such as those in grids or similar structures.

4.4 Linear Programming Relaxations of Max-cut

Before we introduce linear programming (LP) relaxations of max-cut, we derive an integer linear programming formulation of the form (4.5) for max-cut.

An edge set $C = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_0\} \subseteq E$ is called a *cycle* (of length k) in G. An edge $v_iv_j \in E$ where $j \neq (i+1) \bmod k$ is called a *chord* of the cycle C. The integer linear programming formulation is based on the observation that the intersection of a cut and a cycle in a graph G always contains an even number of edges and that this property characterizes those subsets of the edge set E that correspond to cuts. Using characteristic vectors of edge sets, this condition can be translated into algebraic terms by stipulating the *cycle inequality* $\chi(Q) - \chi(C \setminus Q) \leq |Q| - 1$ for all cycles C in G and each odd cardinality subset Q of C. (Recall that $\chi(S) = \sum_{e \in S} \chi_e$ for $S \subseteq E$.) For example, if C is a cycle consisting of an odd number of edges, the inequality for $Q = C$ says that any cut can contain at most $|C| - 1$ of the edges in C. If C is an arbitrary cycle and Q any odd subset of edges with $Q \neq C$, the inequality says that if a cut contains all edges of Q, i.e., $\chi(Q) = |Q|$, it must also contain at least one edge in $C \setminus Q$, i.e. $\chi(C \setminus Q) \geq 1$.

This leads to the following formulation of max-cut as an integer linear program:

$$\begin{aligned} \max\{c^T x \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 & \text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ & \text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 & \text{ for each } e \in E,\\ x_e & \text{ integer for each } e \in E\} \end{aligned} \tag{4.16}$$

Our first linear programming relaxation for max-cut is obtained by dropping the integrality conditions in (4.16):

$$\begin{aligned} \max\{c^T x \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 & \text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ & \text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 & \text{ for each } e \in E\} \end{aligned} \tag{4.17}$$

Obviously (4.17) possesses all the features that we required in the discussion of the previous section, except that it cannot immediately be “easily” solved. After all, (4.17) has an exponential number of inequalities, thus simply feeding it into computer memory is out of the question. Nevertheless, in practice (4.17) can indeed be solved efficiently in polynomial time, as we shall see later in this section. We call (4.17) the *cycle relaxation* of max-cut. As the feasible solutions of the cycle relaxation are the solutions of a system of linear inequalities, and they are all contained in the $|E|$-dimensional hypercube, they define a polytope which we call the cycle polytope

$$\begin{aligned} P^G_{\text{CYCLE}} = \{x \in \mathbb{R}^E \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 & \text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ & \text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 & \text{ for each } e \in E\}. \end{aligned} \tag{4.18}$$

If we take the convex hull of all characteristic vectors of cuts, or equivalently, all integer points in the cycle polytope, we obtain the *cut polytope*

$$\begin{aligned} P_{\mathrm{CUT}}^{G} &= \operatorname{conv}\{\chi^F \mid F \in \mathcal{F}\} \\ &= \operatorname{conv}\{x \in P_{\mathrm{CYCLE}}^{G} \mid x \in \{0,1\}^E\}. \end{aligned} \tag{4.19}$$

The vertices of the cut polytope are the characteristic vectors of the cuts in G, therefore max-cut can be formulated as

$$\max\{c^T x \mid x \in P_{\mathrm{CUT}}^{G}\}. \tag{4.20}$$

From the classical theorems of Weyl and Minkowski, we know that there is a matrix A and a vector b such that

$$P_{\mathrm{CUT}}^{G} = \{x \in \mathbb{R}^E \mid Ax \leq b,\ x \geq 0\}. \tag{4.21}$$

However, complexity theory tells us that such an inequality system is enormously large and that its explicit computation is hopeless in practice.

Let us consider the triangle graph as shown in Figure 4.3 as the smallest interesting example.

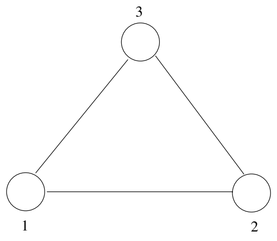


Figure 4.3: A graph consisting of a triangle.

The set of characteristic vectors of cuts is

$$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}, \tag{4.22}$$

where we order the coordinates as (x_{12}, x_{23}, x_{13}).

The cycle inequalities read

$$\begin{aligned} x_{12} + x_{23} + x_{13} &\leq 2 \\ x_{12} - x_{23} - x_{13} &\leq 0 \\ -x_{12} + x_{23} - x_{13} &\leq 0 \\ -x_{12} - x_{23} + x_{13} &\leq 0 \end{aligned} \tag{4.23}$$

The reader may easily verify that the vectors in (4.22) satisfy all four inequalities of (4.23) while other possible integer vectors violate at least one of them. For example, the vector $(1, 1, 1)$ (not corresponding to a cut) violates the first inequality in (4.23). In Figure 4.4 we show the cut polytope $P_{\mathrm{CUT}}^{K_3}$, i.e., the convex hull of the cut vectors (4.22) for the triangle graph of Figure 4.3. The cycle inequalities (4.23) are hyperplanes in $\mathbb{R}^3$, and it is not hard to see that each of them corresponds to a face of the tetrahedron in Figure 4.4. Hence, $P_{\mathrm{CUT}}^{K_3} = P_{\mathrm{CYCLE}}^{K_3}$.

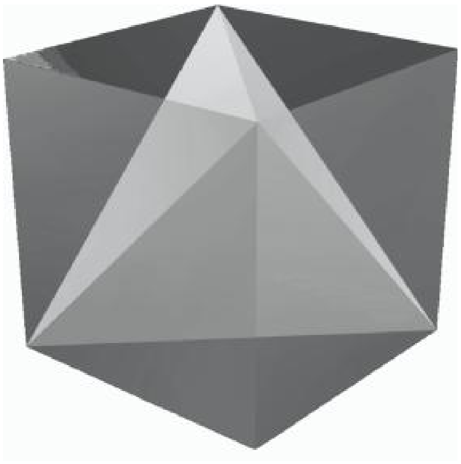


Figure 4.4: The cut polytope $P_{\mathrm{CUT}}^{K_3}$ in the cube $[0,1]^3$

We shall see that even partial descriptions of P_{CUT}^{G} by linear inequalities lead to relaxations of max-cut such as (4.17) that are useful in practical computations.

If $P \subseteq \mathbb{R}^n$ is a polytope, we say that the inequality $a^T x \leq a_0$ is *valid* for P if $P \subseteq \{x \in \mathbb{R}^n \mid a^T x \leq a_0\}$. For any valid inequality $a^T x \leq a_0$ for P, the polytope $\{x \in P \mid a^T x = a_0\}$ is called a *face* of P. Faces of P different from P itself are called *proper faces* of P. The proper faces of minimum dimension 0 are the vertices of P, and the proper faces of maximum dimension $\dim P - 1$ are called *facets* of P.

It can be shown that $\dim P_{\mathrm{CUT}}^{G} = |E|$, therefore the facets of P_{CUT}^{G} are the $(|E|-1)$-dimensional proper faces of P_{CUT}^{G}. The most compact description of a full-dimensional polytope consists of facet-defining inequalities only, one for each facet. It turns out that the cycle inequalities define facets of P_{CUT}^{G} whenever the defining cycle C has no chord. A trivial inequality $x_e \geq 0$ or $x_e \leq 1$ defines a facet of P_{CUT}^{G} when e is not contained in a triangle of G. All non-facet-defining inequalities can be eliminated from the linear description of P_{CYCLE}^{G} and P_{CUT}^{G}. Therefore, when G is the complete graph K_p, of all cycle and trivial inequalities, the only remaining inequalities are

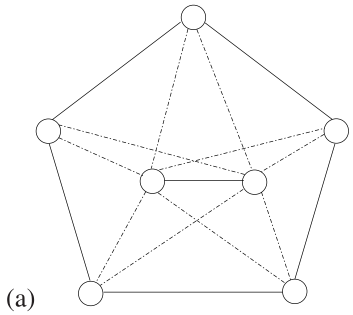
$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\leq 2 \\ x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0\,. \end{aligned} \tag{4.24}$$

In the following, we first introduce several classes of facets for the cut polytope. Later we concentrate on the cycle inequalities as they turn out to be especially important for practical spin glass ground-state computations.

A graph is called a *bicycle p-wheel* if it consists of a cycle of length p and two nodes adjacent to each other and to every node of the cycle. Let (W, B) be a bicycle $(2k+1)$-wheel, $k \geq 1$, contained in G. Then the inequality

$$x(B) \leq 2(2k+1) \tag{4.25}$$

defines a facet of P_{CUT}^{G}, [9]. See Figure 4.5(a) for a bicycle 5-wheel together with a cut of maximum cardinality satisfying the inequality with equality.



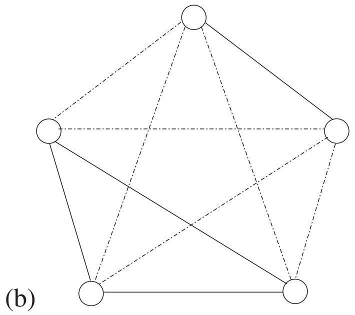


Figure 4.5: Valid inequalities for the cut polytope. (a) Bicycle $2(k+1)$-wheel with $k = 2$. A cut of maximum cardinality cuts all "spoke" edges. (b) K_5. The dash-dotted edges correspond to a cut of maximum cardinality.

Let $K_p = (W, E_p)$ be a complete subgraph of order p of G. Then the *K_p-inequality*

$$x(E_p) \leq \left\lceil \frac{p}{2} \right\rceil \left\lfloor \frac{p}{2} \right\rfloor \tag{4.26}$$

is valid for P_{CUT}^{G}, see Figure 4.5(b); this inequality defines a facet of P_{CUT}^{G} if and only if p is odd [9].

The polytope P_{CUT}^{G} has been studied by Barahona and Mahjoub [9] and there are further classes of facets known. In particular there exist interesting methods to construct new facet-defining inequalities from given facet-defining inequalities.

The usage of any of the above mentioned and further facet-defining inequalities tightens the relaxation. In practice, already the relaxation obtained by optimizing over the cycle polytope has proven useful, so let us first concentrate on the cycle polytope. At the beginning of this section, we claimed that we can efficiently solve the cycle relaxation even though it is defined by an exponential number of inequalities.

The solution is to start with a small subset of the constraint set, solve the LP to optimality, and check if the optimum solution satisfies the constraints that have not been taken into account. If this is the case, then the computed solution is optimal for the complete problem. Otherwise, there are constraints that are violated. One or some of them are added to the current LP, the LP is reoptimized, and the process continues until all constraints are satisfied. So the core problem that has to be solved here is the *separation problem*: Given some x^* and a set of constraints $Ax \leq b$, verify that $Ax^* \leq b$ or deliver an inequality $a^T x \leq a_0$ of the system such that $a^T x^* > a_0$. We describe below how this can be done for special relaxations that are of interest here.

A special case of a celebrated result of Grötschel, Lovász, and Schrijver [18] says that we can optimize a linear objective function over a polytope P in polynomial time if and only if we can solve the separation problem for P in polynomial time.

This approach is termed the *cutting-plane approach* due to the fact that the constraints added to the current LP "cut off" the current solution because it is infeasible for the original problem. It is important that the approach does not require an explicit list of the constraints defining the original problem. What is required is "only" a method for identifying inequalities that are violated by the current solution.

We show that the cycle inequalities can be separated in polynomial time implying that the cycle relaxation of the max-cut problem can be solved in polynomial time. The algorithm we are going to describe is by Barahona and Mahjoub [9].

Let y, $0 \leq y_e \leq 1$ for all $e \in E$, be the point to be separated. We define a new graph $H = (V' \cup V'', E' \cup E'' \cup E''') = (W, F)$ which consists of two copies of G, say $G' = (V', E')$ and $G'' = (V'', E'')$, and the following additional edges E''': For each edge $uv \in E$ we create the two edges $u'v''$ and $u''v'$, where $u' \in V'$, $u'' \in V''$ denote the two copies of a node $u \in V$. The edges $u'v' \in E'$ and $u''v'' \in E''$ are assigned the weight y_{uv}, while the edges $u'v''$, $u''v' \in E'''$ are assigned the weight $1 - y_{uv}$. In Figure 4.6 we show a graph consisting of a triangle and the corresponding doubled graph H.

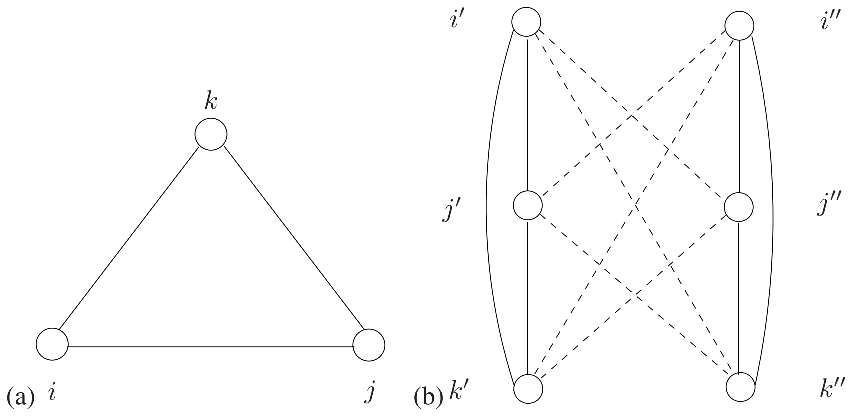


Figure 4.6: Separation of the cycle inequalities. (a) A graph consisting of a triangle i, j, k. (b) The corresponding graph H. The bold edges u', v', u'', v'' have weights x_{uv}, the dashed edges $u'v''$ and $u''v'$ have weights $1.0 - x_{uv}$.

For each pair of nodes $u', u'' \in W$ we calculate a shortest (with respect to the weights just defined) path in H. Such a path contains an odd number of edges of E''' and corresponds to a closed walk in G containing u. Clearly, if the shortest of these (u', u'')-paths in H has a length of less than 1, then there exists a cycle $C \subseteq E$ and an edge set $Q \subseteq C$, $|Q|$ odd, such that y violates the corresponding odd-cycle inequality. (C and Q are easily constructed from a shortest path.) If the shortest of these (u', u'')-paths has a length of at least 1, then y satisfies all these inequalities.

Since shortest-path computations can be efficiently performed in polynomial time, we have described a polynomial time separation algorithm for cycle inequalities, and therefore, the cycle relaxation can be solved in polynomial time.

Gerards [17] has shown that bicycle wheels can also be separated in polynomial time, but no polynomial time separation algorithm is known for the separation of K_p-inequalities.

A cutting-plane procedure requires the solution of lots of linear programs, most of which arise by appending further inequalities to a linear program solved previously.

A linear program can be considered an easy problem. From a theoretical point of view there are polynomial time algorithms (ellipsoid method, interior point methods) for its solution. But also from a practical point of view there are effective algorithms (simplex algorithms, interior point algorithms) which are able to solve very large problems with several ten-thousands of constraints and millions of variables.

For in-depth treatments of linear programming techniques, see the textbooks by Chvátal [12] and Padberg [26]. In the context of cutting-plane procedures, the preferred linear programming solver is the so-called *dual simplex method*. It is not a polynomial time method but its advantages are its practical efficiency and, in particular, the elegant and efficient re-optimization of a linear program after the addition of further inequalities.

The result of Grötschel, Lovász, and Schrijver is based on the ellipsoid method that solves linear programs in polynomial time yet does not perform well in practice for numerical reasons. By using the simplex method, we trade theoretical for practical efficiency.

4.5 Branch-and-cut

We can now describe our approach for solving hard spin-glass problems. The basic idea of the algorithm is to implement the bounding part in branch-and-bound by a cutting-plane algorithm for linear programming relaxations of max-cut as described in the previous section. Efficient separation is crucial for success. Therefore, we first describe how we separate cycle inequalities.

In the basic algorithm for cycle inequality separation, described in the previous section, we can use Dijkstra's well-known algorithm for the shortest–path computations and this results in an implementation with $O(|V|^3)$ running time.

For practical purposes this is rather slow. Therefore, we have added faster heuristics for finding violated cycle inequalities in order to avoid calling the "exact separation routine" as much as possible. We describe them in the order in which we call them in the algorithm.

Suppose $y = (y_1 \dots y_{|E|})$ is the optimum solution of the last linear program. We have to check whether y is the characteristic vector of a cut, and if not, find odd-cycle inequalities violated by y, if there are any.

For $0 \leq \varepsilon \leq \frac{1}{2}$ we define the graph $G_\varepsilon = (V, E_\varepsilon)$ as follows:

$$E_\varepsilon := \{e \in E \mid y_e \leq \varepsilon \text{ or } y_e \geq 1 - \varepsilon\}. \tag{4.27}$$

We try to two-color the nodes of G_ε with red and green, say. First we pick an arbitrary node $v \in V$ and color it red. For all neighbors w of v in G_ε we do the following: If w is not colored, w receives the color of v if $y_{vw} \leq \varepsilon$, otherwise w receives the complementary color. If w is already colored, there are two cases. If w has the same color as v and $y_{vw} \leq \varepsilon$ or if v and w have complementary colors and $y_{vw} \geq 1 - \varepsilon$, we continue. Otherwise we have found a cycle C with an odd number of edges of value at least $1 - \varepsilon$. Let Q be the set of these edges. We check whether $y(Q) - y(C \setminus Q) > |Q| - 1$. If this is the case, a violated odd-cycle inequality is found. When all neighbors of v have been considered, we pick a new, colored node, consider its neighbors, etc., and proceed in breadth-first search manner. In Figure 4.7 we show an example where the graph G consists of a square. The edge labels correspond to the values y_{uv}. We first color node 1 red and its neighbors 2 and 4 green and red, respectively. In Figure 4.7) we use dots for red-colored nodes and dash-dots for green-colored nodes. We then consider the neighbors of node 2 and color node 3 red. When coloring the neighbors of node 3, we would have to assign the color "green" to node 4. However, node 4 is already colored red and we end up with a contradiction. The corresponding violated cycle inequality reads $x_{12} + x_{23} - x_{34} + x_{14} \leq 2$.

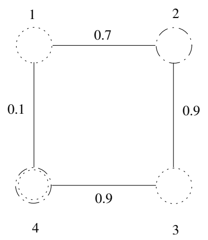


Figure 4.7: Example for the two-coloring heuristic.

If y is integral – which we check on the run – and not a cut, this procedure guarantees that a violated odd-cycle inequality will be found. So, if for an integral y, the procedure does not produce a violated inequality, y is the incidence vector of a maximum weight cut in G. The breadth-first search tree built up in this process allows us to generate the violated odd-cycle inequalities efficiently. The worst-case running time of our implementation depends on the structure of G_ε and is between $O(|V|)$ and $O(|E| \log |V|)$. Empirically it is $O(|V|)$ and extremely fast.

If we solve spin-glass problems on a grid, then the unchorded 4-cycles of the graph correspond exactly to the grid squares. We scan through all of these and check each of the eight associated cycle inequalities for violation. This can be done in $O(|V|)$ time.

In three dimensions, we also enumerate the four different possibilities for chord-less cycles along the elementary cubes in the graph. This can also be done in $O(|V|)$ time.

If we have an external magnetic field, then all three-cycles in the graph contain the corresponding field node. By scanning through all grid edges vw we check the four possible three-cycle inequalities (4.16) that can be derived from the triangle with the external field. This algorithm has $O(|V|)$ running time.

In the last heuristic we calculate a maximum weight spanning tree $T_{\max}$ of G with edge weights $|y_e - \frac{1}{2}|$. For any non-tree edge $e = ij$, we consider its *fundamental cycle* C consisting of the union of e and the unique path from i to j in $T_{\max}$ and set $Q := \{e \in C \mid y_e > \frac{1}{2}\}$. We check whether $|Q|$ is odd and the corresponding odd-cycle inequality is violated by y. Using Kruskal's algorithm, this heuristic runs in time $O(|V| \log |V|)$ on the average, and $O(|V|^2)$ in the worst case.

The above described heuristics are called in the same order as we have outlined them above. However, the following heuristic is only called if the previous one did not generate any or too few cuts.

The "exact" separation routine is called if all heuristics together found less than a certain number of cutting planes. This kind of parameterization keeps the program flexible by allowing us to test various cutting-plane generation strategies. In practical spin-glass computations on a regular grid, our heuristics for generating violated cycle inequalities work very well. They are both fast and effective. When we determine ground states of spin glasses in two or three dimensions, there is usually no need to call the exact separation routine.

If we want to determine a maximum cut in an arbitrary graph and all of the above does not produce a cutting plane, we apply the polynomial separation algorithm for bicycle-wheel separation and simple heuristics for K_p-inequality separation. However, these two separation algorithms are useful only for rather dense graphs. Two or three-dimensional grid graphs, even if we have an external magnetic field, do not contain $(2k+1)$-wheels and do not have complete subgraphs larger than a triangle. Nevertheless, in Section 4.8 we will see how these separation algorithms may turn out to be useful for these kinds of graphs also.

To keep the number of constraints small we eliminate inequalities in the following way. Whenever the objective function value has decreased by more than some specified amount compared to the previous solution value, all inequalities $a^T x \leq a_0$ with $a^T x^\star < a_0$ for the current optimum solution $x^\star$, are eliminated, otherwise no elimination is performed.

As a by-product of the described separation strategy for cycle inequalities, we obtain cuts in G that arise from (fractional) LP-solutions, e.g., the coloring heuristic and the spanning tree heuristic produce cuts in G, i.e., feasible solutions that can be used as starting solutions for improvement heuristics. Whenever we find a feasible solution with a higher objective function value than the incumbent solution $\bar{x}$ that serves as the best known upper bound for the optimum cut value at any point during the computation, we can update $\bar{x}$. We refer to this technique as “exploiting the LP”. Thus our branch-and-cut algorithm produces a sequence of decreasing upper bounds for the optimum cut value, obtained by solving LP-relaxations, and at the same time, a sequence of feasible solutions with increasing objective function value. The branch-and-cut process terminates as soon as these bounds coincide.

As an example, we show in Figure 4.8 the evolution of upper and lower bounds computed by our branch-and-cut program for a randomly generated two-dimensional ± 1 spin glass of size 50×50 with periodic boundaries. The instance took 128 seconds on a 1400 MHz Athlon computer.

Our branch-and-cut software basically results from integrating the above separation and LP-exploiting procedures into the ABACUS framework [23] for branch-and-cut algorithms that provides the branch-and-bound frame, and provisions for dealing with the LP-relaxations.

4.6 Results of Exact Ground-state Computations

For short-range Ising spin glasses like the 3d grid, the nature of the spin-glass phase is not thoroughly understood, and different models have been described in the literature. For long-range spin glasses the so-called RSB solution of Parisi [29] is assumed to hold. In the RSB picture of the spin-glass phase, there are infinitely many states with an energy difference of order $O(1)$, independent of their size. Therefore, the infinite system has many low-lying energy excitations that involve an infinite number of spins. The surface of these excitations is space filling, i.e., the fractal dimension d_s of the surface equals the space dimension d. For short-range spin glasses, the picture is less clear. It is possible that the RSB solution correctly describes the spin-glass phase in finite dimensional spin glasses. However, Fisher

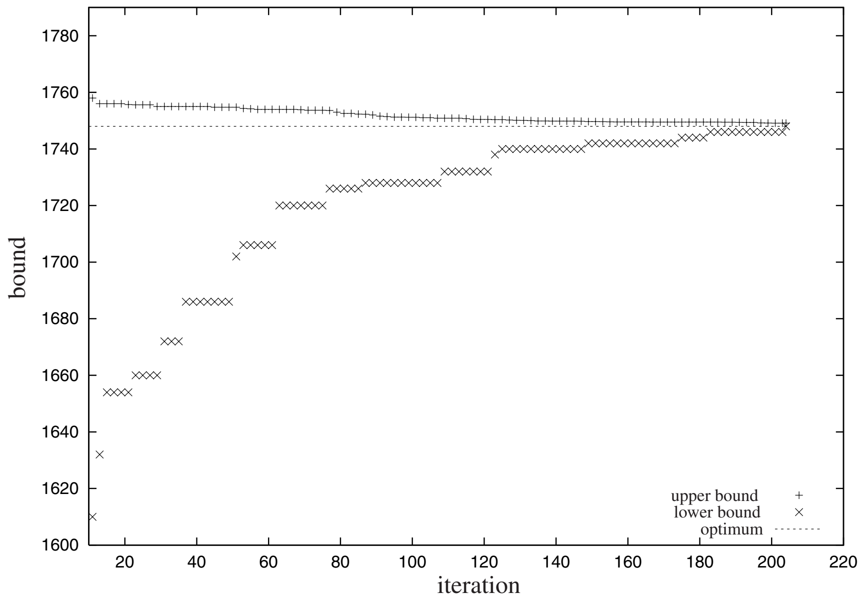


Figure 4.8: Evolution of upper and lower bounds on the max-cut value for a 50×50 grid. For clarity, the data is only shown for every second iteration.

and Huse [15] proposed the "droplet theory" which implies that the energy cost of flipping a cluster of length scale l is of order $O(l^{\theta})$ with θ being a positive constant in three dimensions. Consequently, in this picture there are no excitations of finite energy involving infinitely many spins. The fractal dimension of the surface of these excitations d_s is less than the space dimension d. Also "intermediate" scenarios have been proposed, e.g., [24], [28]. In the so-called TNT scenario we have two exponents θ and θ', which describe the growth of the energy of an excitation of scale L:

(i) θ (> 0) such that L^{θ} is the typical change in energy when the boundary conditions are changed, for example from periodic to anti-periodic, in a system of size L, and

(ii) $\theta' \sim 0$, which characterizes the energy of clusters excited within the system for a *fixed* set of boundary conditions.

Many numerical studies, mostly done with heuristic algorithms, tried to clarify the nature of the spin-glass phase, but yet none of them could undoubtedly settle this issue.

In the following we describe some results from our work together with M. Palassini and A. P. Young in which we study the nature of the spin-glass state. For more details, see [27]. We examine 3d Gaussian distributed spin glasses with free boundary conditions by following the "bulk perturbation" approach of [28] that works as follows. For a specified instance, we determine the (exact) ground state with spin configuration $S_i^{(0)}$. We choose a small parameter ε and add the amount $-\frac{\varepsilon}{N_b} S_i^0 S_j^0$ to all couplings J_{ij} (N_b is the number of bonds) and compute the ground state of the perturbed system with spin configuration $\tilde{S}_i^{(0)}$. This perturbation possibly induces a change in the ground state, since the energy of the former ground state increases by ε, whereas the energy of any other state increases by a lesser amount $\frac{\varepsilon}{N_b}\sum_{\langle i,j\rangle} S_i^0 S_j^0 S_i S_j$, which can easily be verified. We study systems of size $L \leq 12$ with different perturbations ε and high statistics.

As we are interested in how the solution changes from the unperturbed to the perturbed system, we analyze different correlation functions and show here the results only for the absolute value of the so-called *box overlap* defined as

$$q_B = \frac{1}{L_B^d}\sum_i S_i^{(0)} \tilde{S}_i^{(0)} \tag{4.28}$$

where the sum runs over the sites contained in a central cubic box of fixed size $L_B = 2$. As the box overlap is measured away from the boundaries, it should have smaller corrections to scaling and should be less sensitive to boundary conditions than, e.g., the usual *spin overlap* q in which the sum in (4.28) extends over the whole lattice. We analyze the average $\langle\cdots\rangle$ of the box overlap as well as restricted averages $\langle\cdots\rangle_c$ where the average is taken only over samples for which the unperturbed and perturbed ground states are very different, i.e., where the spin overlap satisfies $q \leq q_{\max}$ with $q_{\max}$ chosen appropriately.

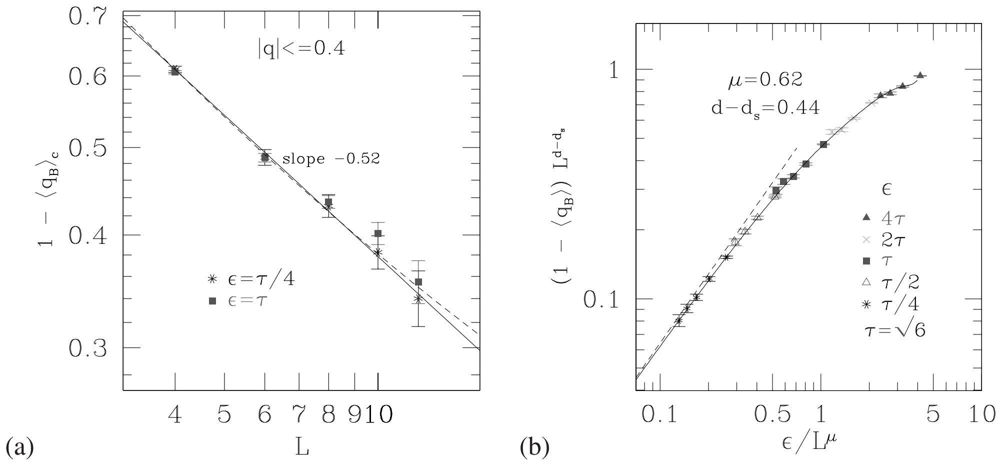


Figure 4.9: Box overlaps. (a) Logarithmic plot of the average box-overlap, restricted to samples such that $q \leq 0.4$. The lower continuous line is a power-law fit for $\varepsilon/\tau = 4$. The dashed line is the fit with $\langle 1 - q_B\rangle_c = a + b/L + c/L^2$. (b) Scaling plot of the box-overlap according to Eq. (4.30). The continuous line is a polynomial fit of order $n = 6$, which gives χ^2/d.o.f = 0.63, and a goodness-of-fit parameter $Q = 0.85$. The dashed line is the linear term of the polynomial fit, corresponding to the asymptotic behavior for $L \to \infty$.

Figure 4.9(a) shows the *restricted* average $\langle 1 - q_B\rangle_c$, with $q_{\max} = 0.4$, as a function of L for two values of ε. The data are clearly decreasing with L and close to a straight line on the logarithmic plot, consistent with the droplet or the TNT scenarios. When a large-scale cluster of spins is flipped, for large L, the probability that its surface goes across the central box is proportional to the ratio of its surface area, $\sim L^{d_s}$, to the volume, L^d. Therefore

$1 - q_B \sim L^{-(d-d_s)}$. The exponent $d - d_s$ can be read off from the log-log plot Figure 4.9(a) as the slope of the straight line. We obtain the estimate

$$d - d_s = 0.48 \pm 0.03. \tag{4.29}$$

The box overlap is expected to scale as [27]

$$\langle 1 - q_B \rangle = L^{-(d-d_s)} F_{q_B}(\varepsilon / L^{\mu}), \tag{4.30}$$

where $F_{q_B}(x) \sim x$ for small x and $\mu \equiv \theta' + d - d_s$. With θ' we denote the energy exponent that might be different from the exponent θ introduced in the droplet theory. We observe that our data scale well, according to (4.30), see Figure 4.9(b), and we obtain the best data collapse for $\mu = 0.62 \pm 0.04$. We find $\theta' = 0.15 \pm 0.7$ which is consistent with the droplet scaling picture.

However, the data also fit the RSB picture well, if we allow large corrections to scaling. Under the RSB assumption we estimate $\lim_{L\to\infty} \langle 1 - q_B \rangle_c = 0.25 \pm 0.10$ (not derived here). In this case, the good scaling behavior we observed would only be a finite-size artifact, and would disappear at larger sizes. We get comparable results when analyzing other correlation functions, e.g., the link overlap. Therefore, by current standards, it is not possible to clarify the nature of the spin-glass state by our work [27]. In order to do this, larger system sizes will be needed.

4.7 Advantages of Branch-and-cut

Usually, in order to be able to draw physical conclusions, we have to average over the disorder which means here that we calculate the ground state of many (hundreds, thousands or more) realizations of a spin glass with the same characteristics (e.g., of the same size) and study the average, higher moments or even the whole probability distribution of the physical quantities we are interested in. Obviously, for good statistics one needs sufficient computer resources.

Physical projects like the one described above, are often of this type: First, we take an instance and compute the ground state. Then we slightly change the problem (e.g., perturb some couplings), compute the new ground state and study the relations between the solutions. With methods like Monte Carlo simulations, we would have to compute both solutions from scratch. However, when doing branch-and-cut we know that the formerly optimal solution is still feasible for the new problem and can be used as a starting solution. The linear programs generated during the run of the algorithm consist only of valid inequalities for the cut polytope. The last solved linear programs might still be a "tight" relaxation for the new problem and can serve as starting relaxations. By using this information from the previous computation, we can save a considerable amount of running time.

The spin glass susceptibility is another quantity for which our branch-and-cut approach is especially suited. Suppose we have found a ground state and we wish to see how the spin configuration changes when an external magnetic field of continuously increasing strength is applied. The obvious way is to increase the field strength in fixed increments and compute a new ground state starting with the optimum solution of the previous field strength. If this previous optimum solution was found without branching, we can do even better. Linear programming techniques that are beyond the scope of this introduction can be used to determine the maximum field for which the current spin configuration, which is the optimum integral solution of some linear programming relaxation, remains optimum. Up to this field strength, the energy drops linearly and the magnetization stays constant. Beyond this point, a new optimization "warm starts" on the previous optimum solution. This technique is currently under investigation [21]. It has the clear advantage over the obvious fixed-increment approach that the exact break points in the piecewise linear energy curve are determined and much fewer reoptimizations are necessary.

In Figure 4.10 we show the energy per spin as a function of the strength h of the external field for a randomly chosen two-dimensional ± 1 instance of size 20×20. Figure 4.10(a) shows the data generated by increasing the field in fixed increments $\Delta h = 0.1$. Figure 4.10(b) shows the corresponding plot for the same instance when the data is generated by the sensitivity method described above. Here the field strengths at which the solutions change are determined exactly.

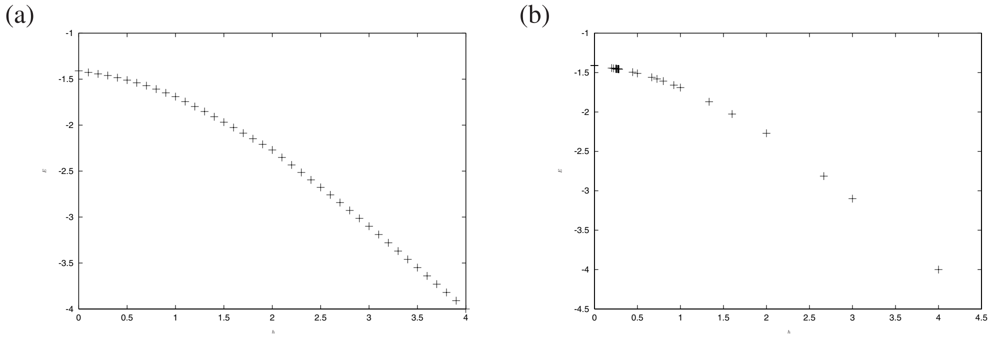


Figure 4.10: Energy E per spin as a function of the external field strength h for a randomly chosen $2d$ instance of size 20×20. (a) Energy per spin as a function of the field strength h for a 20×20 ± 1 spin glass. The field is increased in fixed increments $\Delta h = 0.1$. (b) Energy per spin as a function of the field strength h for a 20×20 ± 1 spin glass. The field strengths at which the solutions change are determined exactly.

4.8 Challenges for the Years to Come

One approach in order to reduce the running time of the algorithm is to reduce the number of subproblems to be solved. Some work in progress [21] tries to exploit the many results on the facial structure of the cut polytope $P_{\mathrm{CUT}}^{K_p}$ for the complete graph with p nodes compiled by Deza and Laurent in [13], without completing the graph with 0-weight edges, because this would render the solution practically impossible. The idea is as follows:

We are given a point $\bar{x}_n \in P_{\mathrm{CYCLE}}^{G}$ that does not belong to P_{CUT}^{G}, G being an arbitrary graph with n nodes. We want to find an inequality valid for P_{CUT}^{G} (possibly facet-defining) that is not satisfied by $\bar{x}_n$. To do so, we want to use the algorithmic and the structural results that are available for $P_{\mathrm{CUT}}^{K_p}$, the cut polytope for a complete graph.

First, by a sequence of operations on $\bar{x}_n$ that amount to switching along certain cuts and contracting node pairs corresponding to the end-nodes of integral edges, such a point is transformed to $\bar{x}_{n'} \in \mathbb{R}^{E'}$, where n' is usually much smaller than n. The point $\bar{x}_{n'}$ is always guaranteed to be outside $P_{\mathrm{CUT}}^{G'}$ but inside $P_{\mathrm{CYCLE}}^{G'}$. It can be seen as a fractional solution of a cutting-plane algorithm applied to a max-cut instance on a smaller and denser graph $G' = (V', E')$ where $|V'| = n'$.

At this point all the machinery available for the max-cut problem on complete graphs can be used for each complete subgraph K_p of G'. Therefore, some separation procedures for the cut polytope on complete graphs are applied to the restriction of $\bar{x}_{n'}$ to the edges of these components that (hopefully) generate an inequality $a_{n'}x_{n'} \leq \alpha$, valid for $P_{\mathrm{CUT}}^{G'}$ and violated by $\bar{x}_{n'}$ by an amount β.

Finally, a sequence of "lifting" procedures is applied to $a_{n'}x_{n'} \leq \alpha$ that transforms it to an inequality $a_n x_n \leq \beta$ valid for P_{CUT}^{G} and violated by $\bar{x}_n$ by the same amount β. As a by-product, one of these lifting procedures provides a way to generate facet-defining inequalities for P_{CUT}^{G}. Namely, under certain conditions, this procedure, applied to a facet-defining inequality for $P_{\mathrm{CUT}}^{G'}$, produces not only a valid, but also a facet-defining inequality for P_{CUT}^{G}.

These separation and lifting procedures enrich the description by linear inequalities of the cut polytope on arbitrary graphs and, at the same time, constitute an algorithmic tool for exactly solving the max-cut problem on these graphs.

Some preliminary experiments indicate that the above sketched procedure can indeed reduce the number of subproblems considerably. However, currently it is still too slow to give a significant overall performance gain. However, we are confident that optimizing our procedure will result in an improvement of the running time in the future. Another clear advantage of such a procedure is that it helps in solving problems without enumeration, which is a prerequisite for applying the sensitivity method discussed in the previous section.

In practical computations, around 90% of the total running time is spent in solving the linear programs by the simplex algorithm. Therefore, a topic of current research is to study the performance of branch-and-cut by replacing the simplex algorithm with fast approximate linear program solvers. The rationale for using an approximate solver is that particularly at the beginning of the optimization process, the current relaxation is not a "tight" relaxation of the cut polytope anyway. Additionally, methods like interior point solvers have a nice feature: they tend to return a solution that is "near" an optimal face, and not an optimal vertex. Intuitively this means that in the next round of separation the generated cutting-planes will cut "deeper" than if only a vertex is cut off. Therefore we expect less rounds of separation.

Recently, Barahona et al. suggested to use the so-called volume algorithm [5], [8] inside a branch-and-cut framework. The volume algorithm is a subgradient method for generating an approximate solution of a linear program that can be modified to converge to its optimal solution, see [4]. During the run of the algorithm, the current relaxation is solved exactly, always before a possible branching step takes place, in order to correctly check whether a node can be fathomed. It turns out in [8] that the running times of their volume-based code are much better for two-dimensional ± 1 spin-glass problems than their simplex based code. However, the performance for the other problem classes (e.g., three-dimensional instances) is much worse with the volume algorithm. Thus the volume algorithm seems to improve the performance only in limited cases.

In ongoing studies we replace the simplex method with an interior point solver resulting in a considerably smaller number of linear programs to be solved. Despite the smaller numbers of linear programs, the overall running time is worse than with the traditional simplex as the time for solving each LP increases. Up to now, we also find worse performance when using so-called bundle methods [31] that are also approximate solvers for which there exists a proof of convergence. Thus the question still remains of whether we can gain performance by replacing the simplex algorithm by other methods.

Acknowledgments

This work is partially supported by the Future and Emerging Technologies Programme of the European Union under contract number IST-1999-14186 (ALCOM-FT).

References

[1] K. Aardal, S. P. M. van Hoesel, A. Koster, C. Mannino, and A. Sassano, *Models and Solution Techniques for Frequency Assignment Problems*, ZIB-Report 01-40 (2001).

[2] F. Barahona *On the Computational Complexity of Ising Spin Glass Models* J. Phys. A: Math. Gen. **15** 3241 (1982).

[3] F. Barahona *The Maxcut Problem on Graphs not Contractable to K_5*, Operations Research Letters **2**, 107 (1983).

[4] L. Bahiense, N. Maculan, and C. Sagastizábal, *The Volume Algorithm Revisited: Relation with Bundle Methods*
`www.optimization-online.org/DB_HTML/2001/11/400.html`

[5] F. Barahona and R. Anbil, *The Volume Algorithm: Producing Primal Solutions with a Subgradient Algorithm*, Mathematical Programming **87**, 385 (2000).

[6] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, *An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design*, Operations Research **36**, 493 (1988).

[7] F. Barahona, M. Jünger, and G. Reinelt, *Experiments in Quadratic 0-1 Programming*, Mathematical Programming **44**, 127 (1989).

[8] F. Barahona and L. Ladányi,
`www.optimization-online.org/DB_HTML/2001/12/420.html`

[9] F. Barahona and A. R. Mahjoub, *On the Cut Polytope*, Mathematical Programming **36**, 157 (1986).

[10] M. Bellare, O. Goldreich, M. Sudan *Free bits, PCPs and Non-approximability - Towards Tight Results* Proc. of 36th Ann. IEEE Symp. on Fundations of Computer Science, IEEE Computer Society **422-431** (1995).

[11] E. Boros and P. L. Hammer, *The Max-Cut Problem and Quadratic 0/1 Optimization*, Polyhedral Aspects, Relaxations and Bounds, Annals of Operations Research **33**, 151 (1991).

[12] V. Chvátal, *Linear Programming*, (W. H. Freeman and Company, New York, 1983).

[13] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*. Algorithms and Combinatorics Vol. **15** (Springer-Verlag, Berlin, 1997).

[14] M. Elf, M. Jünger, and G. Rinaldi, *Minimizing Breaks by Maximizing Cuts*, Report No. 2001.409, Angewandte Mathematik und Informatik, Universität zu Köln, to appear in: Operations Research Letters, 2003.

[15] D. S. Fisher and D. A. Huse, J. Phys. A. **20**, L997 (1987); D. A. Huse and D. S. Fisher, J. Phys. A. **20**, L1005 (1987); D. S. Fisher and D. A. Huse, Phys. Rev. B **38**, 386 (1988).

[16] A. Galluccio, M. Loebl, and J. Vondrak, *Optimization via Enumeration: A New Algorithm for the Max Cut Problem*, Mathematical Programming **90**, 273 (2001).

[17] B. Gerards, *Testing the Odd Bicycle Wheel Inequalities for the Bipartite Subgraph Polytope*. Mathematics of Operations Research **10**, 359 (1985).

[18] M. Grötschel, L. Lovász, and A. Schrijver, *The Ellipsoid Method and its Consequences in Combinatorial Optimization*, Combinatorica **4**, 169 (1981).

[19] A. K. Hartmann, H. Rieger, *Optimization Algorithms in Physics* (Wiley-VCH, Berlin, 2002)

[20] M. X. Goemans and D. P. Williamson, *.878-approximation Algorithms for MAX CUT and MAX 2SAT.* ACM Symposium on Theory of Computing (STOC) (1994).

[21] M. Jünger, G. Reinelt, and G. Rinaldi, *Lifting and Separation Procedures for the Cut Polytope*, Technical Report, Universität zu Köln, in preparation.

[22] M. Jünger and G. Rinaldi, *Relaxations of the Max Cut Problem and Computation of Spin-Glass Ground-States*, in: P. Kischka et al. (eds.), *Operations Research Proceedings 1997*, p. 74, (Springer, Berlin, 1998).

[23] M. Jünger and S. Thienel, *The ABACUS System for Branch-and-Cut-and-Price Algorithms* in Integer Programming and Combinatorial Optimization, Software Practice and Experience **30**, 1325 (2000).

[24] F. Krzakala and O. C. Martin, Phys. Rev. Lett. **85**, 3013 (2000).

[25] A. H. Land and A. G. Doig, *An Automatic Method for Solving Discrete Programming Problems*, Econometrica **28**, 497 (1960).

[26] M. Padberg, *Linear Optimization and Extensions*, (Springer-Verlag, Berlin, 1995).

[27] M. Palassini, F. Liers, M. Jünger, and A. P. Young, *Low Energy Excitations in Spin-Glasses from Exact Ground-States*, Phys. Rev. B 68, 064413 (2003).

[28] M. Palassini and A. P. Young, Phys. Rev. Lett. **85**, 3017 (2000).

[29] G. Parisi, Phys. Rev. Lett. **43**, 1754 (1979); J. Phys. A **13**, 1101, 1887, L115 (1980); Phys. Rev. Lett. **50**, 1946 (1983).

[30] L. Saul and M. Kardar, Phys. Rev. **E 48**, R3221 (1993); Nucl. Phys. B **432**, 641 (1994).

[31] H. Schramm and J. Zowe, *A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results* SIAM J. Opt. **2**, 121 (1992).

5 Counting States and Counting Operations

A. Alan Middleton

5.1 Introduction

Combinatorial optimization algorithms developed by computer scientists can be applied to a number of problems in condensed matter and statistical physics. A direct application is finding the classical ground-state configuration for a many-degree-of-freedom system with quenched disorder. Examples of such complex systems with frozen-in disorder include many models of random magnets and models of elastic lines or surfaces pinned by a disordered potential. By comparing ground-state configurations found for distinct boundary conditions (e.g., periodic vs. anti-periodic), one can find the energy and shape of domain walls in these disordered models. Averaging the numerical results for ground-state or domain-wall energies over many samples, gives the statistical properties of the ground state and of the large-scale excitations above the ground state. By simulating millions of smaller samples and thousands of samples with millions of degrees of freedom each, low enough statistical errors can be achieved that systematic errors dominate the analysis. The dependence of the results on system size and other parameters can be compared quantitatively with analytic computations, both to check predictions for exponent values and to verify scaling relations between these dependencies.

Given the speed of algorithms and hardware, it is now possible to broaden the types of questions about disordered systems for which precise answers can be obtained. Such questions include those about the general structure of the energy landscape, such as the degeneracy of the ground state in the thermodynamic limit or barriers between the low-lying states. The thermodynamic limit, for example, can be studied by computing the ground-state configurations for a sequence of samples of increasing size, each sample being a subsample of an infinite sample. This type of computation extends the direct application of optimization algorithms. There is a greater need for very efficient algorithms, the motivation to explore algorithms that find ground states for many similar samples (for example, with slightly different disorder, boundary conditions or external perturbations), and the increased use of code organization and data analysis that is complex in itself, with the optimization algorithm treated as a core routine. The "new" optimization algorithm, then, is based upon utilizing standard optimization methods to the fullest, organizing the computations for single samples within a broader framework and applying extensive analysis to the physical results and algorithm timings. (For some examples of analysis of results and timings, see Refs. [34, 40, 47], among others cited in this Chapter.)

The most precise numerical results available are for combinations of models and questions that can be addressed with known polynomial-time algorithms. It is often not obvious which questions can be studied using polynomial-time algorithms and which cannot. Just finding

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

the ground state exactly can be NP-hard and thus impracticable to study [64]. Even when the ground state can be found in polynomial time, computing quantities that characterize the energy landscape in the same model, such as the highest energy state, the partition function, or the height of barriers between low-lying states, may be NP-hard. It is unclear whether the distinction between P and NP-hard optimization problems, so important in complexity theory, leads to distinct behavior of the physical models. Regardless of the physical importance of this distinction, it is clear that the discrimination between P and NP-hard optimization problems has been extremely useful in organizing simulations.

This chapter first describes classical models with quenched disorder, inspired by the study of "pure" models and the questions that they raise. The application of optimization algorithms to disordered models is discussed in general, in Section 5.2. An important example, finding ground states for the random-field Ising magnet, is reviewed within Section 5.3. The use of these algorithms in studying the degeneracy of ground states is then described in some detail in Sections 5.4 and 5.5. The deduced low degeneracy of the ground state and scaling laws for the physical system of interest can in turn be applied to the statistical study of the algorithms themselves. The degeneracy of the ground state also has implications for the running time of the algorithm. For these polynomial-time algorithms, there can be a peak in running time at the physical transition. As reviewed in Section 5.6, the understanding of diverging length scales at phase transitions can be applied to explaining the slowing down of the non-physical dynamics of the optimization algorithm and to exploring parallel algorithms. Further directions for this field are discussed in Section 5.7.

5.2 Physical Questions about Ground States

5.2.1 Homogeneous Models

In order to understand disordered materials, it is useful to briefly review microscopically homogeneous models. The simplest non-trivial description of an extended magnetic system is the ferromagnetic Ising model. The electronic spins $\{s_i\}$ in this model for a magnetic material have only two discrete directions, $s_i = \pm 1$, where the N magnetic ions are indexed by $i = 1, \ldots, N$. Interactions lower the energy when neighboring atoms have parallel magnetization. In the pure model, the topology of these nearest-neighbor connections is uniform. For example, i might be the index of a node of a regular lattice in finite-dimensional space, with neighbors being pairs of nodes with minimal spatial separation. The Hamiltonian is $H_I = -J \sum_{\langle ij \rangle} s_i s_j$, where $\langle ij \rangle$ lists all nearest-neighbor pairs i and j. In a finite system, the ground-state configuration is a choice for $\{s_i\}$ that minimizes H_I. Clearly, there are exactly two ground-state configurations: $s_i \equiv +1$ and $s_i \equiv -1$, where the spins are chosen uniformly "up" or "down". This two-fold degeneracy holds independent of system size and hence is well-defined in the limit of infinite system size.

Statistical mechanics considers not only the ground-state configuration but the excited states. One type of excitation is given by flips, relative to the ground state, of single spins or small localized clusters. These have finite energy and thus finite density even at small temperatures. However, these excitations have little effect on the large-scale behavior at low temperatures. Under the coarse graining of the spins that is considered in the renormalization group [17], these excitations are irrelevant. The lowest energy system-spanning excitations, domain walls, split the sample into regions of mostly up and mostly down spins. In a finite sample of linear size L, the interfaces have energy $E \sim L^{\theta}$, with $\theta = d - 1$ in the pure d-dimensional Ising model. In the Ising model for lattices of dimension $d = 2$ and higher, there is a critical temperature above which the large-scale excitations are common enough to disorder the spins and the observed state is no longer approximated by the ground state. Below this critical temperature, however, domain walls vanish, as their energy is large compared to their entropy, and the spins are correlated over large distances. The ground state is therefore a useful starting point for understanding the low-temperature behavior. In addition to understanding the statics, the dynamics of these walls are important in understanding large-scale relaxation at low temperature [15].

5.2.2 Magnets with Frozen Disorder

Experiments [59] show that heterogeneous magnets, with missing or impurity atoms, or even long-range correlated defects such as grain boundaries, have low-temperature properties that are qualitatively distinct from homogeneous materials. Most notably, the dynamics of the spin correlations are much slower than in pure materials (even though there may be no net magnetization). This relative slowness arises from the complex relationship between spin configurations and free energy. The reduction in symmetry from the pure case leads to many more energy minima that are stable to the flipping of a small number of spins. At finite temperature, the rearrangement of spins takes place by a combination of direct relaxation to these local minima and thermal activation over barriers that separate the basins of attraction associated with each local minimum of the free energy. Translation invariance in the pure case implies that a domain wall can move with a sequence of local moves that does not usually (but see [55]) require overcoming a large barrier. Due to the lack of translational symmetry in disordered magnets, low-lying states need not be related by local low-cost changes. The barriers to domain-wall motion can therefore be quite large. A set of spins gets stuck for a long time in a low-lying energy state before thermal effects can activate it to jump to an even lower energy state. This is consistent with numerical and analytic work [5, 14] that indicates that the barrier to a region of domain wall of size ℓ scales as ℓ^{ψ}, with the barrier exponent $\psi \geq \theta$ in disordered magnets.

As in the pure case, it is generally believed that, for sufficiently high dimension and in most models, there is a low-temperature phase described by a zero-temperature fixed point. The low-temperature phase is dominated by disorder, not by thermal fluctuations. Time-independent quantities, such as the distribution of energies for domain walls or time-averaged correlation functions, can then be studied at zero temperature. The ground-state configuration of the model is therefore of central interest. (This approach fails in some ∞-dimensional models, where there is a lack of universality in the zero-temperature state [12]. Universality is usually expected to be recovered when considering generic distributions of disorder in finite dimension).

The relaxation of the disordered material is inhibited by the same feature that slows down numerical simulations: the ground state is often much less obvious than in the pure case. This is another aspect of the complexity of the relationship between spin configuration and energy. The crucial feature of this complexity is the *competition* between various terms of the

Hamiltonian which results in *frustration*, the inability to separately minimize all of the terms of the Hamiltonian. For example, consider the Hamiltonian for the random-field Ising model (RFIM),

$$H_{\mathrm{RF}} = -J \sum_{\langle ij \rangle} s_i s_j - \sum_i h_i s_i, \tag{5.1}$$

which is the same as the Ising Hamiltonian H_I, but for the introduction of a random field h_i. The external field h_i is a *quenched* random variable, that is, it changes over time scales much longer than the s_i do, if at all. It is taken to have a probability distribution independent of i; here, it will be assumed that the mean value is $\overline{h} = 0$ (overlines indicate averages over the disorder distribution). This random field gives each spin a preferred direction. The ratio $\Delta = (\overline{h^2})^{1/2}/J$ can be used to characterize the strength of the disorder, given a form of the distribution for h_i (e.g., uniform, bimodal, or Gaussian). The strength of the disorder determines the phase of the RFIM. Note that this Hamiltonian can be rewritten as

$$H_{\mathrm{RF}} = H_{\mathrm{sat}} + 2 \sum_{\{(i,j) | s_i \neq s_j\}} J + 2 \sum_{\{i | h_i s_i < 0\}} |h_i| \,, \tag{5.2}$$

where H_{sat} is the minimal conceivable energy, if all interactions could be satisfied. The competition in this model is between the h_i, which favor a mix of spin directions, and the couplings J, which favor aligned spins. This competition implies that $\min_{\{s_i\}} H_{\mathrm{RF}} > H_{\mathrm{sat}}$, except when all h_i have the same sign.

The exact ground-state configuration depends on the realization of the h_i in a particular finite sample. If the typical magnitude of the h_i is much greater than J, the spins will, except in rare isolated clusters, individually align with the sign of h_i in the minimal-energy configuration. There will be a single unique ground-state configuration. In contrast, if J is larger than the typical h_i, the spins will tend to align over large regions. This ferromagnetic phase will have a net magnetization that is *fixed in direction in a given finite sample*, as the disorder realization breaks the up-down spin symmetry. If this finite sample is expanded in volume, however, by adding more spins and random fields, the direction of the magnetization can change sign. The importance of these changes in exploring the infinite-volume limit will be discussed in Section 5.3.3.

Another example of a random magnet is the random-bond Ising magnet (RBIM). Here, there is no external random field, but the strength of the couplings between spins can vary, as modeled by the Hamiltonian

$$H_{\mathrm{RB}} = -\sum_{\langle ij \rangle} J_{ij} s_i s_j, \tag{5.3}$$

where the random couplings $J_{ij} \geq 0$ are ferromagnetic (spins tend to be aligned), but vary in magnitude. Here, the ground state with uniform fixed or open boundary conditions is clear (uniform s_i), as there is no competition between the interactions among spins. Boundary conditions can introduce competition, however. If anti-periodic or mixed fixed boundary conditions are used, bonds will be unsatisfied, even in the ground state. In these cases, the large-scale low-energy excited states, characterized by domain walls, are not trivially apparent. In particular, the domain walls are not "flat" as in the pure Ising model, due to the competition between effective elastic forces arising from the tendency of the spins to align, which tend to minimize the size of the domain wall or set of unsatisfied bonds, and the randomness of the J_{ij}, which favors roughness of the domain wall. Note that, even though the ground state of the RBIM is simple, the complexity of domain-wall structure leads to very slow (glassy) dynamics in this model, say upon cooling from the high-temperature paramagnetic phase or in response to a small change in an external magnetic field.

A problem of great interest, is the physics of spin glasses [59]. (Also see Chapters 4 and 3 for advances in optimization algorithms for this NP-hard problem). The Edwards–Anderson spin glass Hamiltonian H_{SG} is identical to H_{RB}, except that the distribution of the J_{ij} can be negative. Competition (frustration) arises from closed loops of bonds in this model that have a negative product of J_{ij}: not all bonds on these loops can be satisfied. In the spin-glass phase without external field, there is global spin inversion symmetry, so there are at least two degenerate ground states.

The physical dynamics of all of these magnets is extremely slow, due to lack of symmetry, the large number of metastable configurations, and the large barriers to domain-wall motion [13, 57]. This lethargy results from continuity: domain walls move by a sequence of small changes. Magnetic spins can only very rarely make lucky global guesses and they do not employ complex data structures. In the next section we will see that statistical physicists are not constrained by physical processes, however.

5.3 Finding Low-energy Configurations

5.3.1 Physically Motivated Approaches

One approach to finding optimal configurations for disordered magnets and related systems is to use simulation methods directly inspired by physical dynamics. Given sufficient computer resources and a sufficiently accurate model, the simulation dynamics would replicate those of the physical system. Even lacking these conditions, it might be hoped that approximate models and moderate resources might give sufficient accuracy for the problem of interest. Such direct approaches are often an important first step in a study and are sometimes the only known way to find useful results. One example of such a method is simulated annealing. In this method, the temperature of a sample is gradually lowered, numerically. At each temperature, Monte Carlo sampling is used to generate configurations, usually through local moves, such as flips of individual spins (algorithms that flip large clusters of spins, while very useful for pure systems, have not been sufficiently effective for disordered systems [10]). Cooling the system while carrying out spin flips that maintain detailed balance at each temperature relaxes the configuration to a relatively low-energy configuration. In the limit of adiabatic cooling, this method will lead to the ground-state configuration.

The difficulty with such methods, though they are of practical import for smaller systems, is that the time to cool a disordered sample effectively adiabatically requires a number of Monte Carlo steps that can grow exponentially with L, the size of the system. As the barriers grow as L^{ψ}, the equilibration times τ are expected to grow at least as rapidly as $\tau \sim \exp(L^{\psi}/T)$ (neglecting the possibility of exponentially many easily accessible paths).

Even when the energy barriers are low at large scales, as in the 2D spin glass, domain walls get frozen in during cooling, as the Monte Carlo flips are not sufficient to efficiently move domain walls to their lowest energy state, due to the complexity of the landscape and the large number of available configurations ("entropic barriers"). The low-energy configuration found at some slow but finite cooling rate will be likely to have an energy *density* near that of the true ground state, but the cooling process "freezes in" domain walls at some characteristic scale that depends on the rate of cooling. Hence, though relatively accurate estimates of the ground-state energy density may be found, the ground-state configuration itself can be hard to determine. The necessary cooling rate has been explored in many numerical simulations – recent simulations (e.g., [5]) confirm that the length scale grows at least as slow as logarithmically in time, consistent with analytic expectations [14].

Of course, this slow dynamics mimics what is seen experimentally [59]. One might then ask if this limitation to simulations is a positive feature. In part, the answer is yes, this first step is sufficient for many purposes. However, as the microscopic dynamics of real materials takes place at extremely high frequencies, large volumes of equilibrated spins can be created experimentally, with a relatively low density of domain walls relative to the lowest free-energy state. It is not currently possible to span the experimental range of length and time scales using simulations with local dynamics. Note that simulated annealing and related Monte Carlo techniques have been applied to optimization problems in computer science and give good heuristic solutions in some cases [25], as well as the useful approximations for disordered models from statistical physics mentioned here.

5.3.2 Combinatorial Optimization

In order to study the ground-state configuration for physical systems with competition or frustration over large volumes, other methods, less physical in structure, can be used. One can ask if, by using unphysical "dynamics", it is possible to determine, say, the ground state exactly. One approach is to look for an expression of the ground-state problem as a tractable *combinatorial optimization problem* [2, 9, 20]. If the physical model and energy function can be translated, say, to an easily solvable optimization problem on a graph, large model systems can be directly solved. The physical problem is expressed as a discrete optimization problem: determining the extremum of a function, the Hamiltonian, over the set of possible configurations. Part of the theoretical challenge in studying optimization for a complex system is determining whether this optimization problem can be solved in time polynomial in the length of the problem description. A constructive proof of this gives an algorithm for solving the problem. In many cases, it has been shown that the minimization of the Hamiltonian is an NP-hard problem and hence likely to be unsolvable in polynomial. This classification relies upon a sensible definition of the length of the problem description [64]. For the disordered Hamiltonians from statistical physics, the description of the problem is given by the lattice geometry and the random variables, such as bond strengths or random fields. Each degree of freedom requires, at most, a number of bits polynomial in N (see Section 5.5.1), so that the length of the problem description is polynomial in the number of degrees of freedom.

Sometimes the mapping of a physical problem to a solved computer science problem is direct. For example, finding the ground-state configuration for an elastic line in a disordered background can be solved using a shortest-path algorithm [23]. Finding the ground-state configuration for many non-overlapping lines in a disordered background is less clear and first attempts to solve this problem took a time exponential in the number of lines. Mappings to a maximum-weight assignment (complete matching) problem [60] and minimum-cost flow problems [45, 46] were found, reducing the problem of the ground state for many lines to a polynomial-time problem.

The 2D spin glass, where the lattice can be embedded in a plane, with nearest neighbors coupled by bonds that lie in the plane, can also be solved in polynomial time, using non-bipartite matching [3]. In fact, the full density of states can be computed (and hence the partition function at any temperature) on graphs that are planar, toroidal, or embedded on any finite genus surface [16, 49] in time polynomial in L.

There are many other algorithms for similar problems. Reviews [2, 20] give some of the details. Generally speaking, though, these polynomial-time methods either use alternate representations of physical configurations or work in an extended space containing configurations that cannot be represented as physical configurations. In either case, additional variables can be added that are used by the algorithm and updates are carried out through carefully constructed non-local moves. In algorithms that work with a physical space of configurations, these non-local moves improve the solution until no more energy-lowering updates are available. In the case of algorithms that work with a space that includes non-physical configurations, the moves or iterations bring the representation to a configuration that is both optimal and physical (that is, it lies on the subset of physical configurations within the extended space). Such algorithms circumvent the barriers to the ground state by taking a route monotonic in cost toward the global, physical minimum *in the extended space.* The push-relabel algorithm for the RFIM, described in Section 5.3.3 is an example of such an algorithm. In this algorithm, at least some physical spins are undefined until the algorithm terminates.

Heuristic methods are available (and discussed in other parts of this book) to solve typical cases of NP-hard problems either approximately or exactly (but with potentially very large times). This chapter, however, will focus on those problems which can be guaranteed to be exactly solved in polynomial time. Though this set of models does not include, for example, 3D spin glasses, the results for these models are of significant interest in themselves and also provide a guide to the potential difficulties that might arise in studying NP-hard problems.

It is an open problem as to how the computationally important distinction between P and NP-hard optimization problems is related to physical questions. For example, does this distinction affect the qualitative picture of ground states and dynamics? Problems whose ground states can be found exactly in polynomial time still exhibit glassy (slow dynamical) behavior with many metastable states. Barrier heights can grow with system size L for problems of both classes.

The primary example that I will discuss here is the random-field Ising magnet, as it is one of the first problems that was addressed by combinatorial optimization and clearly exemplifies some of the general issues. The RFIM also has the advantage that the dynamics of the algorithm can be directly explained [32]. I will also describe some results for matching problems and problems that can be posed in terms of finding domain walls in random-bond magnets.

5.3.3 Ground-state Algorithm for the RFIM

It was noted some time ago [44] that the ground state of the random-field Ising model can be found in polynomial time by solving a corresponding combinatorial optimization problem. Though the details of this mapping and the relevant algorithms are now well known, a brief review of an algorithm used for the RFIM is given here, as the details will be important in understanding the dynamics and critical slowing down of the algorithm. In addition, the variant of the algorithm that has been used in the study of the critical slowing down and in some ground-state studies is slightly different from the form initially used. For the proofs needed to justify the validity of the algorithm described here and polynomial bounds on the number of operations needed to find the ground state, the reader is referred to computer science texts, such as the book by Cormen, et al. [9]. For application details, also see earlier reviews of applications of combinatorial optimization methods to statistical physics [2, 20].

In seeking the minimal-energy configuration RFIM, the goal is to find a minimal-cost set of "bonds" to "break". Here, the "bonds" include both the physical nearest-neighbor interactions of strength J and auxiliary bonds that connect each spin to one of two external fixed spins, $s^+ \equiv 1$ and $s^- \equiv -1$. These auxiliary bonds are used to express the random external magnetic field as ferromagnetic interactions: if a spin s_i is subject to a random field $h_i > 0$ ($h_i < 0$), it can be considered connected to the fixed-up spin s^+ (respectively, the fixed-down spin s^-) by a bond of magnitude $|h_i|$. This device allows the RFIM Hamiltonian to be mapped to that of a random-bond Ising ferromagnet, with the constraint of two fixed spins. The ground state is then given by the minimum energy domain wall between up and down spin regions. An example of such a domain wall in the extended system is shown in Figure 5.1. Applying Eq. (5.2), the ground-state energy is $H_{\mathrm{sat}} + 2\sum_{e\in C} w_e$, where the weights of the edges e are J or $|h_i|$, depending on whether e is a physical or an auxiliary bond, respectively, and C is the set of bonds crossed by the domain wall.

The problem of finding a minimal-energy domain wall that separates two given spins, given ferromagnetic bonds (all non-negative values), is exactly the minimum-s-t cut problem in graph theory. Given a graph G with vertex set V and edges E, a cut is the partition of V into two sets (here, the partition will be into sets of up and down spins). Given two nodes $s, t \in V$, an s-t cut requires s to be in one partition and t in the other (here, $s \equiv s^+$ and $t \equiv s^-$). The cost of the cut is the total of the weight of edges that must be removed to partition the nodes. Note that this correspondence is applicable to any RFIM, regardless of its dimensionality.

By the standard mincut-maxflow theorem (see, e.g., [9]), the minimum-s-t cut problem can be mapped to another optimization problem, the maximum-flow problem. For the RFIM, this problem is defined on the directed graph $G' = (V', E')$ with vertices $V = \{i\} \cup \{s^+, s^-\}$ and directed edges E'. For each physical bond connecting spins i and j, there are two directed edges in E', (i, j) and (j, i), each with weight J. In addition, for each node i, there is a directed edge (s^+, i) when $h_i > 0$ or a directed edge (i, s^-) when $h_i < 0$; this edge has weight $|h_i|$. Given a graph, a flow $f(e) = f((a, b))$ is a number defined for each edge $e = (a, b)$ that is constrained to the range $0 \leq f(e) \leq w(e)$ and that satisfies a conservation constraint, i.e., zero total flow at each internal node: $\sum_{\langle ij\rangle} f(i,j) + f(s^+, i) + f(i, s^-) = 0, \forall i \in V' \backslash s^+, s^-$. This flow field gives a conserved fluid flow on the directed edges that is subject to maximum flow constraints at each edge. The solution to the max-flow problem maximizes total "flow" between s^+ and s^-. The mincut-maxflow theorem states that the value of the maximum flow

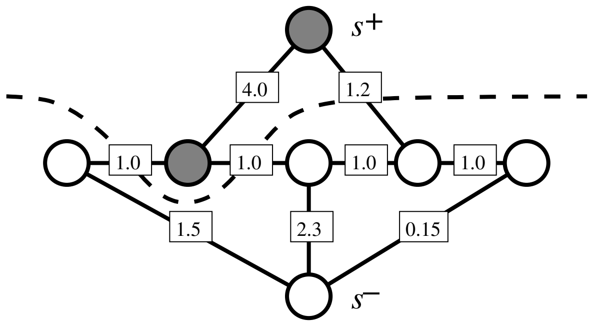


Figure 5.1: Representation of the ground-state problem for the RFIM as an RBIM domain wall or minimum-cut problem. The physical spins are the five nodes in the single row in the figure, while the fixed external spins are s^+ and s^-. The physical RFIM coupling is $J = 1.0$. A spin with $h_i > 0$ ($h_i < 0$) is connected by an auxiliary coupling of strength h_i ($-h_i$) to s^+ (s^-). The weights of each bond are indicated: the random fields are, from left to right, $h = -1.5$, $+4.0$, -2.3, $+1.2$, and 0.15. In the ground state, the interfacial energy between up-spin and down-spin domains is minimized, i.e., the spins are partitioned into two sets with minimal total cost for the bonds connecting the two sets. The dashed curve indicates the minimal-weight cut. The white (dark) nodes indicate up (down) spins in the ground-state configuration.

is equivalent to the value of the minimum cut, as the cut provides the value of the collective bottleneck for the flow from s^+ to s^-.

The push/relabel algorithm introduced by Goldberg and Tarjan [18] is an algorithm for solving the max-flow problem. The algorithm determines the maximal flow by successively improving a "preflow". When the preflow can no longer be improved, it can, if desired, be converted to a maximal flow, proving the correctness of the algorithm. This conversion is not needed to determine the ground state of the RFIM, however. A preflow is an edge function $f(e)$ that obeys the range constraint $0 \leq f(e) \leq w(e)$, but the conservation constraint at each node is relaxed: the sum of the $f(e)$ into or out of a node can be non-zero at internal (physical) nodes. The amount of violation of conservation at each node v give "excesses" $e(v)$. The basic operations of the algorithm, push and relabel, are used to rearrange these excesses. I will consider a variation that excludes the auxiliary nodes s^+ and s^- and the edges to these nodes. This variant can significantly reduce memory usage. Its focus on the physical nodes makes the correspondence between the physics and algorithm more clear.

The preflow algorithm uses auxiliary fields $r(i, j)$, the residual flow for each directed edge, $e(i)$, the excess at each physical node, and $d(i)$, a distance or potential defined at each node that guides the rearrangement of the excess. The algorithm for the RFIM starts by assigning residual flow values $r(i, j) = r(j, i) = J$, for nearest neighbors $\langle ij \rangle$. All excesses at each node are set equal to the external field: $e(i) = h_i$ for all i. Initially, $d(i) = 0$ for all nodes with $h_i = e(i) < 0$. To define initially the $d(i)$ on the nodes with $h_i > 0$, a *global update* can be carried out. The global update sets each $d(i)$ to be the *minimal distance*, measured in number of edges with $r(i, j) > 0$, from each node with non-negative excess to a node with negative excess. This update is implemented using a breadth-first search: starting from the set of nodes with negative excesses, arcs with positive residual flow $r(e)$ are followed *in reverse* from the set of sites which have height assignments. Sites i which cannot be reached using these reversed arcs are assigned $d(i) = \infty$.

Given this initialization, the algorithm is now executed by performing simple local operations, push and relabel, and by occasional global updates. The order of the operations and the frequency of global updates do affect the speed of the algorithm, but not its correctness. The solution will be found in polynomial time regardless of the heuristics used to make these choices. The goal of the local operations is to rearrange the positive excesses, in a fashion guided by the $d(i)$, to establish a maximal flow. A push operation can be executed when $d(i) = d(j) + 1$ for some neighboring pair (i, j) with $r(i, j) > 0$ and $e(i) > 0$. The push moves as much excess as possible through the arc that has residual flow: the excess is reduced via $e(i) \leftarrow e(i) - \delta$ and the residual capacities are adjusted using $r(i, j) \leftarrow r(i, j) - \delta$ and $r(j, i) \leftarrow r(j, i) + \delta$, where the amount of excess pushed is $\delta = \min(r(i, j), e(i))$. The relabel operation $d(i) \leftarrow \min_{\{j|\langle ij\rangle\}} d(i) + 1$ sets the distance at i sufficiently high that a push can be carried out; this step can be executed when $e(i) > 0$ and the condition for a push is not met. In this variant of the algorithm, the negative excess is reduced in magnitude or even changes sign when positive excess is pushed on to a node with negative excess, but the negative excess is not itself rearranged (though such an approach is possible). The sequence of steps in the application of the push/relabel algorithm to finding the ground state of a 1D RFIM is presented in Figure 5.2.

The push-relabel algorithm for the RFIM is extremely fast in practice and its application is often limited by memory, rather than time, constraints (for a parallel implementation of push-relabel, see [53]). Besides its practical utility in addressing questions about the RFIM, it also has interesting dynamics, as discussed in Section 5.6.

5.4 The Energy Landscape: Degeneracy and Barriers

Studying the energy landscape for a complex system naturally raises issues of computational complexity and efficient algorithms. As reviewed in the previous section, especially for the random-field Ising magnet, finding the ground-state energy and a ground-state configuration can be accomplished in polynomial time for many models, even with competition caused by quenched disorder.

However, being able to find a ground state does not always imply that studying the full structure of the energy landscape is easy. A natural set of questions about the structure of the energy landscape includes:

- What is the degeneracy of the ground state? That is, how many configurations have the minimum energy? Are the minimal-energy configurations all near each other in configuration space or are they separated by large distances?
- What is the minimal energy of large-scale excitations, such as boundary walls between two low-energy states?
- What is the full spectrum and density of states? What are the non-minimal-energy values and how many configurations have each energy? This information is needed to compute

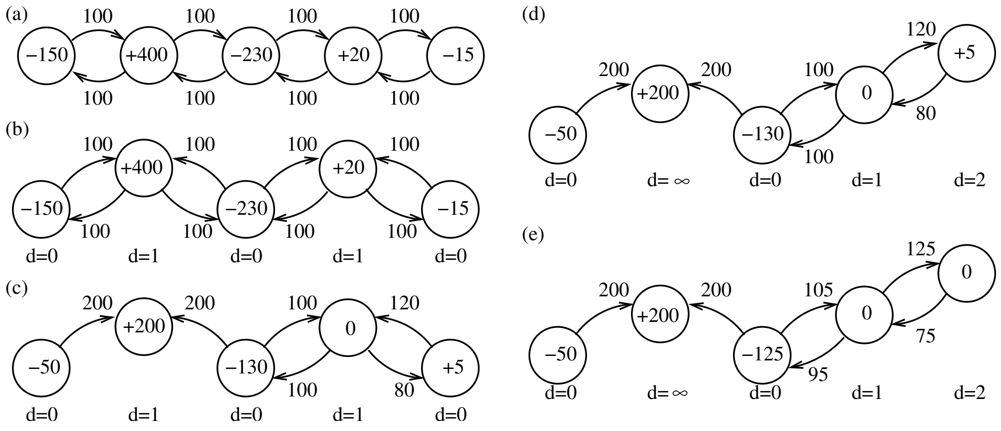


Figure 5.2: A sequence of operations that finds the ground state for the RFIM sample shown in Figure 5.1. Residual arcs, with capacities indicated, connect nodes with excess values indicated within the circles. (a) Initial excesses are assigned to each node, indexed $i = 0, \ldots, 4$ from left to right, with excess $e(i) = h_i$ and residual arc capacities are set to $r(i, j) = r(j, i) = J$ (the h_i and J have been multiplied by 100 to give integer values). (b) The distance field $d(i)$ is set by a global update, which computes, for each node with $e(i) > 0$ the minimal distance along residual arcs with positive flow to a node with $e(i) \leq 0$. The nodes are offset vertically by a distance proportional to $d(i)$ (the nodes $i = 1$ and $i = 3$ have $d(i) = 1$). (c) The auxiliary fields after three push operations. Two pushes of value $\delta = 100$ through arcs to the left and right from the $i = 1$ node and one push with $\delta = 20$, from $i = 3$ to $i = 4$ were carried out. (d) The configuration after two relabels of the rightmost node (this configuration would also be obtained from a global update of (c)). (e) The final preflow configuration, with values for the distances d, reached by pushing excess from node $i = 4$ to node $i = 3$ and then from $i = 3$ to $i = 2$, with $\delta = 5$ in each case. The second node ($i = 1$) is not reachable by reversed arcs from nodes with negative excess and so has height $d = \infty$. This corresponds to an up-spin. The remaining nodes are reachable by reversed arcs from negative excess nodes and therefore correspond to down spins, in the ground state of the RFIM.

the partition function, for example, to study finite-temperature phase transitions, or to study the density of states in disordered quantum magnets [26].

- For dynamics, it is important to understand how the configurations are connected to each other, via sequences of spatially localized changes (for example, spin flips). Thermally activated processes may be studied by finding the barrier between two low-lying configurations. Can this barrier value be determined?

Some of these questions might be easily answered, while others may not, depending on the *exact* details of the model that is being studied.

For example, consider the problem of finding the shortest path on a graph with random weights. Physically, this might correspond to finding the lowest-energy configuration of a flux-line in a type-II superconductor or a polymer in a disordered background, constrained to connect two points. It also corresponds [23] to studying domain walls in a 2D random-bond Ising magnet (with random $J_{ij} > 0$). The ground state for this problem can be solved very quickly using transfer-matrix methods for directed polymers [23] or more generally using shortest-path algorithms such as the Dijkstra algorithm or adopting a max-flow algorithm [8,9,50,52]. The full partition function for a directed polymer (where the steps are all positive along one of the spatial directions) can be computed [23], so that the statics can be characterized in polynomial time for an arbitrary sample. In contrast, finding the barrier between two configurations for a polymer in a disordered background is NP-hard. Very briefly, this can be shown by asking whether the barrier between two configurations for a line object is zero. A lattice can be constructed [31] where the barrier will be zero if and only if the energy of the polymer can first be lowered by a certain amount q, so that a move that costs energy q can then be executed. The graph is constructed such that the lowering of energy by q via local moves of the loop is possible if and only if an instance of a problem, planar 3SAT, can be decided. As this latter problem is NP-complete, the barrier problem for the motion of the loop is NP-hard. Besides the example of the loop considered above, an abstract mapping can be made to problems such as the NP-hard register-allocation problem [54], which optimizes the amount of memory used to evaluate a mathematical expression [31]. The "energy" is the number of intermediate results stored in memory and the path through "phase space" is given by the order of evaluation of parenthesized expressions. There are cases of domain-wall motion where the barrier can be efficiently determined. If the structure of the graph on which the polynomial moves is hierarchical (i.e., series-parallel), the barrier can be found in polynomial time [1]. Heuristic methods developed by Mikheev, Drossel, and Kardar have been used to place bounds on the barrier to the motion of a single line in a disordered background [29,35].

Finding a minimum-weight matching on a bipartite graph can be solved in polynomial time, but even *counting* the number of matchings in a given graph is NP-hard [56]. It directly follows that exact computation of the partition function for the heterogeneous problem (say, a lattice with missing bonds) is NP-hard. Note that, as always, Monte Carlo sampling and other heuristic or approximation schemes can give good estimates of free-energy differences at least in the case of pure systems and in smaller disordered systems, so the NP-hardness of a problem does not preclude its study.

In the RFIM, though the ground state can be found in polynomial time, the excited states in principle cannot. That is, finding the spectrum of states is NP-hard, as even finding the highest energy state is equivalent to max-cut [24], which is well known to be NP-hard [64]. However, studying some defined configurations in the RFIM, such as domain walls, can be accomplished in polynomial time solving for ground states with a variety of boundary conditions (fixing spins at the surface).

5.5 Counting States

Many of the outstanding questions in the study of disordered systems arise, naturally enough, from an interest in the thermodynamic limit. One of the basic questions about the large-volume limit asks whether the thermodynamic states are well-defined and, if so, how many states there are. Given a set of values for the parameters, such as the strength of disorder or temperature, the number of states in the large-volume limit is a crucial characteristic of the thermodynamic phase at the given point in parameter space. If there is more than one state, there exist large-scale domain walls (or interfaces [37]) that separate domains associated with distinct states. The set of thermodynamic states and their possible symmetries underlie discussions of the equilibration and macroscopic low-frequency response of the system: dynamic processes can cause phase domains to grow, e.g., by domain-wall motion, and excitations can cause transitions between the local configurations associated with distinct states.

Counting thermodynamic states is quite distinct from counting degenerate lowest-energy configurations in a given finite sample, though the issues can be confused by the similar language used when studying finite samples. This section comments on the existence of degenerate ground-state configurations, especially as it might affect the study of thermodynamic states, and the problem of counting thermodynamic states. The principal result reviewed is the uniqueness, up to global symmetries, of the thermodynamic limit found in numerical work at $T = 0$, for the set of models that can be studied in most detail. This set of models is a subset of those that have been addressed using polynomial-time algorithms in finite dimensions. Even in such relatively solvable cases, it can be difficult to extrapolate from small samples to large systems. This suggests that great care should be taken in interpreting results for the small samples that can be exactly studied for NP-hard optimization problems.

5.5.1 Ground-state Configuration Degeneracy

Solving a single optimization problem gives a lowest energy and, using the simplest algorithms, a single ground-state configuration. When the local disorder values have a small number of values (for example, limiting J_{ij} in a spin glass to take on two values, $\pm J$, or finding shortest paths on a graph with edge weights $w(e) \in \{0, 1, 2, \ldots, m\}$ for m fixed as the path length $L \to \infty$), it is usual for the ground-state configuration to be highly degenerate. It is possible to modify the ground state, say, by flips of individual spins or clusters of spins in a spin glass, or by rearrangement of a portion of a shortest path, without changing the total energy. This local degeneracy results in a large number of ground-state configurations that are connected by flips of finite sets of spins. Such models have an intensive entropy: the number of ground-state configurations grows exponentially with the volume of the sample. More complex algorithms can be used to count exactly (or estimate heuristically) the number of degenerate ground states in a given finite sample [4, 19, 26, 49].

The ground state for a given sample of a given size will be unique for disorder values chosen from a distribution with *sufficiently many* values. Keeping dimensionless ratios of energies fixed (such as Δ in the RFIM), while increasing the range from which random integers are chosen from, the ground-state configuration in a given finite sample approaches a single non-degenerate set of variables, except for global symmetries of the Hamiltonian. The range needed to have a unique ground-state configuration with arbitrarily high confidence increases with the size of the system. The uniqueness of the ground-state configuration is a provably true claim for *continuous disorder distributions* with no δ-function components. This uniqueness can be shown to hold with arbitrarily high probability for a discrete distribution, say uniformly chosen values from $1 \ldots m$, with resolution *m exponential in sample volume N*. This provable uniqueness is easily explained: if two configurations are not related by an exact symmetry, some of the terms in the energy sum will not be the same. For a truly continuous disorder distribution, a coincidence in these sums will occur with probability zero. For a discrete disorder distribution, there are “only” an exponential number of configurations, leading to an arbitrarily small probability for a coincidence in the energy of two distinct configurations, for an exponential refinement of the disorder values.

One of the first technical issues to be considered when implementing combinatorial optimization algorithms is whether to use real or integer values to represent the disorder variables in the problem. It *may* be that using real values does not slow down the computer code, in practice. In any case, arbitrary precision can be attained with sufficiently large integers and the codes that are available from the computer science community often expect integers.

This technical question of which variable type to use to represent the disorder is related to a deeper discussion of the complexity class and the physics of the model. In the treatment of complexity classes for models in statistical physics, it is often (and usually implicitly) assumed that the problem can be accurately described by a string whose length grows only polynomially with system volume. This requirement implies that the typical disorder realization is accurately represented with a precision that grows no faster than exponentially in a power of the volume of the system. Exponential accuracy for the microscopic description gives, with high probability, a unique solution to the optimization problems we consider here, as exponentially small changes in the disorder will not change the ground state. It follows that the assumption about the problem description is not a strong constraint. In fact, one can argue that only $O(\log(N))$ bits, i.e., accuracies polynomial in N, are needed to accurately describe the system.

This more limited accuracy in the problem description is acceptable if the ground-state configuration is not sensitive to changes in the disorder that are bounded by an inverse power of the system size. Scaling arguments support such a claim for many physical problems. Consider, for example, a ground state for a given set of $\{J_{ij}\}$ that describe either a random-bond magnet with anti-periodic boundary conditions or a spin glass. Bray and Moore [7] estimate the size of variation in the disorder that is necessary to change the ground state. Such a discussion usually assumes that the cost to create an excitation (e.g., a region of flipped spins) of size ℓ scales as $\sigma\ell^{\theta}$, where σ^2 is the variance of the (reasonably well-behaved) distribution for J_{ij}. The excitation is bounded by a surface that crosses ℓ^{d_f} bonds, where d_f is a fractal dimension. All energy changes result from changes in interaction energy along this surface, for spin glasses and other random-bond magnets (for the RFIM, one must consider the interior of the excitation in addition). (Even if the *largest* excitation in a system has a different scaling, e.g., a cost that approaches a constant as $L \to \infty$, as in the "TNT" scenario [21, 42] the following argument can be carried through, as long as the *number* of low-energy excitations does not grow exponentially with volume.) For changes $J_{ij} \to J_{ij} + \delta_{ij}$, with δ_{ij} independently chosen random variables of variance δ^2 and mean 0, the surface cost to create this domain wall changes by an amount of order $\delta\sqrt{\ell^{d_f}}$, as this is the order of change in the sum of the J_{ij} over ℓ^{d_f} bonds with independent changes. The cost of a large excitation, positive in the ground state for the original coupling, then could become negative for δ of the order of $\sigma\ell^{1/(\theta-d_f/2)}$. For smaller values of δ, the ground state is stable. This argument can be carried through for the worst case, where the δ_{ij}, while bounded, are chosen specifically to create an excitation [42]. Regardless of the details, the conclusion is that the ground state is stable to perturbations of typical magnitude $\delta < L^{-y}$, for some fixed y. As the errors arising from approximating real numbers by finite-precision integers are well described as changes in the disorder of order $1/k$, with k being a number characterizing the integer precision, the computed ground state will not change with improved precision, for k bounded below by a power of L.

The probability that the ground-state configuration will be unique up to exact symmetries can be very high, in practice, *even for a moderate value of the distribution width* m, at least in finite-dimensional systems with moderate connectivity. For example, in the 3D RFIM near the paramagnetic–ferromagnetic transition, the probability per site of an "accidental" degeneracy is less than 10^{-6}, when the h_i are set by choosing a real number from a Gaussian distribution, with mean zero and unit variance, multiplying by $m = 10^4$, and rounding the real number toward zero to get an integer value. Even when there is a degeneracy, it is associated with single spins or small clusters of spins which can be flipped at no cost in energy, so that the long-scale behavior of the RFIM is not affected. In any case, whenever using integer values to model continuous disorder, such checks for the needed range of disorder values should be carried out.

5.5.2 Thermodynamic State

The notion of thermodynamic state is a basic concept in understanding results from statistical mechanics. The states are used to characterize the bulk behaviors of a system: changes in the parameters characterizing a typical sample lead to phase transitions, which separate qualitatively different sets of states. The structure of the phase space affects the dynamics of the system, as well, as the rigidity of the configuration to long-wavelength perturbations is affected by the symmetries and multiplicities of the thermodynamic state. The behavior of the free-energy landscape as a function of configuration, especially how separate portions of configuration space are separated into distinct regions by large barriers, determines the states, in the large-volume limit.

Though the rigorous study of states can be quite involved [48], even in a simple system, many homogeneous models have obvious ground states, making the thermodynamic state structure straightforward at zero temperature. For example, in a pure Ising magnet, there is a single high-temperature state, the paramagnetic state, where spin-spin correlation functions decay exponentially with separation. At low temperature, there are two states, corresponding to mostly spin-up and mostly spin-down states, with thermal fluctuations about each uniform state. These two states are related by the global spin-flip symmetry of the Hamiltonian H_I.

Many disordered systems are believed to have a transition at some positive temperature T to a low-temperature glassy state or states. It is generally argued that in the lowest temperature state that temperature is an irrelevant variable. The energy fluctuations from the finite temperature are much smaller than the barriers between large-scale rearrangements of the spins and hence temperature is not necessary to describe the long-wavelength behavior. In this case, by studying the zero-temperature states, one can understand many properties of the finite-temperature states in disordered systems.

At zero temperature, a configuration of spins for an infinite-volume system is a ground state if its energy cannot be lowered by flipping a finite number of spins. The ground state can be defined by a limit of the ground-state configurations for a sequence of nested finite systems of increasing size [37, 48]. The couplings in each finite sample are given as a subset of the couplings $\mathcal{J}$ for the infinite-volume sample. Note that the ground-state configuration one finds, when the limiting configuration for an infinite-volume system is well defined, can depend on the choice of boundary condition *and* the exact choice of the subsequence of growing volumes that is chosen. In keeping with the usual discussion, the boundary condition choice for each sample must be independent of $\mathcal{J}$ (there might be "invisible" states [37] that appear for coupling-dependent sequences of boundary conditions).

There are several pictures for the degeneracy of thermodynamic states in disordered materials. Replica-symmetry breaking (RSB) [43] calculations suggest that, at least in infinite dimensions, there are many states which differ by a finite fraction of spins but have only finite energy difference, even in the infinite-volume limit. The droplet picture, developed first by Fisher and Huse for spin glasses, assumes that large-scale excitations (droplets) have typical energies that scale with a power law of excitation diameter ℓ, $\Delta E(\ell) \sim \ell^{\theta}$. When $\theta > 0$, the number of ground states in the infinite-volume limit is given by the global (statistical) symmetries of the Hamiltonian. Variations suggested by Martin, Houdayer, and Krzakala [21] and Palassini and Young [42], suggest that there may be system-size excitations with low-energy, but that small excitations have an energy that scales as in the droplet picture. Many of the consequences of these pictures and variant scenarios for thermodynamic limits and interfaces have been developed in detail by Newman and Stein [37] (also see Ref. [22]).

Numerical results for NP-hard problems such as 3D spin glasses have not been decisive in distinguishing among scenarios. Extensive recent work (e.g., see Ref. [40]) is consistent with aspects of both the RSB scenario and the droplet picture. The largest systems that can be studied exactly have a linear size $L \approx 12$.

Models for which ground states can be found in polynomial time can be studied in more detail [30, 41]. Here it is possible to carry out studies of the infinite-volume limit, for a variety of boundary conditions and a large range of system sizes in a sequence of nested finite samples. This work has been inspired by extensive studies by Newman and Stein [37], who have examined the conditions for the existence of many states. Of particular interest for numerical work is the characterization of the thermodynamic state by the convergence of the configuration in a "window" as the infinite-volume limit is taken. For example, if the configuration or correlation functions approach a constant in any fixed-volume window as $L \to \infty$, the system approaches a single state (up to global symmetries).

5.5.3 Numerical Studies of Zero-temperature States

Implementing this type of investigation of the zero-temperature states is conceptually straightforward. First, we require a method for defining an individual infinite-volume sample, given by an infinite set of couplings $\mathcal{J}$, and a method for generating finite subsamples. Then one solves for the ground state in a nested (finite) sequence of finite-volume subsystems, with sample-independent boundary conditions. Convergence of subsequences of samples to an infinite volume limit is then estimated. One method of doing this is to examine the configuration for each ground-state solution in a common "window" of fixed volume. The portions of the solutions that lie within this common spatial window are compared and examined for convergence to a finite set of subsolutions, in the large-volume limit.

One set of practical challenges is to organize the code, which generates disorder realizations, solves for ground-state configurations for various boundary conditions, and compares the solutions. The complexity of investigating the large-volume limit makes an object-oriented approach to the programming quite useful. One can specify boundary conditions, finite samples, associated ground-state configurations and even the infinite sample itself as types of data. The infinite sample $\mathcal{J}$ can be defined by an integer and a procedure for generating the disorder at a particular location, given the integer. The defining integer, for example, can be used as the seed for a random number generator and the disorder can be generated for the sites and bonds on the lattice, ordered by distance from the origin. The disorder values are generated by starting the random number generator, using the defining integer as a seed, and spiraling out toward infinity, mapping the linear sequence of random values onto the lattice sites and bonds (see Figure 5.3(a)). A constructor method for a finite sample takes the infinite sample and a bounding volume description as parameters. One can then construct finite samples of any size, centered at the origin, with common disorder values in the overlapping regions. The ground-state algorithm then takes a finite sample as a parameter and returns a ground-state configuration. Methods for comparing ground states of different sizes, but centered at a common origin are used to determine whether the ground-state configurations are identical, in a common volume centered at the origin.

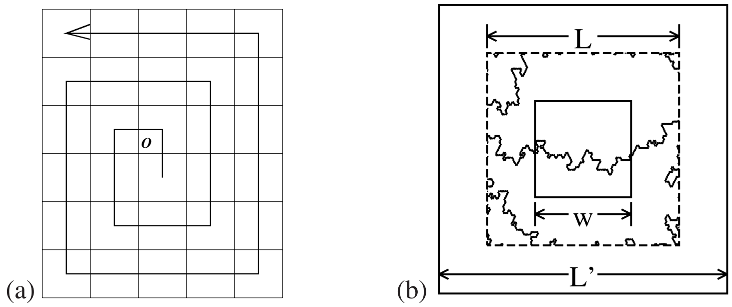


Figure 5.3: (a) Schematic showing how the disorder values are generated for a given sample. Each infinite sample $\mathcal{J}$ is indexed by a seed for a random number generator (there are an uncountable number of samples, but computations are limited to a finite number of samples). The random exchange couplings or fields are generated sequentially along a path spiraling from the origin O out to the desired system size. (b) Depiction of a comparison of the ground states for a nested pair of 2D spin-glass samples, of linear sizes L and L'. The value of the bonds J_{ij} are identical in the common volume of size $L \times L$. The region of size $w \times w$ is the common window where the ground-state configurations are compared. The ground states (not directly shown) have nearly equal numbers of $+$ and $-$ spins that are not apparently correlated. The comparison of the ground states in the common volume is given by lines showing domain walls; these walls cross bonds that are “broken” in one solution, but not the other. As a domain wall crosses the common window, this sample has $D_{\mathcal{J}}(L', L, w) = 1$.

An example of such a comparison in a window of size $w \times w$ for two subsamples of a 2D spin glass, of size $L \times L$ and $L' \times L'$ is shown in Figure 5.3(b) [61]. The interface between the two configurations is shown by the paths within the $L \times L$ box. In the example shown, the configurations are not related by a spatially-independent spin-flip operation within the window. For each sample (realization of disorder), at selected values of w, L, and L', one can compute the indicator function $D_{\mathcal{J}}(L', L, w)$, which is equal to 1 when the configuration in the window changes (as in Figure 5.3(b)) and is 0 otherwise.

The more subtle difficulty in this numerical work is in extrapolating to the infinite-volume limit, based on finite-size samples. This is of course a common problem in numerical work.

One must make sure that finite-size effects are reduced, by seeing how these effects change with system size. It is important to trace these finite-size effects starting from the smallest systems, where they can be clearly identified, and then working up to the largest sizes practicable, where, in the best case, the small volume effects have plainly diminished.

Results for the 2D spin glass, the 2D elastic medium, the 3D elastic medium, and 3D dimer matching are plotted in Figure 5.4. The elastic-medium models describe elastic media with scalar displacements, subject to a pinning potential periodic in the displacements (equivalent to an interface in a $d+1$-dimensional RBIM with periodic bond values along the direction perpendicular to the interface orientation). The dimer-matching model optimizes the sum of weights, randomly assigned to each edge connecting two sites, subject to the constraint that each site (node) belongs to exactly one edge (this model is equivalent to strongly screened vortex lines [60]). The quantity $P(2L, L, w) = \overline{D_{\mathcal{J}}(2L, L, w)}$, with the overline indicating averaging over disorder realizations $\mathcal{J}$, is the probability that the configuration changes (configurations related by global symmetries, such as spin flips in the spin glass, are considered identical) in the central window of w^d sites when the volume with free boundary conditions is expanded from L^d to $(2L)^d$. This quantity is plotted versus L/w, for each of these models. In all cases, the probability decreases toward zero as L/w increases. This strongly suggests convergence to a *single* ground state, up to global symmetries, as the sample volume is increased. The 2D spin glass has also been extensively studied by Palassini and Young [41], with the same conclusion of convergence to a unique ground state.

These results are consistent with a simple scaling argument based on taking the domain walls to have a fractal dimension. This fractal dimension d_f can be measured independently, using standard methods such as twisted boundary conditions [6]. As the domain walls are fractal, $d_f > d-1 \geq d/2$, for $d \geq 2$, the number of *system-size* domain walls that can fit inside an isotropically shaped volume is $O(1)$. The domain walls are not space filling in that they intersect a finite fraction of the bonds, as $d_f < d$, but single domain walls do have a finite chance of intersecting any given volume of size zL^d for a given constant z, so they do wander throughout the sample. (Self-affine interfaces with a transverse width that scales as L^ζ have fractal dimension $d_f = d-1$ whenever $\zeta < 1$, in contrast with the domain walls studied here.) Domain walls are non-intersecting objects; two typical domain walls with $d_f > d/2$ will intersect with constant probability if each is of linear size L, as would any pair of fractal objects with $d_f > d/2$. In addition, if the number of domain walls increased as a power of system size, the domain walls would have to be confined and locally oriented to avoid intersection. Besides geometric constraints, arbitrarily confining a domain wall to a smaller transverse dimension $\ell < L$ typically increases the domain wall's energy whenever $\theta - d + 1 < 0$, as at least $(L/\ell)^{d-1}$ subvolumes of dimension ℓ^d and energy ℓ^d would be needed to span the system, giving a confined domain wall energy that scales faster than $L^{d-1}\ell^{\theta-d+1}$, larger than the original domain wall energy of L^θ [14]. Hence a realization-independent change in boundary conditions will typically create $O(1)$ interfaces that might intersect the central window. These scaling arguments, based on both geometry and energy, are very strong, given the full droplet picture [14] or even the slightly weaker geometric assumption of fractal domain walls and are supported by numerical data that we now discuss.

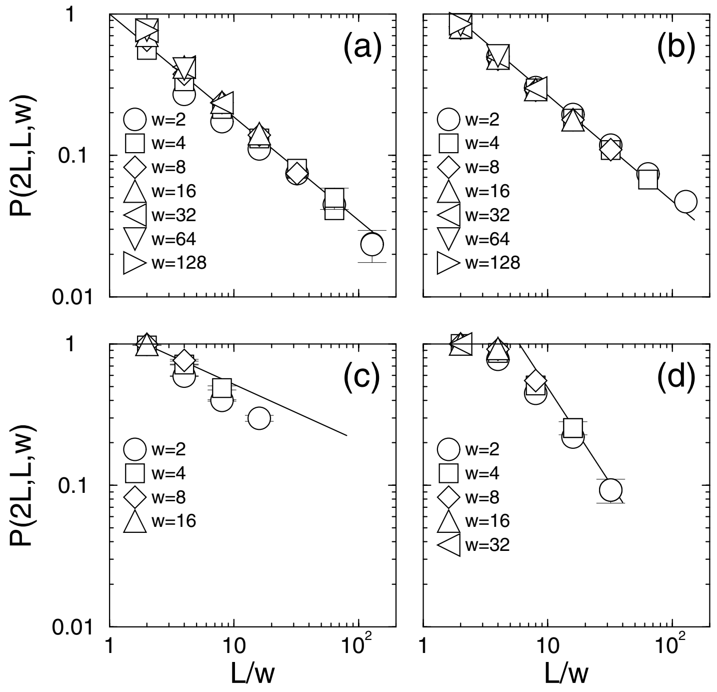


Figure 5.4: Scaling plots for numerical results for the probability $P(2L, L, w) = D_{\mathcal{J}}(2L, L, w)$ that the configuration in the window of size w^d changes upon doubling of the system size. The symbols give plots for four different models that can be solved in polynomial time: (a) the 2D dimer-matching model (equal to the elastic medium in a periodic potential and multiple directed lines in a random potential), (b) the 2D spin glass, (c) the 3D elastic medium in a disordered potential, and (d) the 3D dimer-matching model (equivalent to multiple directed lines in a random potential). The slope of the straight lines is $d_f - d$, where d_f is the independently measured fractal dimension of domain walls in the given model. Scaling suggests that $P(2L, L, w) \sim (w/L)^{d_f - d}$ in the limit $L \to \infty$. Statistical error bars are generally small compared to the size of the symbols, though systematic finite-size corrections are quite apparent, especially for the 3D models.

Given its fractal dimension, the number of volumes of linear size w that an interface intersects is $(L/w)^{d_f}$. As there are $(L/w)^d$ subvolumes of measure w^d, the probability that the interface will intersect the given central volume scales as $\sim (L/w)^{d_f - d}$. The straight lines plotted in Figure 5.4 indicate an exponent of $d_f - d$; the numerical data from P is in accord with this slope for larger L/w. The uniqueness of the *ground state* for typical boundary conditions is a *geometric effect*, not an energetic one. Even though the energy of large-scale excitations may vanish as $L \to \infty$ (e.g., the 2D Ising spin glass) the ground state is unique, due to the domain walls being fractal.

Note that the range of fits is not large in the case of 3D dimer matching. Though the probability of change in window configuration decreases rapidly for $L/w > 10$, in accord with the expected slope, for smaller ratios, the probability is nearly constant and close to unity (i.e., $P(2L, L, w) > 0.5$ for $L \leq 8$). The convergence is also somewhat slower for the 3D elastic medium. This slow convergence to the asymptotic form, which at small scales mimics the results expected for a many-states picture, suggests that in general one needs to be cautious in drawing conclusions from smaller systems, apparently especially so in the case of the two 3D models. It is possible that, as the surface forms a larger fraction of the sample in higher dimensions, the boundary effects are more important, thereby requiring a larger L/w value to approach convergence.

The case of the RFIM requires a slightly different treatment, as the two configurations found with uniform fixed (up or down) boundary spins are not related by a global spin flip. Local fluctuations in the h_i can pin spins to be independent of the boundary condition. In addition, the ground-state structure away from the transition is not expected to be complex, so one should examine the ground-state structure near the transition, to seek any glassy phase with more than two states. It is also near this transition where the domain walls might have a fractal structure, as they do in the models whose results are plotted in Figure 5.4.

One method to address the structure of states for the RFIM is to compute the probability $P_{P,+-}(L, w)$ that the lowest energy configuration with periodic boundary conditions does not coincide with the solutions for *either* up or down fixed boundary spins, in the window of volume w^d. In the ferromagnetic phase, periodic boundary conditions are expected to give the up or the down configuration, so that $P_{P,+-} \to 0$ as $L \to \infty$. Interfaces that cross the entire sample have diverging energy $\sim \sigma L^{d-1}$, with finite surface tension $\sigma(\Delta)$ for $\Delta < \Delta_c$, in the ordered phase, so that when the correlation length is smaller than L, $P_{P,+-}$ is expected to vanish very rapidly with increasing L. The orientation of the ground-state magnetization is *roughly* in correspondence with the sum of the h_i. The sign of this sum has finite probability to change sign when the volume is expanded by a fixed factor. The ground-state magnetization will therefore change sign with finite probability for each expansion of system size by a constant factor: the limit $L \to \infty$ in a given sample includes subsequences that lead to both the up and down states (for numerical confirmation, see [34]). In the paramagnetic phase, the solution in the interior should be independent of boundary condition for $(\Delta - \Delta_c)^{-\nu} \gg L$, as the spins are not correlated over long ranges. Only in some intermediate many-state phase would P_{+-} not decrease rapidly. In the simplest scenario for the states, the correlation length diverges at the transition and the domain walls become fractal. The scaling argument for relative interfaces intersecting the interior window then applies, with $P_{P,+-}$ decreasing as a power law in L. The data, plotted in Figure 5.5 for the 4D RFIM, indicate that $P_{P,+-} \to 0$ as $L \to \infty$, for *all* Δ. This suggests that no more than two states exist at any Δ. The dependence of the peak height $P^{\max}_{P,+-}$ on L near Δ_c is consistent with $P_{P,+-}(L, w, \Delta) \sim L^{d_f - d}$ for a fixed value of w, supporting the simplest scaling scenario.

The results for six models, the 2D spin glass, the 2D elastic medium, the 3D elastic medium, 3D dimer matching, and the 3D and 4D random-field Ising magnet, all indicate convergence, up to global symmetries, to one or two thermodynamic ground states, as the system size is increased. Specifically, the spins in a given volume converge to a fixed configuration as $L/w \to \infty$, for some subsequence of boundary conditions.

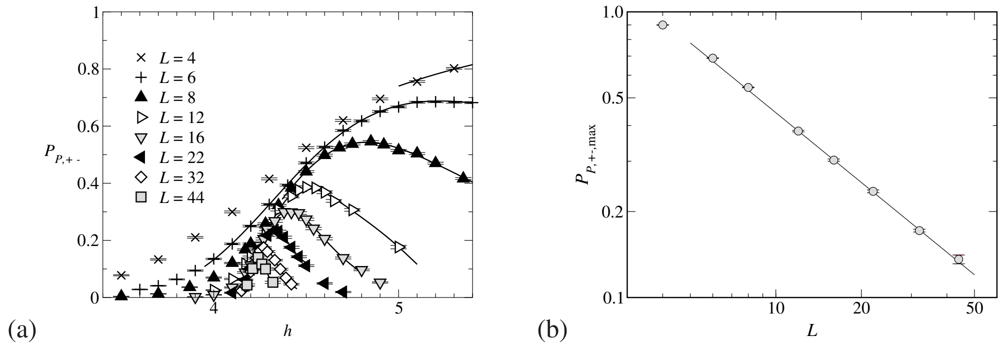


Figure 5.5: (a) Probability $P_{P,+-}$ that periodic boundary conditions give a window configuration distinct from *both* fixed-up and fixed-down boundary conditions for the 4D RFIM, as a function of disorder Δ. The window is of size $w^d = 2^4$. Symbols indicate the linear system size L and the curves are fits to the exponential of a fourth-order polynomial. As L increases, $P_{P,+-}$ decreases at fixed Δ, indicating that at most two states are needed to describe the infinite-volume limit. (b) A plot of the peak value for $P_{P,+-}$ vs. L. The line shows the power-law behavior $P \sim L^{d-d_f}$, where $d_f = 3.20(0.12)$ is the fractal dimension of domain walls in the 4D RFIM.

5.6 Running Times for Optimization Algorithms

While much of the theoretical study of algorithms is based on worst-case complexity analysis, typical or average-case estimates of time complexity are also an important topic in computer science. The typical running time of particular algorithms is certainly of practical interest when the ensemble of problems is well-defined. This is the case in the study of disordered materials, where the notion of a typical problem is naturally and precisely defined. It is very attractive to search for connections between the time and length scales in physical systems and the time to complete numerical computations on models of physical systems.

In physical systems, the characteristic time scale for equilibration can depend on the phase of the system. In a high-temperature phase, the equilibration time is often finite, while it often diverges as a power law of the system size in the low-temperature, broken-symmetry phase of a pure system, where the domains of the low-temperature phase must extend themselves across the system (e.g., see references [15]). A characteristic of continuous transitions in finite-dimensional systems with short-range interactions is the divergence of a *length scale* at the transition [17]. This divergence is connected with the non-analytic behavior of thermodynamic quantities (e.g., the energy density or magnetization) at the transition and leads to a characteristic divergence of time scales at the transition. This *critical slowing down* takes place in physical systems, which have local interactions. It is also quite apparent in simulations of models for phase transitions that update the configuration using local dynamics, such as the Metropolis algorithm at finite temperature. Critical slowing down also affects algorithms with large-scale cluster updates, such as the Wolff or Swendsen-Wang methods [58], though the time scales diverge more slowly. This section addresses critical slowing down observed [39] in algorithms for finding ground states at zero-temperature transitions, especially in the push-relabel algorithm for the RFIM.

5.6.1 Running Times and Evolution of the Heights

To motivate the discussion of the running time of algorithms for the RFIM, it is useful to consider empirical data. Figure 5.6 plots a measure of the sample-averaged running time, and its variance, for the push-relabel algorithm applied to the RFIM in four dimensions, as a function of the disorder parameter Δ, for several values of linear size L. The measure of the running time displayed is r, the number of relabel operations per spin, (a very similar plot results when plotting push operations). For this study, the highest height heuristic, where sites with maximal $d(i) \neq \infty$ are examined for push and relabel operations, was implemented, and global update heuristics were carried out every N relabels. Evident in the plot is the peak in the running time per node near the ferromagnetic–paramagnetic transition.

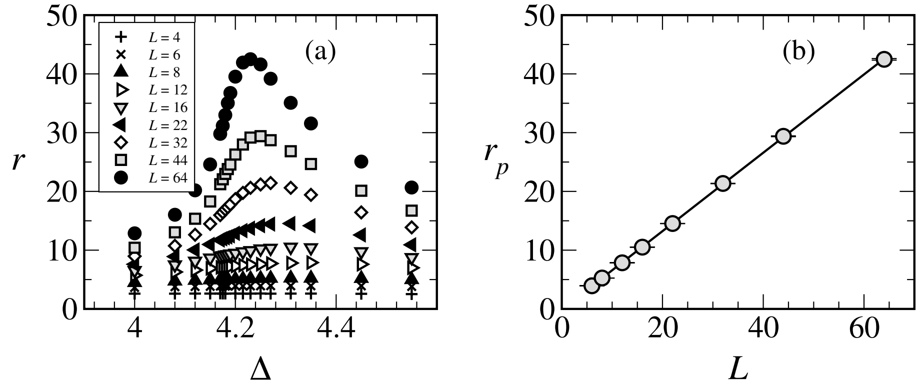


Figure 5.6: Critical slowing down in the push-relabel algorithm for the 4D RFIM. (a) Plot of the number of relabel operations per site in the ground-state computation. The number of operations peaks nearer the ferromagnetic transition $\Delta_c = 4.179(2)$ as L increases. (b) Fitted values of r_p, the peak number of relabels per spin, as a function of linear sample size L. The line is a linear fit to $r_p(L)$.

Not all optimization algorithms will show a sharp change in the running time at a phase transition, while some algorithms will exhibit a *transition* in the running time, though there will not be an actual *peak* in the running time. For example, the shortest-path zero- and finite-temperature transfer-matrix algorithms used to study directed polymer problems run in time linear in the sample size, regardless of correlations in the disorder or temperature [23]. In the closely connected problem of percolation, changes in the solution time can be found. The union-find algorithm, as applied by Newman and Ziff to percolation [38], exhibits a very mild singularity. In this algorithm, bonds are added sequentially, forming a growing set of connected clusters. For a fixed bond density below the percolation threshold, constant work per bond is needed to update the clusters. The number of operations per added bond peaks when the size of the bond set approaches the percolation point. For bond densities at or above percolation, the integrated number of operations to determine clusters scales *almost* linearly in N (the correction is at most the inverse of a variant of the extremely rapidly growing Ackermann function), when a tree structure is used to dynamically organize the bonds or sites into clusters. Also studied in Ref. [38] are more primitive versions of the union-find algorithms whose total running time is bounded by $O(N \log(N))$ steps above the percolation transition. These transitions are rather mild, but can be studied in detail. A physical, though mean-field, problem where an easy–hard transition can be seen is for the spin glass on a Bethe lattice [27].

Returning to the 4D RFIM [33], the empirical curve in Figure 5.6 has several apparent important features. For disorder values above the transition value Δ_c, the number of relabels per spin r appears to converge to a finite value of the order of $r \approx 15$ (for values of Δ greater than shown), though these calculations do not rule out a mild dependence on size. Below the transition, for weak disorder (i.e., the ferromagnetic phase), the running time does not appear to converge to a constant time per spin. The value of r apparently diverges with L *at the transition*. Figure 5.6 plots the fitted peak height in r vs. L. The solid line indicates a simple linear dependence, $R \propto L$. The data is very consistent with this linear dependence.

In addition to averaging running times, insight into the operation of the algorithm can be gained by studying the evolution of the state of the variables used in the algorithm. The evolution of the state of the algorithm can be visualized in several ways; one convenient method is to display the variables $d(i)$ [62]. These distance (or height) variables are non-decreasing during the computation. For a visualization of the evolution of the $d(i)$ auxiliary height variables used in the algorithm in two sample cases for the 1D RFIM, see Figures 5.7(a,b). (Note, though, that the 1D RFIM, which is always paramagnetic, can be solved directly using a simpler and faster algorithm [51], which is linear in L.) These plots have algorithm time running vertically, from top to bottom, and the position i is the horizontal coordinate. Each row of $\{d(i)\}$ is recorded immediately after a global update, which is carried out every $L/20$ relabels. The gray scale intensity corresponds to the value of $d(i)$ at each time. The pixel at global update g and position i is colored white if $d(i) = \infty$, i.e., the spin has been definitively assigned the value $s_i = 1$) or if $d(i) = 0$, that is, the spin has negative excess. Other gray levels indicate the value of $d(i)$ relative to its maximum (non-infinite) value: the pixel (g, i) is colored black at the highest non-L value of $d(i)$, $d_{\max}(g)$ and is lightly colored at the lowest values. In the 1D RFIM, the typical size of the domains with aligned spins scales as $\xi \sim \Delta^{-2}$ [51]. In Figure 5.7(a), where Δ is small enough that $\xi \gg L$, the ground state has all spins negative and the final field of $d(i)$ has a simple sawtooth shape, linear between peaks and valleys, resulting from the coalescence of multiple sawtooths. The evolution of the $L = 2500$ spins is traced over the $g_{\max} = 85$ global updates needed to find the ground state. Part (b) of the figure shows the evolution of the $d(i)$ for a case with $\xi < L$. Here, $g_{\max} = 56$. The final state of the $d(i)$ shows the "slopes" that would funnel away any small amount of excess $\delta e(i)$, if the h_i were increased at some location. Part (c) of Figure 5.7 shows the evolution for two similar cases, $\xi \gg L$ and $\xi \approx L$, in two dimensions, with the $\{d(i)\}$ displayed after selected global updates. Though the assignment of the spin values $\{s_i\}$ is unique, the final configuration of the $d(i)$ in any dimension is not unique and depends on the order in which the sites are relabeled and the flows are rearranged by push operation. The timing of the algorithm also depends on the choices of ordering of the push and relabel operations. There are general features of the timing, however, which are connected to the length scale ξ and the phase of the RFIM.

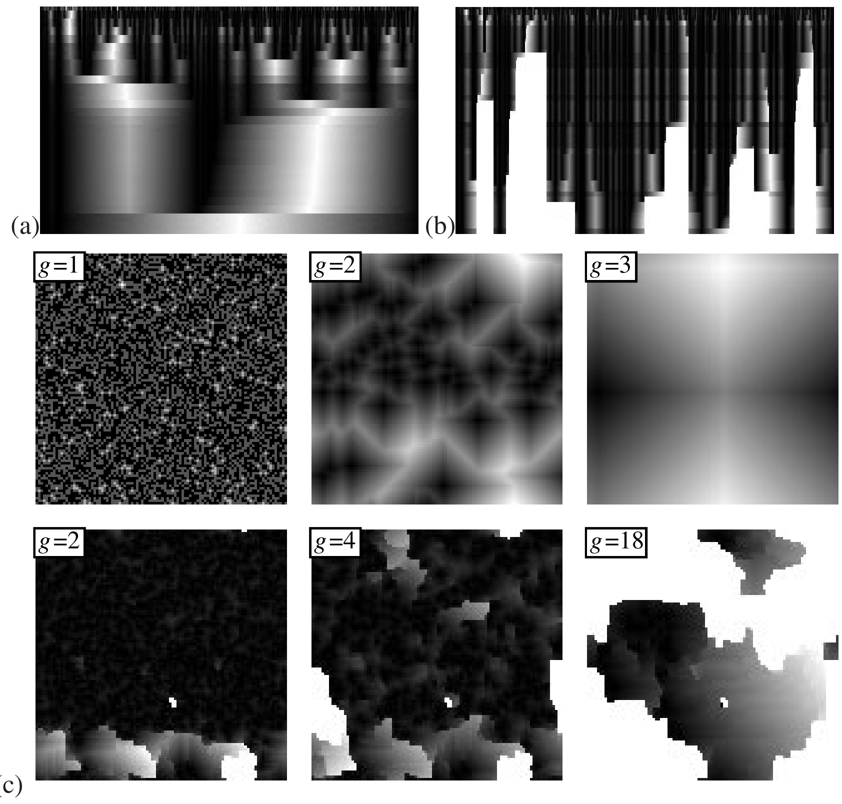


Figure 5.7: Gray-scale plots of the coarsening of the height variables during the push-relabel algorithm. Lighter colored regions correspond to greater height; contiguous white regions indicate regions of up-spin (sites at height $d(i) = \infty$). (a) Evolution of heights, with algorithm time progressing vertically down, for a 1-d RFIM sample with small Δ, where the $\xi \gg L = 2500$. Each row displays the height as a function of position (horizontal) after a global update: the heights are normalized at each time (each row) so that the highest height is white. The formation of a single (down-spin) domain takes place by the coalescence of smaller regions. Global updates are performed every $L/20 = 125$ relabels. (b) A plot similar to (a), with larger Δ and $\xi < L$. The formation of spin-up (white region) and spin-down regions (gray regions) is shown. (c) Sequences of plots for dimension $d = 2$ RFIM samples, with $\xi \gg L$ (upper row) and $\xi \approx L$ (lower row). The labels indicate the number of global updates that have been performed, where global updates are executed after every L^2 relabels.

5.6.2 Heuristic Derivation of Running Times

The scaling of the running time can be heuristically explained, using the degeneracy of the ground states, the characteristics of the push-relabel algorithm, and the divergence of the correlation length near the transition. This explanation relies upon exploring how the excess $e(i)$

is rearranged by the dynamics of the algorithm. The main assumption is that the rearrangement is especially efficient: the landscape of the $d(i)$ that evolves has nearly constant gradients, i.e., the $d(i)$ are linear functions of position. This is consistent, for example, with the form of the peaks seen in Figure 5.7. The downhill paths, the path from any node with positive excess to a node with negative excess, are taken to be non-fractal. This means that the paths might be rough, in that they deviate from a straight line, but the *path distance* from a node i with positive height to a sink is taken to scale as the *spatial distance*. The excess that is rearranged by push operations moves downhill, nearly linearly toward the sinks, as long as this linear path property is maintained. The linear paths can be argued to be maintained by the breadth-first search carried out by global updates, similar to the nearly linear paths seen in shortest-path algorithms [11]. A bound can then be placed on the number of relabel operations: the maximum height, created by relabels *or* global updates, (leaving out the sites with $d(i) = \infty$) is bounded by a multiple of the linear dimension of the region. As the largest regions of connected spins [63] have size ξ, for Δ near Δ_c and larger, the number of relabel operations per site scales as the average height, ξ, over the region, which is, in turn, given by ξ.

In the ferromagnetic phase, the overall spin direction is determined, roughly (i.e., neglecting minority spins), by the sign of the sum of the h_i. In the limit $\Delta \ll 1$, this correspondence is exact. The ground state is simply determined by a global sum of the external fields and could be carried out directly in linear time in this limit. The push-relabel algorithm rearranges excess $e(i)$ *locally*, however. At any given algorithmic time, the sum is determined at some scale $\ell(t)$. The summation is effectively carried out hierarchically. The combination of local rearrangement and linear landscapes leads to a logarithmic dependence of operation count on system size, similar to the efficient parallel summation of a large set of numbers [9].

A check of this scaling generally confirms these heuristic arguments. Most robust is the linear scaling of the solution time per spin *at* the phase transition, as seen in Figure 5.6. This scaling is consistent with taking the average work per spin to scale with the linear size L of the system (equal to the correlation length ξ near the transition). When $d = 1$, there is no phase transition, but the divergence of the correlation length is known exactly: $\xi \sim \Delta^{-2}$ (see, e.g., Ref. [51]). Data for the 1D RFIM, with no global relabeling, shows a peak in the running time at a disorder value consistent with $\Delta_{\text{peak}} \sim L^{-1/2}$ [32], and $r_p \sim L$.

5.7 Further Directions

This chapter has focused on optimization problems with known polynomial-time algorithms. The thermodynamic limit and asymptotic running times can be carefully studied. Nonetheless, the convergence to the thermodynamic limit may become apparent only for relatively large systems and care must be taken in interpreting results from small samples. While problems in P apparently have all of the properties of disordered systems with glassy behavior, there are problems of interest where finding the ground states is NP-hard. One active area of research is aimed at finding heuristic solutions to such problems, especially approaches that are efficient for typical cases (see other chapters in this volume). Progress in this area will allow for a more definitive study of the large-volume limit of, for example, spin glasses in finite dimensions.

The relationship between phase transitions and algorithms is also of much interest, having started with and continuing to emphasize mean-field models, where there are some solid theoretical results. These models resemble prototypical problems from computational science. The finite-dimensional models emphasized in this chapter are important from the physical point of view and also include the importance of finite correlation lengths, which are not well-defined in mean-field models. The connections and differences among this broad variety of models deserves further attention. Of particular interest would be bounds or arguments for the typical running time that would be independent of the algorithm, if this is even possible.

Clearly, the uniqueness of the ground state and the critical slowing down of algorithms near phase transition have implications for parallel simulations of these systems. For example, a random-field Ising magnet in the ferromagnetic phase could be divided into overlapping subsamples and ground states for the subsamples could be determined using both uniform up-spin and uniform down-spin boundary conditions. The up-spin configurations could be merged among themselves and the down-spin configurations could be merged among themselves and the two global configurations would then be compared to determine the true ground state. This suggests that the RFIM could be solved in time linear in the volume, for weak disorder, sufficiently far from the transition ($\xi \ll \ell$, where ℓ is the linear size of the subsamples). For stronger disorders, again with $\xi \ll \ell$, the solutions to be merged would be independent of the boundary condition. Large rare clusters of minority spins, due to strong pinning regions, will exist in the ferromagnetic phase and large clusters of weakly pinned spins will exist in the paramagnetic phase, however. In each of these cases, the merged solution will be in error. The probability of such failures can be made arbitrarily small by choosing larger initial subsamples. This implies that a very robust nearly linear time algorithm can be constructed for the RFIM (and similar problems) away from the critical disorder.

I would like to thank my collaborators who have contributed to my understanding, especially Daniel Fisher, David McNamara, Jennifer Schwarz, and Chen Zeng. I have enjoyed very useful discussions of the RFIM with Eira Seppäla, Mikko Alava, Jon Machta, and Yonathan Shapir. I would like to thank Jan Meinke for critical feedback on this manuscript. This work was supported in part by grants from the National Science Foundation (DMR-0109164 and ITR DMR-0219292). I would like to thank the Kavli Institute of Theoretical Physics for their hospitality (NSF PHY-9907949).

References

[1] H. M. Abdel-Wahab and T. Kameda, *Scheduling to Minimize Maximum Cumulative Cost Subject to Series-Parallel Precedence Constraints*, Op. Res. **26**, 214 (1978).

[2] M. J. Alava, P. M. Duxbury, C. Moukarzel, and H. Rieger, *Exact Combinatorial Algorithms: Ground States of Disordered Systems*, in *Phase Transitions and Critical Phenomena, Vol. 18*, C. Domb and J. L. Lebowitz, eds., (Academic Press, San Diego, 2001).

[3] F. Barahona, *On the Computational Complexity of Ising Spin Glass Models*, J. Phys. A **15**, 3241 (1982).

[4] S. Bastea and P. M. Duxbury, *Ground State Structure of Random Magnets*, Phys. Rev. E **58**, 4261-4265 (1998), cond-mat/9801108.

[5] L. Berthier and J.-P. Bouchaud, *Geometrical Aspects of Aging and Rejuvenation in an Ising Spin Glass: A Numerical Study*, Phys. Rev. B **66**, 054404 (2002).

[6] A. J. Bray and M. A. Moore, *Lower Critical Dimension of Ising Spin Glasses: A Numerical Study*, J. Phys. C **17**, L463 (1984).

[7] A. J. Bray and M. A. Moore, *Chaotic Nature of the Spin-Glass Phase*, Phys. Rev. Lett. **58**, 57 (1987).

[8] M. Cieplak, A. Maritan, M. R. Swift, A. Bhattacharya, A. L. Stella, and J. R. Banavar, *Optimal Paths and Universality*, J. Phys. A **28**, 5693 (1995).

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, Mass., 1990).

[10] E. De Santis, *Swendsen-Wang Dynamics on Z^d for Disordered Non-ferromagnetic Systems*, arxiv.org/abs/math/0206276.

[11] R. Dobrin and P. Duxbury, *Minimum Spanning Trees on Random Networks*, cond-mat/0101340.

[12] P. M. Duxbury and J. H. Meinke, *Ground State Non-universality in the Random Field Ising Model*, Phys. Rev. E **64**, 036112 (2001).

[13] D. S. Fisher, *Scaling and Critical Slowing Down in Random-field Ising Systems*, Phys. Rev. Lett. **56**, 416 (1986).

[14] D. S. Fisher and D. A. Huse, *Equilibrium Behavior of the Spin-glass Ordered Phase*, Phys. Rev. B **38**, 386 (1988).

[15] For a review, see, e.g., H. Furukawa, *A Dynamic Scaling Assumption for Phase Separation*, Adv. Phys. **34**, 703 (1986).

[16] A. Galluccio, M. Loebl, and J. Vondrak, *New Algorithm for the Ising Problem: Partition Function for Finite Lattice Graphs*, Phys. Rev. Lett. **84**, 5924 (2000).

[17] For a review, see, e.g., N. Goldenfeld, *Lectures on Phase Transitions and the Renormalization Group* (Addison-Wesley, Reading, Mass., 1992).

[18] A. V. Goldberg and R. E. Tarjan, *A New Approach to the Maximum Flow Problem*, J. Assoc. Comput. Mach. **35**, 921 (1988).

[19] A. K. Hartmann, *Ground-state Landscape of 2d $\pm J'$ Ising Spin Glasses*, Eur. Phys. J. B **8**, 619 (1999); *Ground State Structure of Diluted Antiferromagnets and Random Field Systems*, Phys. A **248**, 1 (1998); *Ground-state Clusters of Two-, Three- and Four-dimensional $\pm J$ Ising Spin Glasses*, Phys. Rev. E **63**, 016106 (2001).

[20] A. K. Hartmann and H. Rieger, *Optimization Problems in Physics* (Wiley-VCH, Berlin, 2002).

[21] J. Houdayer and O. C. Martin, *Droplet Phenomenology and Mean Field in a Frustrated Disordered System*, Phys. Rev. Lett. **81**, 2554-2557. F. Krzakala and O.C. Martin, *Spin and Link Overlaps in Three-Dimensional Spin Glasses*, Phys. Rev. Lett. **85**, 3013 (2000).

[22] D. A. Huse and D. S. Fisher, *Pure States in Spin Glasses*, J. Phys. A **20**, L997 (1987); *Absence of many States in Realistic Spin Glasses*, J. Phys. A **20**, L1005 (1987).

[23] D. A. Huse and C. L. Henley, *Pinning and Roughening of Domain Walls in Ising Systems Due to Random Impurities*, Phys. Rev. Lett. **54**, 2708 (1985).

[24] S. Istrail, *Statistical Mechanics, Three-dimensionality and NP-completeness*, *ACM Symposium on Theory of Computing*, (ACM, New York, NY, 2000).

[25] S. Kirkpatrick, C.D. Gelatt Jr., M. P. Vecchi, *Optimization by Simulated Annealing*, Science **220**, 671 (1983).

[26] J. W. Landry and S. N. Coppersmith, *Dynamics of a Complex Quantum Magnet*, arxiv.org/cond-mat/0301251 (2003).

[27] F. Liers, M. Palassini, A.K. Hartmann, and M. Jünger, *Ground State of the Bethe Lattice Spin Glass and Running Time of an Exact Optimization Algorithm*, Phys. Rev. B 68, 094406 (2003).

[28] O. C. Martin, R. Monasson, and R. Zecchina, *Statistical Mechanics Methods and Phase Transitions in Optimization Problems*, Th. Comp. Sci. **265**, 3 (2001).

[29] D. McNamara, Ph.D. Thesis, Syracuse University (2000).

[30] A. A. Middleton, *Numerical Investigation of the Thermodynamic Limit for Ground States in Models with Quenched Disorder*, Phys. Rev. Lett. 83, 1672 (1999).

[31] A. A. Middleton, *Computational Complexity of Determining the Barriers to Interface Motion in Random Systems*, Phys. Rev. E **59**, 2571 (1999).

[32] A. A. Middleton, *Critical Slowing Down in Polynomial Time Algorithms*, Phys. Rev. Lett. 88, 017202 (2002).

[33] A. A. Middleton, *Scaling, Domains, and States in the Four-dimensional Random Field Ising Magnet*, http://arXiv.org/cond-mat/0208182.

[34] A. A. Middleton and D. S. Fisher, *The Three-dimensional Random Field Ising Magnet: Interfaces, Scaling, and the Nature of States*, Phys. Rev. B, **65**, 134411 (2002).

[35] L. V. Mikheev, B. Drossel, and M. Kardar, *Energy Barriers to Motion of Flux Lines in Random Media*, Phys. Rev. Lett. **75**, 1170 (1995).

[36] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Determining Computational Complexity from Characteristic Phase Transitions*, Nature **400**, 133 (1999).

[37] C. M. Newman and D. L. Stein, *Ordering and Broken Symmetry in Short-ranged Spin Glasses*, arxiv.org/cond-mat/0301403; *Finite-Dimensional Spin Glasses: States, Excitations, and Interfaces*, arxiv.org/cond-mat/0301022.

[38] M. E. J. Newman and R. M. Ziff, *Fast Monte Carlo Algorithm for Site or Bond Percolation*, Phys. Rev. E **64**, 016706 (2001).

[39] A. T. Ogielski, *Integer Optimization and Zero-temperature Fixed Point in Ising Random-field Systems*, Phys. Rev. Lett. **57**, 1251 (1986).

[40] M. Palassini, F. Liers, M. Juenger, and A. P. Young, *Low Energy Excitations in Spin Glasses from Exact Ground States*, arxiv.org/cond-mat/0212551.

[41] M. Palassini and A. P. Young, *Triviality of the Ground State Structure in Ising Spin Glasses*, Phys. Rev. Lett. **83**, 5129 (1999). M. Palassini and A. P. Young, *Trivial Ground State Structure in the Two-Dimensional Ising Spin Glass*, Phys. Rev. B **60**, R9919 (1999).

[42] M. Palassini and A. P. Young, *Nature of the Spin Glass State*, Phys. Rev. Lett. **85**, 3017 (2000).

[43] M. Mezard, G. Parisi and M. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).

[44] J.-C. Picard and H. D. Ratliff, *Minimum Cuts and Related Problems*, Networks **4**, 357 (1975).

[45] H. Rieger, *Ground State Properties of Fluxlines in a Disordered Environment*, Phys. Rev. Lett. **81**, 4488 (1998).

[46] H. Rieger and U. Blasum, *Ground-state Properties of Solid-on-solid Models with Disordered Substrates*, Phys. Rev. B **55**, R7394 (1997).

[47] H. Rieger, L. Santen, U. Blasum, M. Diehl, M. Jünger, *The Critical Exponents of the Two-dimensional Ising Spin Glass Revisited: Ground State Calculations and Monte Carlo Simulations*, J. Phys. A **29**, 3939 (1996).

[48] D. Ruelle, *Statistical Mechanics: Rigorous Results* (W. A. Benjamin, Reading, MA, 1967).

[49] L. Saul and M. Kardar, *Exact Integer Algorithm for the Two-dimensional $\pm J$ Ising Spin Glass*, Phys. Rev. E **48**, R3221 (1993).

[50] R. Schorr and H. Rieger, *Universal Properties of Shortest Paths in Isotropically Correlated Random Potentials*, Eur. Phys. J. B **33**, 347 (2003).

[51] G. Schroöder, T. Knetter, M. J. Alava and H. Rieger, *Ground States Versus Low-temperature Equilibria in Random Field Ising Chains*, Eur. Phys. J. B **24**, 101 (2001).

[52] N. Schwartz, A. L. Nazaryev, and S. Havlin, *Optimal Path in Two and Three Dimensions*, Phys. Rev. E **58**, 7642 (1998).

[53] E. Seppäla, M.S. Thesis, Helsinki University of Technology (unpublished).

[54] R. Sethi, *Complete Register Allocation Problems*, SIAM J. Comput. **4**, 226 (1975).

[55] J. D. Shore, M. Holzer, and J. P. Sethna, *Logarithmically Slow Domain Growth in Non-randomly Frustrated Systems: Ising Models with Competing Interactions*, Phys. Rev. B **46**, 11376 (1992).

[56] L. G. Valiant, *The Complexity of Computing the Permanent*, Theoretical Comp. Sci. **8**, 189 (1979).

[57] J. Villain, *Nonequilibrium Critical Exponents in the Random-field Ising Model*, Phys. Rev. Lett. **52**, 1543 (1984).

[58] R. H. Swendsen and J.-S. Wang, *Nonuniversal Critical Dynamics in Monte Carlo Simulations*, Phys. Rev. Lett. **58**, 86 (1987).

[59] For surveys, see *Spin Glasses and Random Fields*, A. P. Young, ed. (World Scientific, Singapore, 1998).

[60] C. Zeng, A. A. Middleton, and Y. Shapir, *Ground-State Roughness of the Disordered Substrate and Flux Line in d=2*, Phys. Rev. Lett. **77**, 3204 (1996).

[61] For this diagram and other results published by the author, each sample is directly mapped to an instance of a non-bipartite weighted matching, using the method of Barahone [3]. The Blossom IV algortihm developed by Cook and Rohe [W. Cook and A. Rohe, INFORMS J. Comp. **11**, 138 (1999)] was then used to find the matching and, subsequently, the ground-state configuration.

[62] For applets,
`http://physics.syr.edu/research/condensedmatter/RFIM/`.

[63] In the 3D with $\Delta > \Delta_c$ and in the 4D RFIM even for Δ somewhat smaller than Δ_c, this picture is complicated by the simultaneous existence of spanning clusters of both up and down spins. The regions should be thought of then as the correlated domains, not as simply up and down spin regions.

[64] For exact definitions and extensive discussion of complexity classes, see, e.g., C. H. Papadimitriou, *Computational Complexity* Addison-Wesley, Reading, MA, 1994). Determining exact answers for NP-hard problems, using the best known algorithms, appears to require computational times exponential in the size of the problem description in the worst case. Problems in the class of problems P can can be solved in time polynomial in the problem size.

6 Computing the Potts Free Energy and Submodular Functions

Jean-Christian Anglès d'Auriac

6.1 Introduction

A growing part of statistical mechanics relies now on the use of simulation and computers. New exact results are extremely scarce, and it appears that almost all exact results attainable with known methods have been found. This is particularly true with disordered models. In this context numerical methods become very important.

Various generalizations of the Ising model have been introduced. Two types of reasoning motivated these generalizations – some were intended to describe more realistic situations, while the merit of the others was solvability. Exact solutions for disordered models exist mainly for one-dimensional models, or for infinite dimensional models (mean-field models). In two and three dimensions various approximate solutions exist, most of them relying on heavy numerical calculations. A variety of numerical methods have been designed to study these models. In particular, low-temperature behavior leads to combinatorial optimization problems, since in this limit only the states of minimum energy contribute. It is worthwhile noticing at this point that the exchanges between numerical statistical mechanics and combinatorial optimization have been useful in both directions: several improvements in the simulated annealing method are now in common use in optimization on one hand, while, conversely, properties of the ground states of the disordered Ising model have been elucidated using optimization methods.

The problem we address in this chapter is not a low-temperature problem. On the contrary, we will compute the free energy of a Potts model at *any temperature*, including some phase transition temperatures. To transform the problem of computing the free energy into an optimization problem (i.e. to find a minimum in a finite set), we need to take some limit. Usually this is a zero-temperature limit. Here this will be the limit of an *infinite number of states*. At first sight this limit seems quite academic, since no experimental situation at this time is described by the infinite state Potts model. However, this model is still of importance since it is very likely that it belongs to the same universality class as the Random Transverse Quantum Ising chain, a model which has a very interesting behavior under renormalization.

This chapter is organized as follows: in the first part the Potts model is introduced. In the second part the submodular functions are introduced. Only a very few results and examples are presented: from this very active field of discrete mathematics, only the strict minimum needed to follow the algorithm of the last section is covered. The references included are far from being exhaustive. In the third part the connection between submodular function and the

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

free energy of the Potts model in the limit of an infinite number of states is given. This part presents the Optimal Cooperation algorithm and its proof in detail. The final section discusses the practical implementation of the algorithm and its evaluation. The empirical complexity is evaluated, and examples are given.

This chapter is completely self-contained and its purpose is two-fold. First, we present a new method to numerically compute the free energy of the Potts model in the limit of an infinite number of states, with enough details to allow anyone to use it. Second, we try to draw the attention of physicists to the powerful methods of submodular function theory. It is indeed very likely that there will be other applications of this theory in lattice statistical physics.

6.2 The Potts Model

The standard scalar Potts model has been introduced as a generalization of the Ising model [14]. The purpose of introducing this model was to describe more realistically new physical situations. However, it was soon realized that this model is very interesting also on its own, from the point of view of integrability. There is a huge literature concerning the Potts model. The reference [17] is not very recent but is an excellent entry point to this model.

6.2.1 Definition of the Potts Model

The Potts model is a model on a lattice. The dynamical variables (spin variables) σ_i are discrete variables living on the vertices of the lattice, taking one of q discrete values ($\sigma_i \in Z_q$). The parameter q is the number of states. To each edge ij, with extremities i and j, a coupling constant J_{ij} is associated. The Hamiltonian reads:

$$H = -\sum_{<ij>} J_{ij}\delta(\sigma_i - \sigma_j) \tag{6.1}$$

where the summation runs over all the edges ij, and $\delta(x)$ is the Kronecker function ($\delta(x) = 1$ if $x = 0$ and zero otherwise).

Most attention has been devoted to regular lattices, like, for example, the one-dimensional chain, the two-dimensional square, triangular or hexagonal lattices, or the three-dimensional cubic lattice. When all the couplings J_{ij} have the same common value J, the model is a pure model, and if $J > 0$ it is said to be a ferromagnetic model. The Potts model covers a variety of situations depending of the number of states, the type of lattice, and the kind of coupling constants.

The first quantity one needs to compute is the free energy F, from which much information can be extracted, taking the various partial derivatives. It is actually a major challenge to compute the free energy of the Potts model in the most general possible case. The free energy is related to the partition function $Z(\beta) = \exp(-\beta F)$ and the partition function is

$$Z(\beta) = \sum_{\{\sigma\}} \exp\bigl(-\beta H(\{\sigma\})\bigr) \, . \tag{6.2}$$

The sum runs over the q^N possible states of the N spins and H is a function of the spins given by (6.1). The parameter β is the inverse temperature. The dependence of Z on the coupling constants J_{ij} and on the number of states q is not explicitly stated. Of particular interest are the values where the partition function presents a singularity. Singularities in the partition function can occur only when the lattice size or the number of states goes to infinity.

6.2.2 Some Results for Non-random Models

Let us briefly summarize some well known rigorous results concerning the non-random model: for the one-dimensional chain, the free energy can be calculated for any value of the number of states q and for any inverse temperature β: there is no singularity in Z at finite temperature. The partition function can also be computed on complete graphs, in which every site is connected to every other site. One gets in this case a mean field description of the Potts model. For the case of the two-dimensional models it has been shown that there is a high-temperature disordered phase separated from a low-temperature ordered phase by a transition point, where a non-analyticity in the free energy occurs. The partition function has been calculated but only at the transition point, except for $q = 2$ (the Ising model) where it can by calculated at any temperature. At the transition point the model has an enhanced symmetry and possesses the property of integrability. When the number of states is $q \leq 4$ then the transition is second order, while when $q > 4$ it is first order, with a more and more pronounced first-order character (a smaller correlation length and a larger latent heat) when q increases.

6.2.3 The Ferromagnetic Random Bond Potts Model

The Ferromagnetic Random Bond Potts Model (RBPM) corresponds to the case where the coupling constants J_{ij} are positive random variables. The positivity of the J_{ij} ensures that the ground states of the model (i.e. the spin states of lower energy H) are the q states where all the spins have the same value. For random models it is necessary to compute quantities averaged over the probability distribution of the couplings J_{ij}. The free energy is then

$$f(\beta) = \int dJ_1 \cdots \int dJ_m P(J_1, \cdots, J_m) f_{J_1 \cdots J_m}(\beta) \tag{6.3}$$

such an average is a so-called quenched average. The task of computing the average free energy is extraordinarily difficult, and has been rigorously achieved only in peculiar cases (for example the Nishimori line). In this chapter we show how to compute exactly $f_{J_1 \cdots J_m}(\beta)$ for a particular choice of the couplings $J_1 \cdots J_m$, in the q infinite limit. The average (6.3) is performed by sampling the probability distribution P. However, if the number of possible configurations of the couplings $J_1 \cdots J_m$ is finite and small enough, it is possible to enumerate all of them.

6.2.4 High Temperature Development

Let us now present an expansion of the partition function in inverse temperature β which is useful both to compute Z and also to *define* a continuation of the Potts model for non integer values of the number of states q. Let us regard the lattice as a graph G: the sites and the bonds of the lattice are the vertices and the edges of the graph. To evaluate the partition function

(6.2) one linearizes the exponential $\exp(a\delta) = 1 + (\exp(a) - 1)\delta$ which is valid when δ is restricted to the two values $\delta = 0$ or $\delta = 1$. Introducing

$$v_{ij} = \exp(\beta J_{ij}) - 1 \tag{6.4}$$

one gets

$$\begin{aligned} Z &= \sum_{\{\sigma\}} \exp\left(\sum_{ij} -\beta J_{ij}\delta(\sigma_i - \sigma_j) \right) \\ &= \sum_{\{\sigma\}} \prod_{ij} \exp\bigl(-\beta J_{ij}\delta(\sigma_i - \sigma_j)\bigr) \\ &= \sum_{\{\sigma\}} \prod_{ij} \bigl(1 + v_{ij}\delta(\sigma_i - \sigma_j)\bigr) . \end{aligned}$$

A careful book-keeping of the terms in the development of the above expression leads to:

$$Z = \sum_{G' \subseteq G} q^{c(G')} \prod_{e \in G'} v_e \tag{6.5}$$

where G' denotes any subgraph of G, i.e. a graph, possibly not connected (but all vertices are kept), where some edges of G have been deleted (there are 2^m subgraphs where m is the number of edges of G). $c(G')$ is the number of connected components of the subgraph G'. For example, for the empty subgraph $G' = \emptyset$, the number of connected components is the number of sites, while for $G' = G$ it is one. The product in (6.5) is over all the edges in G' with the convention that the product over an empty set is one. If the parameter β is small (i.e. high temperature) then the parameters v_{ij} are small and, summing in (6.5), only the subgraphs with few edges provide an approximation to the partition function: this is a high-temperature development. Note also the way the parameter q appears in (6.5): it can be extended to non-integer values, relating the Potts model to other problems (percolation, etc.) [13].

6.2.5 Limit of an Infinite Number of States

Let us now follow [12] and introduce another parameterization of the couplings with new variables w_e defined by

$$v_e = q^{w_e} .$$

Inserting this expression in (6.5) one gets $Z = \sum_{G' \subseteq G} q^{c(G') + \sum_{e \in G'} w_e}$, and defining $f(G) = c(G) + \sum_{e \in G} w_e$:

$$Z = \sum_{G' \subseteq G} q^{f(G')} .$$

If now q goes to infinity only the subgraphs $G^\star$ maximizing $f(G)$ will contribute, and computing the partition function of the Potts model in the infinite number of states limit amounts to finding the subgraphs of the graph G maximizing the function f, i.e. minimizing the function:

$$f_P(G) = -\left(c(G) + \sum_{e \in G} w_e \right) . \tag{6.6}$$

6.3 Basics on the Minimization of Submodular Functions

It turns out that the function (6.6) has a property which allow us to minimize it very efficiently. This paragraph explains this property. Reference [15] contains all the concepts used in this section.

6.3.1 Definition of Submodular Functions

We start by recalling elementary results for submodular functions. The functions we consider here are not the usual functions of some continuous variables. Instead these functions are *set functions:*

Definition. Let V be a finite set and $2^V = \{X \mid X \subseteq V\}$ be the set of all the subsets of V. A function $f : 2^V \to \mathbb{R}$ is called a set function.

Set functions act on a finite set. So in principle it is possible to find the extrema of this kind of function by inspection. However, even if it is finite, the cardinality of the set $2^{|V|}$ is so huge that the inspection of all the possible values is impossible. The task of finding the minimum of a set function becomes possible if this set function is submodular:

Definition. A set function f is submodular if for all subsets $A \subseteq V$ and $B \subseteq V$:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \tag{6.7}$$

From the definition, the notion of submodularity applies to real-valued functions acting on the set of the subsets of a given ensemble. Note that the notion of submodularity can be generalized to functions acting on any finite lattice[1]. Note also that we will use here only rational-valued functions.

6.3.2 A Simple Characterization

It is simple to show that a function f is submodular if and only if for any subsets $S \subseteq R \subseteq V$ and for any $x \in V$:

$$f(S \cup \{x\}) - f(S) \geq f(R \cup \{x\}) - f(R) \tag{6.8}$$

This means intuitively that adding an element to a “small” ensemble S (since $S \subseteq R$) has more effect than adding to a “large” ensemble R.

6.3.3 Examples

Submodular functions arise in many fields. Let us give a few examples:

A first example of a submodular function is given by the following function c. Take a finite set V and give arbitrary labels to each element $x \in V$. The function c associates to every subset $U \subseteq V$ the *number* of distinct labels present in the subset U. Using the property (6.8) and considering successively the case $e \in U$ and $e \in V$, then the case $e \in V$ and $e \notin U$, and finally $e \notin V$ and $e \notin U$, it is easy to show that the function c is submodular.

[1] A lattice is a finite ordered set such that any two elements have a unique least upper bound and a unique greatest lower bound.

As a second example take, for the set E, a collection of p vectors of the vector space $\mathbb{R}^n$ where n and p are some integers. To every subset F of E, the function f associates the dimension of the vector space spanned by the vectors of F. Again, using (6.8) one easily sees that the function f is submodular.

Another example is provided by the following function C. Take a graph $G = (V, E)$ i.e. a collection of points (vertices) V possibly connected by a line (an edge) belonging to $E \subseteq V \times V$. By definition C is a function of the subsets of the V and $C(U \subseteq V)$ is the number of edges having exactly one end in U. This function can be generalized to the case where the edges are directed and weighted, i.e., each edge carries an arrow and a positive number. The function $C(U \subseteq V)$ is then the sum of the weight of the edges having the beginning vertex in U and the ending vertex not in U. This kind of function is called a “cut”. In particular cuts occur in the Random Field Ising Model.

Take any concave real function $\varphi : \mathbb{R} \to \mathbb{R}$ of a real variable, i.e. a function such that for any values a and b, and for every $0 \leq \lambda \leq 1$, $\varphi(\lambda a + (1-\lambda)b) \geq \lambda\varphi(a) + (1-\lambda)\varphi(b)$, then for any finite set V define $g(U) = \varphi(|V|)$ where $|V|$is the cardinality of V (i.e. the number of elements in V). One can show that the function g is submodular.

Finally, it will be shown in this chapter that computing the partition function of a ferromagnetic Potts model on any graph amounts to minimizing a submodular function.

6.3.4 Minimization of Submodular Function

Let us now consider the problem of minimizing a submodular function. We are given a finite set V and a real-valued function acting on the set of the subsets of V. This function is seen as a ”black box”, i.e. a subroutine which returns the value of $f(V)$ when applied to a subset of V. The problem is to minimize f, i.e., to find a subset $A^\star$, called an optimal set, such that $f(A^\star) \leq f(A)$ for any $A \subseteq V$. If nothing is known about f there is no other method (at least on a non-quantum computer) than to apply the black box f to *all* the $2^{|V|}$ ensembles and thereby to find all the optimal sets. The main result of the theory of submodular functions is that if the function f is submodular then there is an algorithm which finds one optimal set by using the black box f a number of times which is only *polynomial* in the cardinality $|V|$ of the set V (instead of the exponential number $2^{|V|}$ for a general function). Moreover, the number of steps needed to complete the algorithm does not depend on the possible output of the function f, provided these values are rational. Such an algorithm is said to be *strongly polynomial*. From a practical point of view it means that the execution time of the algorithm will be polynomial in the number of elements of the set V. We assume that the possible values that f takes are rational numbers which can be stored on a fixed number of bits.

The result mentioned above was first published in reference [9] in 1981. In this paper the authors utilize the so-called ellipsoid method. However, this method is not a combinatorial one and is far from being efficient. In that respect this result was not quite satisfactory at least for the practical applications. Eighteen years later, Iwata-Fleischer-Fujishige [11], and independently Schrijver [16], discovered a combinatorial method which is fully satisfactory from the theoretical, as well as from the practical, point of view.

The general method uses a mathematical programming formulation. The problem is algebraically expressed as a linear program, i.e., a set of variables y_S associated to each subset $S \subset V$ is introduced, these variables are subjected to constraints, and a linear function F of these variables is to be minimized. The constraints include a set of linear equations and the condition that each of the y_S is zero or one. This last condition is in general extremely difficult to realize. However, it turns out that a theorem due to Edmonds [6] indicates this condition can be simply dropped, and that automatically the set of values y_S which minimize F will all be zero or one! Actually only one variable $y_{S^\star} = 1$ will be non-zero and it is precisely associated to the optimal set. Combined with the dual version of this linear program, it provides a characterization of the optimal set.

This chapter presents the application of the submodular function minimization to the particular case of the determination of the free energy of the Potts model in the limit of a large number of states. The general algorithm mentioned above can be applied, however, due to the specific form of the function to minimize, a more suitable method does exist. The general method will not be discussed here, instead one which is specific to the Potts model will be presented in detail. Let us now present a property which is true for any submodular function and that we will use later. To emphasize that the function f to minimize is defined on all the subsets of a set E, we will label f with the index E as f_E. Let us now consider a subset $F \subseteq E$; one can define a set function on F by $f_F(A) = f_E(A)$ for any $A \subseteq F$. If the function f_E is submodular then its restriction f_F is also submodular. We have the following property:

Proposition. Let $F \subseteq E$ and $e \in E$, if A_F is an optimal set of the set function f_F defined on F, then there will be an optimal set $A_{F\cup\{e\}}$ of the function $f_{F\cup\{e\}}$ defined on $F \cup \{e\}$ such that $A_F \subseteq A_{F\cup\{e\}}$.

To make the notation simpler we denote the function $f_{F\cup\{e\}}$ on $F \cup \{e\}$ by f_1. Let A be an optimal set of f_F on F and B an optimal set of f_1 on $F \cup \{e\}$. One has

$$f_1(A \cup B) \leq f_1(A) + f_1(B) - f_1(A \cap B) \tag{6.9}$$

since f_1 is submodular. But $f_1(A) = f_F(A)$ and $f_1(A \cap B) = f_F(A \cap B)$ since both A and $A \cap B$ are in A. Since A is an optimal set one has $f_F(A) \leq f_F(A \cap B)$ and consequently $f_1(A) - f_1(A \cap B) \leq 0$. Inserting this last inequality into (6.9) one finds that $f_1(A \cup B) \leq f_1(B)$ which proves that $A \cup B$ is one of the optimal sets (Q.E.D.).

This property has an important consequence. Let us suppose that the optimal set has been found for a subset F of E. Then all the elements of E which have been selected as belonging to the optimal set of F will still belong to one optimal set of all the sets $G \supseteq F$. In other words, let us find the optimal set for $\{e_0, e_1\}$ where e_0 and e_1 are *arbitrary* elements of E; then if we find that any of these two elements belongs to the optimal set, it will belong to one optimal set for $F \subseteq E$! Such an algorithm which makes a definitive choice at each step is called a *greedy* algorithm.

6.4 Free Energy of the Potts Model in the Infinite q-Limit

In this section, we put together the two preceding ones and show how the calculation of the free energy for a particular realization of the couplings J_{ij} can be achieved by minimizing a submodular function. The Optimal Cooperation algorithm which does this minimization is introduced [1].

6.4.1 The Method

Consider the function (6.6) $f_P(A) = -(c(A) + w(A))$; this is a set function on the set of all the edges of the lattice. Take two sets of edges $A \subseteq B$ and an edge e. Inspecting the three possible cases: $e \in A$, $e \notin A$ and $e \in B$, $e \notin A$ and $e \notin B$ one sees that $c(A \cup \{e\}) - c(A) \leq c(B \cup \{e\}) - c(B)$, which is the reverse of (6.8), so that the function $-c$ is a submodular function. On the other hand it is straightforward to see that the function $w(G) = \sum_{e \in G} w_e$ verifies $w(A \cup C) + w(A \cap C) = w(A) + w(C)$. It is a so-called *modular* function. Consequently the function (6.6) f_P is a submodular function. In summary we are looking for the sets of edges minimizing the submodular function f_P. This problem is precisely the problem introduced in the previous paragraph, and for which a *strongly polynomial* algorithm has been recently discovered.

Making use of the proposition of the previous section and its consequence stated just afterwards, we will find an optimal set A by finding successively the optimal set for subgraph of the lattice. Let us suppose that an optimal set A_k has been found for the graph G_k with vertices $V_k = \{v_0, \ldots, v_{k-1}\}$ and containing all the edges of G having both extremities in V_k. The edges of the optimal set A_k can be grouped into connected subsets – the set of extremities of these edges is called a cluster (so that a cluster is a set of vertices). The weight of an edge between two clusters is the sum of the weights of all the edges having one extremity in each cluster. The isolated vertices will be considered as clusters of size one. We know from the proposition of the previous section that the embedded cluster structure will be preserved in the optimal set of G_{k+1}. Moreover, if two clusters of A_k are merged into a single cluster in A_{k+1} then the new vertex v_{k+1} will also belong to this cluster. So one of the optimal sets of A_{k+1} has the following cluster structure: some (possibly 0) clusters of A_k are merged together with the new vertex v_{k+1}, the other clusters are unchanged (see Figure 6.1). The problem is now to find the clusters to merge with the new vertex v_{k+1}. This defines the auxiliary problem.

6.4.2 The Auxiliary Problem

It turns out that, due to the specific form of the submodular function f_P, the auxiliary problem can be solved in strongly polynomial time. To do that we introduce another weighted and directed graph, called a network. The vertices of this network are (i) the clusters defined by the optimal set A_k of G_k, (ii) the new vertex v_{k+1} and (iii) two additional sites which we call the source s and the sink t. There are two oriented edges, one in each direction, between the vertices of the network corresponding to cluster which are connected in the original graph. In addition there are edges from the source s to all vertices v and from all vertices v to the sink t. Finally every edge (u, v) carries a weight $c(u, v)$ called a capacity. Figure 6.2 presents the correspondence between the lattice and the network. To fully define the network, we need to set these capacities. We will find them in such a way that *solving the auxiliary problem amounts to finding a minimum cut separating s and t* in this network, a problem which can be efficiently solved [7]. Let us briefly recall what a cut is:

Definition. A cut S separating s and t in a network is a partition of all the vertices into two sets S and T, such that $s \in S$ and $t \in T \equiv \overline{S}$. The weight $C(S)$ of the cut S is the sum of the weights $c(e)$ of all the edges e from S to T.

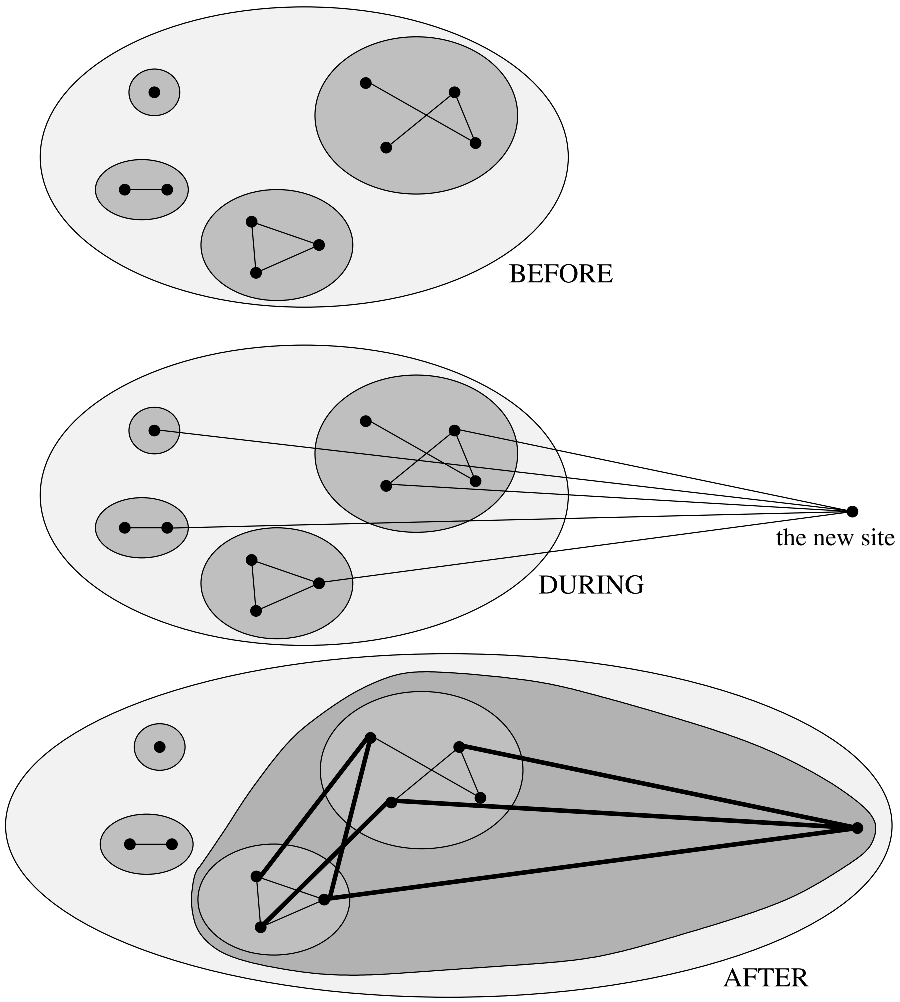


Figure 6.1: Adding a vertex: only the edges belonging to the optimal set are shown in the graphs “BEFORE” and “AFTER”. The thick lines indicate the edges included in the optimal set at this step.

Let us call an *admissible set* a set of edges induced by a connected set of vertices including the new vertex v_{k+1}. The auxiliary problem amounts to finding an admissible set W of edges which maximizes the function $\sum_{e\in W} w(e) + (N - n(W))$, where $n(W)$ is the number of *vertices* in the admissible set W and N the total number of vertices. Equivalently the auxiliary problem is to minimize the function $g(W) = n(W) - \sum_{e\in W} w(e)$. We first add an edge of infinite weight between s and v_{k+1}, so that an arbitrary connected cut S in the network defines an admissible set. We will now choose the capacity $c(u, v)$ in such a way that the minimum cut in the network corresponds to the optimal admissible set A. Then we proceed starting with

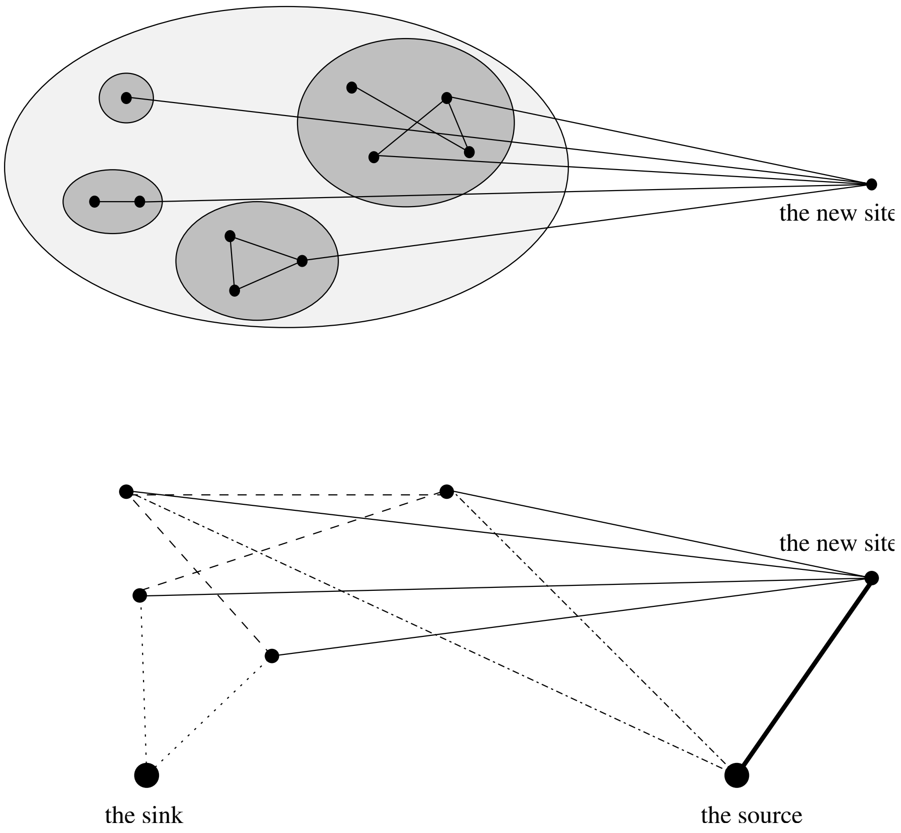


Figure 6.2: Constructing the network associated to the graph: the dotted lines connect each vertex to the sink or to the source according to the condition described in the text, and the thick line carries an infinite weight. Here the edges not belonging to the optimal set are shown as dashed lines.

$S = \{s, v_{k+1}\}$ and adding the vertices one by one. On the one hand adding a vertex v to S will change the capacity of the cut by Δ_C, on the other hand it will change the function g by an amount Δ_g. We will arrange the weights $c(u,v)$ in such a way that $\Delta_C = \Delta_g$ so that finding the minimum cut will provide the set of edges of the original graph which minimizes g and therefore minimizes f_P. A simple inspection shows that

$$\Delta_C(S) = \sum_{u \in V(v) \cap T} c(v,u) - \sum_{u \in V(v) \cap S} c(u,v) + c(v,t) - c(s,v)$$

and

$$\Delta_g(S) = 1 - \sum_{u \in V(v) \cap S} w(uv)$$

where $V(v)$ designates the set of vertices, different from s and t, linked by an edge to v.

Requiring $\Delta_C(S) \equiv \Delta_g(S)$ for any S implies that $c(u,v)+c(v,u) = w(uv)$. It is convenient to choose:

$$c(u,v) = c(v,u) = \frac{1}{2}w(uv) \tag{6.10}$$

for any pair of vertices (u,v) different from s and t. Using this last condition one gets a *single* condition (and not one per set S as one could expect *a priori*):

$$c(v,t) - c(s,v) = 1 - \frac{1}{2}\sum_{u \in V(v)} w(uv) \tag{6.11}$$

A possible choice is

$$\begin{aligned} c(v,t) &= \alpha + 1 + \beta \sum_{u \in V(v)} w(uv) \\ c(s,v) &= \alpha + \left(\beta + \frac{1}{2}\right) \sum_{u \in V(v)} w(uv) \end{aligned}$$

where α and β are two arbitrary positive numbers. However, the other possible choice

$$c(v,t) = \max\left(0, 1 - \frac{1}{2}\sum_{u \in V(v)} w(uv)\right) \tag{6.12}$$

$$c(s,v) = \max\left(0, \frac{1}{2}\sum_{u \in V(v)} w(uv) - 1\right) \tag{6.13}$$

is more suitable since every vertex is connected to s **or** to t.

To summarize we state the main result: the minimum cut S separating $s \in S$ and $t \notin S$ in the network defined above with weights given by (6.10), (6.12), (6.13) is such that the edges having both extremities in S is an optimal set for the function f.

6.4.3 The Max-flow Problem: the Goldberg and Tarjan Algorithm

As explained above, the algorithm needs to find the minimum cut in a series of $|V| - 1$ networks, so that the procedure which finds these minimum cuts is crucial: it is the *engine* of the method. We will not discuss here the min-cut problem in general but refer the reader to [10]. A particularly efficient min-cut finder is the Goldberg and Tarjan algorithm [8]. This algorithm actually solves the dual problem of the min-cut problem the so-called max-flow problem. We do not give here the details and refer to [10] . Note that the Goldberg and Tarjan algorithm has already been used in statistical mechanics to find the ground states of the Random Field Ising Model [4].

6.4.4 About the Structure of the Optimal Sets

The algorithm explained in this paragraph finds one optimal set. It does not give any information about the number of optimal sets. The optimal set actually found does depend on the details of the implementation. However, the algorithm can be tuned in such a way to find one of two special optimal sets that we define below.

Let us suppose that A and B are two optimal sets, from the definition

$$f(A \cup B) \leq f(A) + f(B) - f(A \cap B) \leq f(A)$$

the second inequality comes from B being optimal. We conclude that $A \cup B$ is also an optimal set. In the same way one shows that $A \cap B$ is also optimal. The two following sets

$$\begin{aligned} A_M &= \bigcup_{\text{all optimal sets A}} A \\ A_m &= \bigcap_{\text{all optimal sets A}} A \end{aligned}$$

are optimal. They are of special importance in the applications of the next section. Note that if they are equal then there is only one optimal set.

6.5 Implementation and Evaluation

In this section we discuss the practical implementation of the algorithm, and its evaluation in terms of time and memory requirement.

6.5.1 Implementation

As explained in the previous section, the algorithm works by solving the problem of finding the optimal set on subgraphs of the original graph. In other words to calculate the free energy for a given lattice with N sites, the algorithm proceeds by computing the free energy on $N-1$ sublattices with an increasing number of sites. Moreover the choice of the order in which the sites are added to the current sublattice is arbitrary.

Each step includes three actions: (i) build the network, (ii) find the minimum cut, (iii) contract all the vertices on the source side into a single vertex, keeping track of this contraction since at the end we will need to “uncontract” the vertices. Two main options are possible to represent the data in the computer memory: either one can rebuild completely the data structure representing the network after each step, or alternatively, one can increase the weights of the edges between vertices of the same connected components up to a value sufficiently large to insure that these vertices will never be in different connected components. The latter possibility is less memory consuming, since a single data structure is necessary (the contraction being performed by playing with the weights), whereas in the former case one has to store two data structure, one for the lattice and the other for the network. However, it turns out that the first solution is much faster. This is because the extra time needed to rebuild the data structure at each step is largely compensated for by the gain of finding a cut on a smaller network. Moreover, as detailed below, the bottleneck in this problem is not the memory to store the data, but the time to process them. In the same way it is possible to identify the source s with the new vertex, or to keep two different vertices connected by an edge with a weight $c(s, v_{k+1}) = 1 + \sum_{u \neq v_{k+1}} c(s, u)$. We give the algorithm below.

```
algorithm OptimalCooperation()
begin
    Network := $\emptyset$
    for every vertex $v$ do
    begin
        call AddVertexToNetwork(Network,v);
        $S$ := MinCut(Network);
        call ContractNetwork(Network);
    end
    uncontract all vertices;
    OptimalSet := the edge set induced by the cut;
end
```

```
procedure AddVertexToNetwork(Network,v)
begin
    comment connect vertex $v$ to its neighboring sites as in the lattice
    for $u$ in the lattice neighbor of $v$ do
        for every vertex $x$ in the network to which $u$ belongs do
            $c(v,x) := c(v,x) + w(uv)/2$;
        end do
    $c(x,v)$ := $c(v,x)$;
    end do
    connect the source $s$ and the sink $t$ using weights (6.12-6.13);
end
```

```
procedure ContractNetwork(Network)
begin
    contract the vertices in $S$ into a single vertex $v^*$;
    comment modify the weights accordingly:
    for every vertex $u$ not in $S$ do
        $c(u,v^*) := 0$;
        for every vertex $v$ neighbor of $u$ in $S$ do
            $c(u,v^*) := c(u,v^*) + c(u,v)$;
        end do
        $c(v^*,u)$ := $c(u,v^*)$;
    end do
end
```

The procedure *Min-Cut* is not described here: we have used the well known and efficient Goldberg and Tarjan algorithm [8].

6.5.2 Example of Application

We have applied this algorithm to various two-dimensional and three-dimensional lattices. A realization of the disorder is chosen accordingly to a probability distribution. In practice all the weights $w(e)$ on the edge e are rational numbers with a common integer denominator q. In other words, we choose an integer $p(e)$ for each edge and set $w(e) = \frac{p(e)}{q}$. To work only with integers one maximizes the product qf:

$$qf(A) = qC(A) + \sum_{e \in A} p(e)$$

It is clear that if q is small compared to all the $p(e)$, then all the weights $w(e)$ will be large and the optimal set will be the set of all edges. On the contrary, if q is large all the weights will be small and the optimal set will be empty. These two situations are easy to handle. Between this two limits the optimal set grows, and for a precise value q_c of q, which depends on the lattice, the optimal set percolates. This value corresponds to a phase transition. Depending on the lattice under consideration and on the distribution of the random variables $p(e)$ this transition can be first or second order.

In Figure 6.3, one optimal set is shown for a lattice where each edge carries a weight 1/6 or 5/6 with probability one-half (i.e., it is a critical point). The edges from the optimal set belonging to the percolation cluster are shown in black, while the others are shown in gray. It took two weeks on a Pentium III (1.7 Ghz) to find the optimal set. Note that the optimal set is one of the $2^{2\times 512\times 512} \simeq 2.6\ 10^{157827}$ possible edge sets!

6.5.3 Evaluation of the CPU Time

From the algorithmic point of view, the critical value corresponds roughly to the "worst case": the CPU time needed to find the optimal set is the longest when q is close to q_c. The fluctuations in the execution time to find an optimal set for different realizations of the disorder on the same lattice are large and can vary by a factor greater than two. A crude estimation can be performed averaging the execution time over several samples of the same size, but corresponding to different disorder realization. This empirical evaluation gives

$$t \simeq AN_v^{2.4}$$

at least when N_v, the number of vertices of the lattice, is larger than a thousand. This roughly applies for many two-dimensional and three-dimensional regular lattices. The constant A for a processor Intel(R) Xeon(TM) (2.40GHz) is found to be close to $0.7\ 10^{-7}$ seconds. For a square lattice it gives an execution time varying like $t = 0.7 \times 10^{-7} L^{4.73}$.

6.5.4 Memory Requirement

Adopting the scheme presented above, the memory requirement is $(6n_e+3n_v)+(7m_e+5m_v)$ where n_e and n_v are the number of edges and vertices of the lattice under consideration, while m_e and m_v are the number of edges and vertices of the largest network one need during the execution of the algorithm. In the worst case where the final optimal set is empty, one has

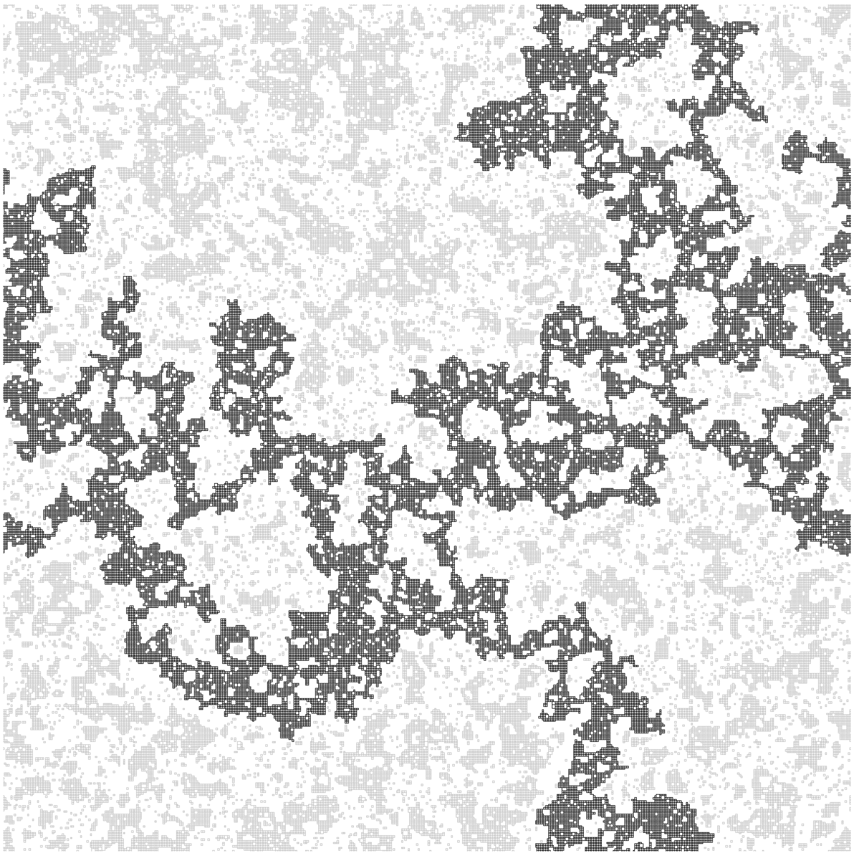


Figure 6.3: A 512×512 lattice. The edges of the optimal set belonging to the percolating cluster are shown in black, and the edges of the optimal set not belonging to the optimal set are in grey.

$m_e = n_e$ and $m_v = n_v$. For a $L \times L$ square lattice this gives $13n_e + 8n_v = 34L^2$words. For a 512×512 lattice it is around 44 Mbytes. This shows that, as previously quoted, that the limiting factor is the execution time and not the memory requirement.

6.5.5 Various Possible Improvements

Various improvements or adaptations are possible. We present some of them in this subsection.

Let us first mention a useful remark. If the optimal set of a graph is known, it is easy to find the optimal set of the same graph where the weight has been increased on some edges. A special procedure has been devised to perform this specific task [16], but it is straightforward with the Optimal Cooperation algorithm. Suppose the capacity on the edge uv is increased by $\Delta(uv)$. To find the new optimal set simply add a new vertex $u^\star$ linked to u by an edge of weight $w(uu^\star) = \Delta(u, v)$ and linked to v by an edge of infinite weight, and apply the Optimal Cooperation algorithm. Note that this trick can be used to add new edges.

The parameter q introduced at the beginning of this section is actually a temperature, and $f(A(q))$ (where $A(q)$ is the optimal set for q) is the free energy. Therefore to study the temperature dependence of the free energy, one is lead to first randomly choose the parameter $p(e)$, and second vary q finding the optimal set for each q. Now, if $A(q_1)$ and $A(q_2)$ are the optimal sets at $q = q_1$ and $q = q_2$, and if $q_1 < q_2$ then $A(q_2) \subseteq A(q_1)$. This is a direct consequence of the preceding remark. It is then possible to greatly improve the performance of the algorithm. Indeed, one can consider the q's in *decreasing* order and contract the vertices of the same connected components after each determination of the optimal set. So the graph in which one seeks the optimal set is getting smaller and smaller when q decreases, and when q becomes sufficiently small, this graph reduces to a single vertex.

It is worthwhile noticing that one can reduce the computing time to get the optimal set with a suitable labelling of the vertices. The best choice is one in which the intermediate graphs have as few edges as possible. For example, for a bipartite graph, choosing first all sites of one of the partition, gives good results, dividing the CPU time by a factor of the order of two.

As already mentioned, the two optimal sets $A_m = \cap_i A_i$ and $A_M = \cup_i A_i$ are important (for example, the quantity $\delta(q) = \frac{1}{q}\Sigma_{e \in A_M/A_m} p(e)$ is a jump in the internal energy). As already mentioned, the min-cut finder actually first finds the max-flow [10], and then one deduces a min-cut from this flow, although there are several. However, it is possible to implement the algorithm in such a way as to find the min-cut of largest cardinality or the one of smallest cardinality. This can be used to find A_m or A_M at will. Note also that this allows us to determine whether or not the optimal set is unique.

6.6 Conclusion

We have presented an efficient strongly polynomial algorithm, the Optimal Cooperation algorithm, to compute the free energy of a ferromagnetic Potts model on an arbitrary lattice in the limit of an infinite number of states. The calculation is performed for a given choice of the Boltzmann weights, i.e., for fixed values of the positive coupling constants and temperature.

The existence of this algorithm is a consequence of the fact that the calculation of the free energy of this model has been transformed into a combinatorial optimization problem. Such transformation is valid in the limit of an infinite number of states. The algorithm proceeds solving the problem on intermediary lattices. This enables various improvements of the basic methods.

The crucial property is that the function to minimize is submodular. Such a situation where a problem of statistical mechanics gives rise to the minimization of a submodular function has already been encountered in the Random Field Ising Model [2, 4]. Since lattice Statistical Mechanics and Combinatorial Optimization appear to be often linked [3], it is likely that other problems of minimization of submodular functions will occur in physics. Keeping in mind that there exist efficient methods to perform this minimization is important.

Acknowledgments

I wish to thank M. Preissmann and M.T. Mercaldo for many discussions and a critical reading of the manuscript.

References

[1] J.-Ch. Anglès d'Auriac, F. Iglói, M. Preissmann, and A. Sebő, *Optimal Cooperation and Submodularity for Computing Potts' Partition Functions with a Large Number of States*, J. Phys. A **35**, 6973 (2002).

[2] J. C. Anglès d'Auriac, M. Preissmann, R. Rammal, *The Random Field Ising Model: Algorithmic Complexity and Phase Transition*, J. Physique Lett. **46**, 173 (1985).

[3] J.-Ch. Anglès d'Auriac, M. Preissmann, and A. Sebő, *Optimal Cuts in Graphs and Statistical Mechanics*, Journal of Math. and Computer Modelling **26**, 1 (1997).

[4] J. C. Anglès d'Auriac and Nicolas Sourlas, *The 3-d Random Field Ising Model at Zero Temperature*, Europhys. Lett. **39**, 473 (1997).

[5] W.H.Cunningham, *Optimal Attach and Reinforcement of a Network*, Journal of the ACM **32**, No 3, 549 (1985).

[6] J. Edmonds, in *Combinatorial Structures and Their Applications,* R. Guy, H. Hannani, N. Sauer, and J Schóonheim eds., (Gordon and Breach, 1977).

[7] L.R. Ford and D.R. Fulkerson, *Maximal Flow Through a Network*, Canad. J. Math **8**, 399 (1956).

[8] A.V Goldberg and R.E. Tarjan, *A new Approach to the Maximum-flow Problem*, Journal of the Association for Computing Machinery **35**, 921 (1988).

[9] M. Grötschel, L. Lovász, A. Schrijver, *The Ellipsoid Method and its Consequences in Combinatorial Optimization*, Combinatorica **1**, 169 (1981).

[10] A.K. Hartmann und H. Rieger, *Optimization Algorithms in Physics*, (Wiley-VCH, Berlin, 2002).

[11] S. Iwata, L. Fleischer, and S. Fujishige, *A Combinatorial Strongly Poly-nomial Algorithm for Minimizing Submodular Functions*
Journal of the ACM**48**, Issue 4, 761 (2001).

[12] R. Juhász, H. Rieger, and F. Iglói, *Random-bond Potts Model in the Large-q Limit*, Phys. Rev. E **64**, 056122 (2001).

[13] P.W. Kasteleyn and C.M. Fortuin, *Phase Transitions in Lattice Systems with Random Local Properties*, J. Phys. Soc. Jpn **26** (Suppl.), 11 (1969).

[14] Potts, R. B., *Some Generalized Order-disorder Transformations*, Proc. Camb. Phil. Soc. **48**, 106. (1952).

[15] A. Schrijver, *Combinatorial Optimization – Polyhedra and Efficiency Volume B.*, (Springer-Verlag, Berlin 2003).

[16] A. Schrijver, *A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time*, Journal of Combinatorial Theory Ser. B **80**, 346 (2000).

[17] F.Y. Wu, *The Potts Model*, Rev. Mod. Phys. **54**, 235 (1982).

Part II: Phase Transitions in Combinatorial Optimization Problems

7 The Random 3-satisfiability Problem: From the Phase Transition to the Efficient Generation of Hard, but Satisfiable Problem Instances

Martin Weigt

7.1 Introduction

In natural sciences and in artificial systems, there exist many combinatorial optimization and decision problems whose solution requires computational time resources which grow exponentially with the system size N. Concrete examples are NP-hard optimization and cryptographic problems in computer science, glassy systems and random structures in physics and chemistry, random graphs in mathematics, and scheduling problems in real-world applications.

In practice, the exact numerical solution of such problems is thus frequently restricted to small problems. The development of fast and powerful algorithms for these problems is therefore of primary relevance for both their theoretical study and for practical applications. The other chapters contain many examples of highly sophisticated algorithms. For their comparison and evaluation one needs firm benchmarks with the following properties: On one hand, the problem should be computationally demanding for the solvers under consideration, requiring solution times being exponential in the size N of the benchmarks. On one hand, these instances should be generated in a fast way, e.g. in a time growing linearly in N, and they should have well-controlled solution properties. In this way they allow for a systematic analysis and comparison of different solvers. The scope of this chapter is the presentation of such a generator [6]. It is based on the NP-complete 3-satisfiability problem (3-SAT) [13], and outputs hard but satisfiable logical formulas of a given system size.

The main idea for the construction of such hard and solvable problems is straightforward: We hide a known solution within a multitude of coexisting meta-stable configurations (i.e., local minima of a cost function which has to be optimized). These constitute dynamical barriers hindering algorithms from finding the known solution. In the physical approach based on a mapping from 3-SAT to a spin-glass model [22, 23], such random configurations correspond to glassy states [7, 18, 20]. It is to be noted, however, that many previous attempts to implement this idea were unsuccessful, because the random structure was usually easy to remove, or the knowledge that a solution has been forced can be exploited in order to find it. In the instances we propose, instead, the presence of a known solution does not alter the structure of the glassy state, which confuses the solver and makes the problem hard.

A closely related, important application of these ideas appears in cryptography [30]. Imagine you want to code and store a password (corresponding to the imposed solution) on your

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

computer. There are three tasks you have to fulfill. The verification of the password should be efficient, but decoding it from the stored structure has to be extremely time-consuming. To be of practical applicability, the coding itself should be very efficient. These three points are met by the generator described in this chapter. Once the password is given in binary variables, these can be understood as a Boolean configuration. The stored information will be a 3-SAT formula which is constructed such that the password satisfies the formula. Since 3-SAT is NP-complete, we know that the verification can be achieved in polynomial time, but the decoding may take exponential time. For security reasons one has to make the decoding as hard as possible. This situation, usually referred to as a “worst case” in complexity theory, is the aim to be reached in the cryptographic application.

This chapter is organized in the following way. After these introductory notes, we provide an overview of what is known about random 3-SAT and the phase transitions existing there. We are going to first define the problem, provide some numerical observations, and eventually sketch the statistical-mechanics approach which was successfully applied to clarify the solution properties of random 3-SAT. In Section 7.3 we come to the main topic of this chapter. A generator of satisfiable 3-SAT formulas is described. Relying on methods borrowed from statistical mechanics of disordered systems, this generator is optimized in a way that state-of-the-art algorithms require exponential times to solve the generated formulas. At the end, the results are summarized, and an outlook is given.

7.2 Random 3-SAT and the SAT/UNSAT Transition

To start with, we give an overview of phase transitions appearing in random 3-SAT problems. They have played a major role in developing a typical-case computational complexity theory [4, 29] during the last few years.

The 3-SAT problem is defined over a set of N *Boolean variables* $\{x_i = 0, 1\}_{i=1,\dots,N}$, with 0=FALSE and 1=TRUE. They are restricted by M *3-clauses* $\{C_\mu\}_{\mu=1,\dots,M}$, each one consisting of exactly three Boolean variables, which may appear negated or unnegated, and which are connected by logical OR operations ($\vee$), e.g., $C_\mu = (x_i \vee \overline{x}_j \vee x_k)$. To satisfy such a clause, at least one of the contained variables has to be assigned in the right way. Thus the clause is unsatisfied only iff all three variables take the wrong values, for the example clause given above, the only unsatisfying assignment would be $x_i = 0, x_j = 1$ and $x_k = 0$. The other $2^8 - 1 = 7$ assignments fulfill the condition posed by the clause. In the *formula* F, all clauses are connected by logical AND operations ($\wedge$),

$$F = \bigwedge_{\mu=1}^{M} C_\mu \; . \tag{7.1}$$

The formula is *satisfied*, if it evaluates to TRUE. This means that *all* M clauses have to be *satisfied simultaneously*.

The clauses can be understood as basic constraints to a set of discrete variables, and all constraints have to be fulfilled simultaneously. In this sense, 3-SAT can be considered as a prototype constraint-satisfaction problem (CSP). In fact, it is the most fundamental NP-complete problem (NP = nondeterministic polynomial [13]), even the first one which was shown to be NP-complete [9]. This means, in particular, that all currently known solution algorithms require running times which grow exponentially with the number N of Boolean variables and the number M of clauses. On the other hand, the verification of a candidate solution is efficient, i.e., it takes only polynomial time. Given this candidate assignment, we just have to check all the M clauses, one by one, to determine if they are satisfied or not.

The exponential running time mentioned before is to be understood in the worst-case scenario of traditional complexity theory [13]. For each system size, there has to be at least one extremely complicated instance which cannot be solved efficiently by the algorithm under consideration. The question of how far this scenario is relevant in practical applications, is still under discussion. One popular possibility to complement the worst-case picture, and to obtain some information about what is happening in *typical cases*, is given by the investigation of randomly generated 3-SAT formulas [4, 29]. For each clause, three variables were selected randomly and independently among all N Boolean variables, and each one was negated with probability 1/2. The relevant control parameter for the formulas generated in this way is their *constraintness*, as measured by the clauses-to-variables ratio $\alpha = M/N$.

7.2.1 Numerical Results

The formulas generated in this way are random, so their satisfiability is also a random variable. A first interesting quantity is therefore the probability that a random 3-SAT formula is satisfiable, as a function of the constraintness α, for fixed number N of Boolean variables. It is obvious that this probability will be close to one for small α, because there are only few constraints which are unlikely to induce a contradiction. The probability, however, is a monotonously decreasing function of α, since more and more constraints are added.

Measuring this quantity numerically, an astonishing result was found. The decrease from probabilities close to one to those close to zero is not given by a slowly varying function of α, but it is characterized by a sharp drop in a limited α-region close to $\alpha \simeq 4.2 \ldots 4.3$ [21, 26], cf. also Figure 7.1. This drop becomes sharper and sharper if the number of variables is increased, and eventually it tends to a step function in the thermodynamic limit $N \to \infty$ [12]. Formulated in a physical language, there exists a *phase transition* at a critical point $\alpha_c \simeq 4.27$. For $\alpha < \alpha_c$, the randomly generated formula is *almost surely* satisfiable, i.e. the probability of being satisfiable tends to one in the large-N limit. For $\alpha > \alpha_c$, on the other hand, the formula becomes almost surely unsatisfiable, i.e., it contains unresolvable contradictions. This transition is called the SAT/UNSAT-transition.

Even more interestingly for computer scientists, this phase transition is connected to a characteristic resolution time pattern, referred to as the *easy–hard–easy transition*. To get computer- and implementation-independent results, resolution times are usually measured by the number of algorithmic steps. In addition, the median solution time is considered instead of the average solution time, since the latter is dominated by exponentially rare events requiring exponentially longer resolution time, instead of measuring the most probable, or *typical* running time. For readers who are familiar with the statistical mechanics of disordered systems, this argument is analogous to averaging the logarithm of the partition function of a model with quenched disorder, instead of directly averaging the partition function itself.

This median solution time, plotted again as a function of α, shows an interesting behavior. As can be seen in Figure 7.1, it is polynomial for small α. Exponential solution times show

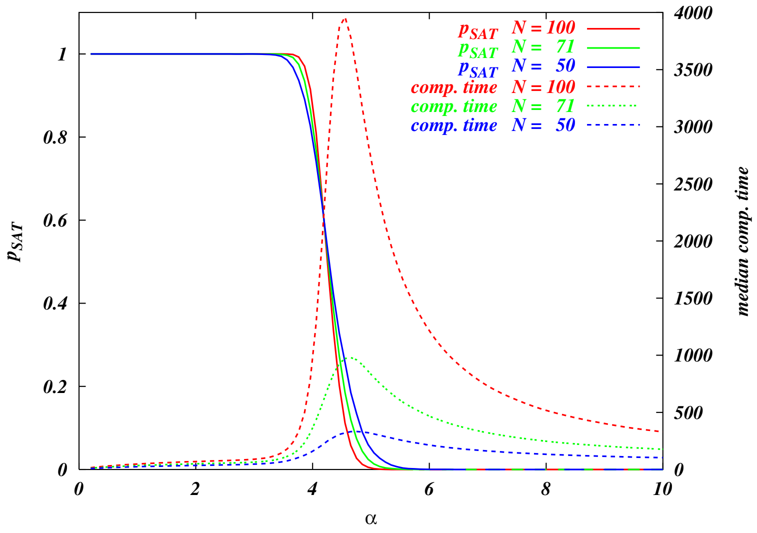


Figure 7.1: SAT/UNSAT transition: The figure shows the probability that a randomly generated 3-SAT-formula is satisfiable, as a function of the constraintness, and the median resolution time required by a complete algorithm. One can clearly see the probability drop close to $\alpha_c \simeq 4.27$ which sharpens with increasing variable number N. The median resolution time shows a pronounced exponential maximum close to α_c.

up only at some algorithm-dependent threshold α_d which is located deep inside the satisfiable phase, i.e. $\alpha_d < \alpha_c$. Directly at the phase transition point α_c a pronounced exponential maximum of the required running time is found, for larger α the instances become again simpler to solve, even if the resolution time still grows exponentially with N. The hardest formulas to solve are thus found at the phase boundary, they are said to be critically constrained. A simple intuitive explanation can be given here. For small α, only a few constraints are present. So there are many solutions, and one of them can easily be found. This task becomes harder and harder if the number of constraints grows, and the number of solutions goes down. For large α, i.e., inside the unsatisfiable phase, the structures leading to logical contradictions in the formula become smaller for growing constraintness. They are therefore more easily identified, and the unsatisfiability of a formula is proved more easily. A deeper and quantitative understanding can be achieved using tools from non-equilibrium statistical mechanics. This will be presented in Chapter 8.

7.2.2 Using Statistical Mechanics

Here we provide a short overview of what is known about this problem from an analytical point of view on the phase transition itself, and on the statistical solution properties.

The rigorous mathematical knowledge is not yet very detailed, even the very existence of a well-defined SAT/UNSAT threshold α_c is not completely established. However, Friedgut proved some years ago that the drop in the probability of satisfiability becomes sharp in the thermodynamic limit [12], but could not prove the convergence to a specific α_c. There are, however, many rigorous lower and upper bounds to the threshold, for an overview see [1, 10, 11]. So far, an oscillating $\alpha_c(N)$ would be the only mathematically consistent possibility besides convergence, but this possibility is widely believed to be extremely unlikely.

More detailed results can be obtained using statistical physics. The existence of a phase transition in random 3-SAT provides obviously a big temptation to apply tools developed in the field of statistical physics of disordered systems, such as the replica trick or the cavity method. Even if many of these methods still lack a rigorous foundation, there is a long-lasting experience with numerous applications, and the methods are generally believed to lead to exact results. The analysis of random 3-SAT is based on a representation in terms of a diluted spin-glass model [22]. Boolean variables $x_i = 0, 1$ are mapped to Ising spins $S_i = -(-1)^{x_i}$. The clauses are uniquely represented by the matrix

$$c_{\mu,i} = \begin{cases} +1 & \text{if } x_i \in C_\mu \\ -1 & \text{if } \overline{x}_i \in C_\mu \\ 0 & \text{else.} \end{cases} \tag{7.2}$$

The Hamiltonian counts the number of unsatisfied clauses,

$$\mathcal{H} = \sum_{\mu=1}^{\alpha N} \delta_{\sum_i c_{\mu,i} S_i, -3} \tag{7.3}$$

which can be easily rewritten as a multinomial in the Ising spins,

$$\mathcal{H} = \frac{\alpha}{8} N - \sum_{i=1}^{N} H_i S_i - \sum_{i<j} T_{ij} S_i S_j - \sum_{i<j<k} J_{ijk} S_i S_j S_k \; . \tag{7.4}$$

This Hamiltonian contains random external fields, two- and three-spin-interactions, representing the quenched disorder of the model. Their values are given by the random choice of the clauses,

$$\begin{aligned} H_i &= \frac{1}{8} \sum_\mu c_{\mu,i} \\ T_{ij} &= -\frac{1}{8} \sum_\mu c_{\mu,i} c_{\mu,j} \\ J_{ijk} &= \frac{1}{8} \sum_\mu c_{\mu,i} c_{\mu,j} c_{\mu,k} \; . \end{aligned} \tag{7.5}$$

The ground states of this Hamiltonian correspond to those assignments of all Boolean variables which violate the minimal number of clauses. In the satisfiable phase, the ground-state energy therefore equals zero, whereas it is strictly positive in the unsatisfiable phase. Consequently, if we were able to determine the ground-state energy as a function of the clauses-to-variables ratio α, we were also able to identify the SAT/UNSAT threshold.

In the statistical mechanics approach, a weight $e^{-\beta\mathcal{H}}$ is assigned to every spin configuration, with $\beta = 1/T$ being a (formal) inverse temperature. For positive β this weight obviously favors assignments of low-energy, and it gets more and more concentrated in such configurations if $\beta \to \infty$. In the zero-temperature limit, i.e. for $\beta \to \infty$, only the ground states keep non-zero probability of appearance. For our combinatorial problem of determining the satisfiability of a formula F leading to a specific Hamiltonian $\mathcal{H}$, we are therefore interested in the zero-temperature thermodynamical properties of the model.

In some pioneering works [22, 23], these ground-state properties were analyzed on the basis of a replica symmetric approximation. The resulting threshold, $\alpha_{\mathrm{rs}} = 4.6$, is obviously larger than the numerical result, and replica-symmetry breaking effects had to be included. Including these into a variational approach [7], a second phase-transition was found to exist inside the satisfiable phase, as is represented schematically in Figure 7.2: Below some threshold α_d, replica symmetry is exact. Pictorially this means that all solutions of a formula are collected in one large, connected cluster inside the configuration space $\{\pm 1\}^N$. At α_d, this cluster breaks discontinuously into an exponential number of smaller clusters, each one still containing an exponential number of solutions. Each pair of these clusters is separated by an extensive Hamming distance. The number of these clusters decreases with increasing constraintness α, until it eventually vanishes at the SAT/UNSAT transition α_c. Above this point, ground states have non-zero energy, and almost all of them are collected in a subexponential number of clusters. The conjectured exact position of the two transitions was recently established using the cavity approach [18, 20]: The clustering transition is located at $\alpha_d \simeq 3.92$, whereas the formulas become unsatisfiable with probability one at $\alpha_c \simeq 4.267$.

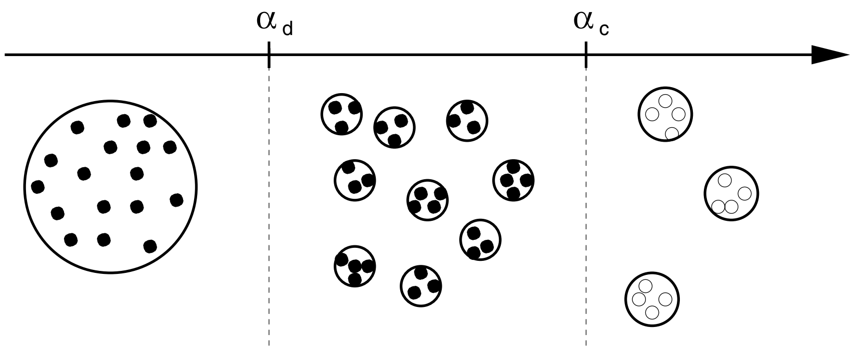


Figure 7.2: Schematic representation of the solution space structure. Below α_d, solutions are collected in one large cluster. Between α_d and α_c, an exponential number of solution clusters can be found. This clustering is accompanied by a proliferation of metastable states, i.e., local minima of $\mathcal{H}$. Above α_c, the formula is not satisfiable any more. Almost all ground states are collected in a subexponential number of clusters, and they have strictly positive energy.

7.3 Satisfiable Random 3-SAT Instances

Taking the numerical results presented in the last section, the hardest-to-solve problems are found close to the SAT/UNSAT transition. It is therefore tempting to use these problems as benchmarks. Generating a random 3-SAT formula takes only linear time, and a systematic check of the N-dependence of the performance of an algorithm can thus be implemented easily. In fact, this is frequently done [27], and it works perfectly for *complete* algorithms, i.e., for algorithms proving whether an input formula is satisfiable or not. Frequently one uses, however, *incomplete* algorithms, in most cases they are of stochastic nature and perform some kind of random walk in configuration space, guided by local heuristic criteria [16, 28]. The most famous and, in its current implementation, also most efficient incomplete algorithm is *walk-SAT* [28]. The central algorithmic step is the following:

1. Randomly select an unsatisfied clause.
2. With probability p perform a *walk step*, with probability $1 - p$ a *greedy step*:
   (a) *Walk step:* Select randomly one of the variables appearing in the clause, and flip its truth value. In this way, the selected clause becomes satisfied. On the other hand, other clauses become unsatisfied if they were satisfied only by the selected variable before flipping. In this way, the algorithm is able also to increase the number of unsatisfied clauses (or energy), and to escape from local minima.
   (b) *Greedy step:* For each of the variables in the selected clause, a certain non-negative number of clauses would become unsatisfied by flipping the variable. Select one leading to the minimal number of newly unsatisfied clauses, and flip it.
3. Iterate 1. and 2. until all clauses are satisfied.

This step describes the very basic version of walk-SAT. There are many heuristic modifications taking into account other local structures than the one used in the greedy step. The most recent implementation of walk-SAT can be downloaded from the SATLIB web page [27]. Stopping only if a solution is hit, these stochastic algorithms are not able to distinguish unsatisfiable formulas from those formulas, which are just too hard to be solved by the algorithm in a reasonable time. For these algorithms it is important to use test instances which are known to be satisfiable. One obvious possibility [27] is to filter the problems at the phase boundary by complete algorithms, and to keep only the satisfiable ones. This method is limited by the small values of N and M which can be handled by the filtering algorithms, thus making the generation itself exponentially long. In addition, the hardest instances in the transition region are the unsatisfiable ones. Other approaches use mappings from various hard problems to 3-SAT, including, e.g., factorization [15], graph coloring [14], and Latin square completion [2].

Here we follow a completely different approach [6]. The idea is very simple: We choose an arbitrary assignment of our logical variables (the so-called *forced solution*) and include into our logical formula, with some prescribed probability, only those clauses which are satisfied by this assignment. Without loss of generality, we restrict ourselves to generating formulas which are satisfied by $x_i^{(0)} = 1,\ \forall i = 1, \ldots, N$. This assignment can be changed to any other one $\vec{x}^{(1)} = (x_1^{(1)}, \ldots, x_0^{(1)})$ by a local gauge transformation of the generated 3-SAT formula:

One has to simply exchange x_i and $\overline{x}_i$ for all i with $x_i^{(1)} = 0$. Restricting the discussion to $\vec{x}^{(0)}$, we find, for any triple $i, j, k \in \{1, \ldots, N\}$, the following situation for the $2^3 = 8$ possible clauses including x_i, x_j and x_k:

$$\begin{array}{llll}
\text{type 0} & x_i \vee x_j \vee x_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_0 \\
\\
\text{type 1} & \overline{x}_i \vee x_j \vee x_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_1 \\
\text{type 1} & x_i \vee \overline{x}_j \vee x_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_1 \\
\text{type 1} & x_i \vee x_j \vee \overline{x}_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_1 \\
\\
\text{type 2} & \overline{x}_i \vee \overline{x}_j \vee x_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_2 \\
\text{type 2} & \overline{x}_i \vee x_j \vee \overline{x}_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_2 \\
\text{type 2} & x_i \vee \overline{x}_j \vee \overline{x}_k & \text{satisfied by } \vec{x}^{(0)} & \text{probability } p_2 \\
\\
\text{type 3} & \overline{x}_i \vee \overline{x}_j \vee \overline{x}_k & \text{violated by } \vec{x}^{(0)} & \text{forbidden}
\end{array} \tag{7.6}$$

i.e., only the last one containing three negated variables is not satisfied by the forced solution $\vec{x}^{(0)}$, and is therefore forbidden in the formula. The other seven clauses, denoted type 0,1, or 2 according to the number of negated variables, are satisfied by $\vec{x}^{(0)}$, i.e., a formula composed by these clauses is forced to have this solution. The generation of a random formula is now achieved as follows:

- For each $\mu = 1, \ldots, M = \alpha N$, randomly and independently select three distinct numbers $i_\mu, j_\mu, k_\mu \in \{1, \ldots, N\}$.
- With probability p_0, set clause $C_\mu = x_{i_\mu} \vee x_{j_\mu} \vee x_{k_\mu}$, and so on with the different allowed clause types according to the probabilities given above.
- Set $F = \bigwedge_{\mu=1\ldots M} C_\mu$.

At the moment, the selection probabilities p_0, p_1 and p_2 are only restricted by normalization, $p_0 + 3p_1 + 3p_2 = 1$. As we will see in the following, the typical hardness of formula F depends crucially on the choice of their values. We will see in particular, that typically hard instances are generated if the parameters are chosen as follows:

$$\begin{aligned}
\alpha &= M/N > 4.25 \\
p_0 &\in (0.077, 0.25) \\
p_1 &= (1 - 4p_0)/6 \\
p_2 &= (1 + 2p_0)/6
\end{aligned} \tag{7.7}$$

To understand this model, and to find values for p_0, p_1 and p_2 such that the instances are as hard as possible, we again use the statistical mechanics approach described in the last section. The results are corroborated by numerical simulations based on both complete (Davis–Putnam algorithm, see the chapter of S. Cocco, L. Ein-Dor, and R. Monasson) and randomized (walk-SAT and simulated annealing) algorithms. Let us therefore recall the Hamiltonian counting the number of unsatisfied clauses, and formulated for the Ising model:

$$\mathcal{H} = \frac{\alpha}{8} N - \sum_{i=1}^{N} H_i S_i - \sum_{i<j} T_{ij} S_i S_j - \sum_{i<j<k} J_{ijk} S_i S_j S_k \tag{7.8}$$

with the couplings given in Eqs. (7.5) and (7.2). They fluctuate from sample to sample, with disorder-averages

$$\begin{aligned}\overline{H_i} &= \frac{3\alpha}{8}(p_0 + p_1 - p_2)\\ \overline{T_{ij}} &= \frac{3\alpha}{4N}(-p_0 + p_1 + p_2)\\ \overline{J_{ijk}} &= \frac{3\alpha}{4N^2}(p_0 - 3p_1 + 3p_2)\end{aligned} \tag{7.9}$$

Since we know that the formula has a forced solution, we also know that the ground-state energy of $\mathcal{H}$ equals zero. Thus we cannot expect to identify a phase transition by a change in the ground-state energy. Here we will find different satisfiable phases, one being relatively easy to solve, another being much harder. In physical language, the first one will be of paramagnetic nature, and no spin will be frozen to its forced value $S_i = 1$. The hard phase will be a ferromagnetic one, with all solutions collected in a cluster around the forced solution. In particular, we will see that this phase is characterized by a large *backbone* which is defined as the set of all spins being constantly assigned to the forced-solution value in all solutions. This backbone has an obvious relation to the hardness of a formula [23]. If, in any algorithmic step, one of the backbone variables is assigned the wrong value, no solution can be found any longer before flipping this specific variable again. This backbone has, of course, to be well-hidden such that it cannot be identified by simple local criteria. We will therefore see that the generated formulas are especially hard if the paramagnetic phase appears in a first-order transition with a large backbone.

7.3.1 The Naive Generator

Going back to the class of generators proposed above, one could naively use $p_0 = p_1 = p_2 = 1/7$ (*model 1/7*), choosing any of the allowed clauses with the same probability. This generator, including some extensions, [5, 17, 24] is known to be efficiently solvable by local search procedures [2]. In our walk-SAT implementation, the maximal resolution time grows as $t \propto N^{1.58}$, and large systems of sizes up to $N \simeq 10^4$ can be easily handled, see Figure 7.3.

The statistical mechanics approach clarifies this result. The proposed generator behaves as a paramagnet in an exterior random field, and no ferromagnetic phase transition appears. The reason is that unnegated variables appear more frequently in the allowed clauses (7.6) than negated ones, leading to average local fields $\overline{H_i} = 3\alpha/56$ pointing into the direction of the forced solution $\vec{x}^{(0)}$. This bias can be exploited by local search procedures, as done, e.g., in walk-SAT, such that the algorithm efficiently finds a solution.

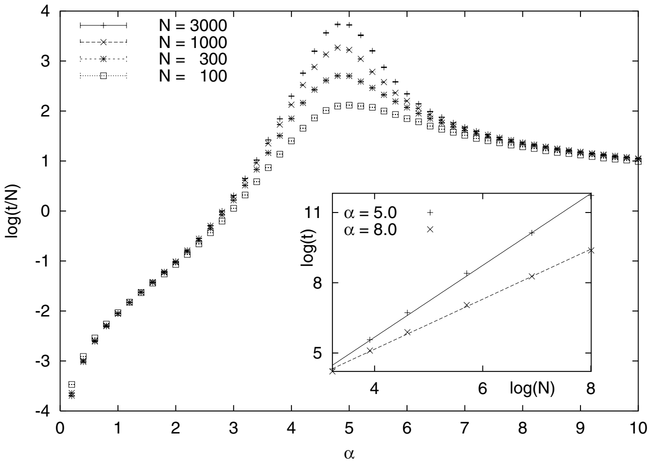


Figure 7.3: Typical walk-SAT complexity for model 1/7. We show the average value of $\log(t/N)$. We find a clear data collapse for small α in the linear regime, $t \propto N$. The complexity peak at $\alpha \simeq 5$ grows polynomially as shown in the inset. The slope of the straight line in the inset is 1.58 for $\alpha = 5.0$, and 1.07 for $\alpha = 8.0$.

7.3.2 Unbiased Generators

To avoid this, we can fix the average local field to zero by choosing $p_0 + p_1 - p_2 = 0$. The probabilities are thus restricted by

$$\begin{aligned} 0 &\leq p_0 \leq 1/4 \\ p_1 &= \frac{1-4p_0}{6} \\ p_2 &= \frac{1+2p_0}{6} \end{aligned} \tag{7.10}$$

There is still one free parameter left. We therefore concentrate first on the extreme cases $p_0 = 0$ (*model 1/6*) and $p_0 = 1/4$ (*model 1/4*), and finally on the intermediate interval.

7.3.2.1 Model 1/6

Let us begin the discussion of these possibilities with the case $p_0 = 0, p_1 = p_2 = 1/6$ (*model 1/6*). In this (and only this) case, there is a second guaranteed solution: $x_i = 0, \forall i$. The average $\overline{J_{ijk}}$ vanishes, too. The model is paramagnetic at low α, and undergoes a second-order ferromagnetic transition at $\alpha \simeq 3.74$ (see full curve in Figure 7.4). But also in the ferromagnetic phase the backbone is still zero as long as $\alpha \leq 4.91$. At this point it appears continuously from strongly magnetized spins.

The existence of this second-order transition can be understood from the fact that the two-spin interactions T_{ij} have a positive mean, i.e., they are on average ferromagnetic. This can also be seen by investigating the six allowed clauses. No variable appears more frequently unnegated or negated. If we, however, fix variable x_i to a specific value, three clauses are already satisfied, whereas the three other become reduced to 2-clauses. In these, x_j and x_k appear more frequently unnegated for $x_i = 1$, or negated if $x_i = 0$. These variables therefore have the tendency to take the same value like x_i, i.e., there is, on average, a ferromagnetic coupling between them. This tendency propagates via the two-spin interactions.

This coupling can be exploited by local algorithms. If we fix one spin, its neighbors become, on average, biased into the same direction, i.e., they tend to one of the two forced solutions. In walk-SAT experiments, we in fact find that the generated instances are solvable in polynomial time, with a peak resolution-time growing as $N^{2.3}$, see Figure 7.5. However, the complexity peak is not at the phase transition, but is quite close to the critical point of random 3-SAT. This is due to the fact that walk-SAT does not sample solutions according to the thermodynamic equilibrium distribution. Most probably it hits solutions with small magnetization, i.e., closer to the starting point (see Figure 7.4). For $N \to \infty$, this magnetization stays zero even after the ferromagnetic transition. Indeed, if we restrict the statistical mechanics analysis to zero magnetization, we find an exponential number of solutions also beyond $\alpha = 3.74$. More interestingly, this number coincides with the solution number of random 3-SAT, the latter jumps to 0 at $\alpha \simeq 4.27$ [22]. So, approaching this point, walk-SAT is no longer able to find unmagnetized solutions for model 1/6, and it has to go to magnetized assignments, giving rise to the resolution-time peak.

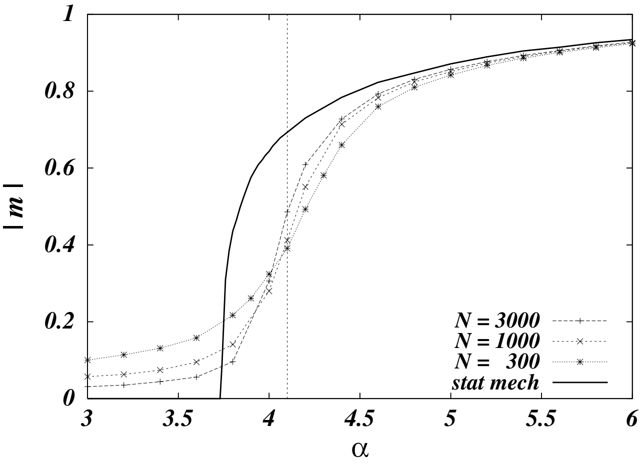


Figure 7.4: Magnetization of the first walk-SAT solution in model 1/6. Due to the (average) spin-flip symmetry, we plot the average of $|m|$. For large N, the magnetization stays zero up to $\alpha \simeq 4.1$. The full curve shows the thermodynamic average, which stays well above the asymptotic walk-SAT result.

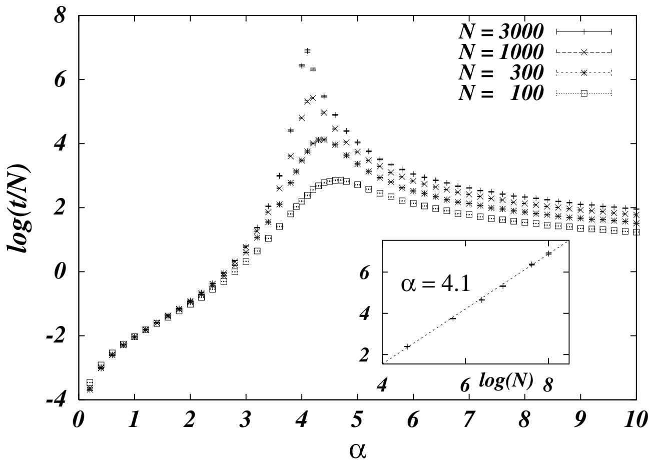


Figure 7.5: Typical walk-SAT complexity for model 1/6. We show the average value of $\log(t/N)$. We find a clear data collapse for small α in the linear regime, $t \propto N$. The complexity peak at $\alpha \simeq 4.1$ grows polynomially as shown in the inset. The slope of the straight line in the inset is 1.3.

7.3.2.2 Model 1/4

Going to the other extreme case of an unbiased generator, we have $p_0 = 1/4$ (*model 1/4*). This results in $p_1 = 0,\ p_2 = 1/4$, and only clauses with zero or two negated variables are allowed. The properties of the model change dramatically compared to model 1/6. First, not only the local fields H_i but also the two-spin interactions T_{ij} vanish on average, so they cannot be exploited in local algorithms. Only the three-spin interactions J_{ijk} have a non-zero mean. Fixing one spin, therefore, does not induce any exploitable information on the neighbors. One has to fix at least two spins in one clause, and consequently this information does not propagate on the graph.

Looking to the statistical-mechanical properties of model 1/4, we find no difference at all to random 3-SAT, for arbitrary values of $\alpha < \alpha_c \simeq 4.27$. Both models have the same solution entropy and they show the same clustering transition at $\alpha = 3.92$. Beyond this transition, the cluster containing the forced solution $\vec{x}^{(0)}$ is statistically equivalent to all other solution clusters. Since the latter are exponentially frequent, a typical solution found by any algorithm will be outside the forced cluster with probability close to one. The statistical ground-state properties of random 3-SAT and model 1/4 are distinct only beyond $\alpha_c = 4.27$. Whereas random 3-SAT becomes unsatisfiable, i.e., all solution clusters disappear at the SAT/UNSAT transition, model 1/4 still stays satisfiable. It undergoes, however, a first-order ferromagnetic transition at α_c. Only one solution cluster remains, namely the one containing the forced solution. The backbone, i.e. the number of completely frozen spins, jumps from zero to about 94% of all variables, and so does the magnetization of solutions, see Figure 7.6. This sudden appearance of the ferromagnetic phase poses serious problems to local algorithms which therefore require exponential solution times, cf. Figure 7.7, with a complexity peak close to the transition point. Note also that the solvable system sizes break down drastically: being about 10^4 for model 1/6, they decrease to slightly above 100 for model 1/4.

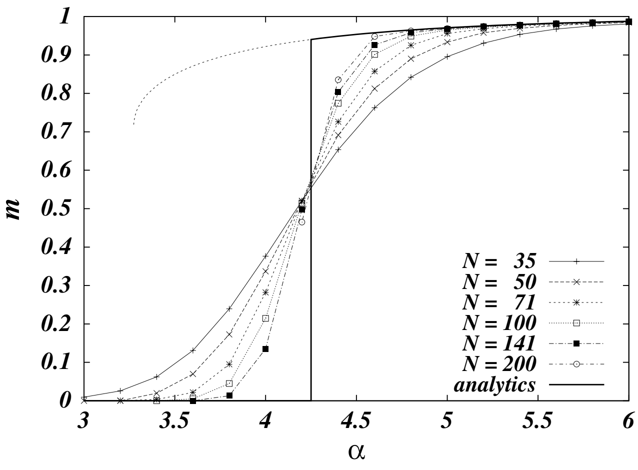


Figure 7.6: Average magnetization of solutions of model 1/4, obtained with a complete algorithm. There, the magnetization equals the backbone size. The finite-size curves cross at $\alpha \simeq 4.25$, and tend to the analytical prediction. The dotted continuation of the analytical line gives the globally unstable ferromagnetic solution, starting at the spinodal point.

So we could be tempted to consider model 1/4 as a generator of really hard, but satisfiable problem instances, but model 1/4 is a very peculiar case. It can always be solved in polynomial time using a *global* algorithm. Indeed, one can unambiguously add three clauses to every existing one, namely the other clauses allowed in model 1/4, without losing the satisfiability of the enlarged formula. In the enlarged formula, *every* clause is thus replaced by groups of four in the same form $(x_i \vee x_j \vee x_k) \wedge (x_i \vee \bar{x}_j \vee \bar{x}_k) \wedge (\bar{x}_i \vee x_j \vee \bar{x}_k) \wedge (\bar{x}_i \vee \bar{x}_j \vee x_k) = (x_i \oplus x_j \oplus x_k)$ with $\oplus$ being the logical XOR operator. The completed formula becomes a sample of random satisfiable 3-XOR-SAT (also known as hyper-SAT [25]) [8, 19]. It can be mapped to a system of αN linear equations modulo 2 by representing each clause via

$$(x_i \oplus x_j \oplus x_k) = \text{TRUE} \qquad \leftrightarrow \qquad x_i + x_j + x_k = 1 \pmod 2 \tag{7.11}$$

where on the right-hand site the 0/1-representation of the truth values of x_i is understood in terms of numbers. These linear equations can be solved in polynomial time growing as $\mathcal{O}(N^3)$.

7.3.2.3 Where the Really Hard Instances Are

This completion algorithm immediately breaks down if we choose $p_0 \neq 1/4$. Indeed, whenever one tries to map the general formula into a completed one, the presence of all three types of clause forces it into a *frustrated* 3-XOR-SAT formula, which undergoes a SAT/UNSAT transition at $\alpha = 0.918$ [25], well below the region of our interest. So the mapping is of no use for $p_0 \neq 1/4$. In this case, any 3-SAT instance with solution $\vec{x}^{(0)}$ (and thus any solvable one using local gauges) can be generated with non-zero probability. The worst case is thus included in the presented generator, and there cannot be any polynomial solver if P≠NP.

Once we use $p_0 > 0$, the situation already changes with respect to model 1/6. The ferromagnetic transition becomes first order, as can be seen best by the existence of a metastable solution for $P(m)$. The transition point moves continuously towards the random 3-SAT threshold α_c, and the computational complexity increases with p_0. Still, for $p_0 \leq 0.077$, the ferromagnetic phase arises without backbone and solutions can be found more easily.

In the region $0.077 \leq p_0 < 1/4$, the first-order transition is more pronounced. The system jumps at $\alpha \simeq 4.27$ from a paramagnetic phase to a ferromagnetic one, with the discontinuous appearance of a backbone. For $p_0 \simeq 0.077$, the backbone size at the threshold is about $0.72N$, and it goes up to $0.94N$ for $p_0 = 1/4$ (see Figure 7.6). We conjecture that the ferromagnetic critical point in these models coincides with the SAT/UNSAT threshold in random 3-SAT, since the topological structures giving rise to ferromagnetism in the former induce frustration and thus unsatisfiability in the latter.

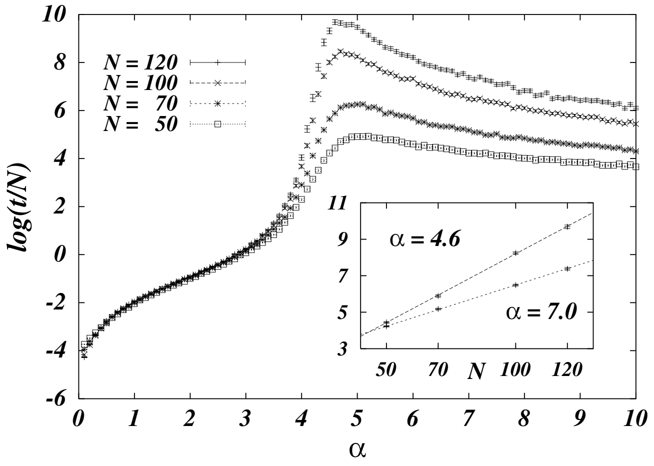


Figure 7.7: Typical walk-SAT complexity for model 1/4. The complexity peak is much more pronounced than in Figure 7.5, cf. e.g., the reachable system sizes. The inset shows the exponential resolution-time scaling near the peak ($\alpha = 4.6$) and deep inside the ferromagnetic phase ($\alpha = 7.0$). The slopes of the straight lines are 0.075 and 0.04 respectively.

In the following table we summarize the main results for the investigated combinations of p_0, p_1 and p_2. Where only p_0 is reported, $p_{1,2}$ are given by (7.11), i.e. the local fields are, on average, balanced. We show the location α_c and order of the ferromagnetic phase transition, together with the point α_{ws} of maximal walk-SAT complexity and the maximal system-size scaling (P/EXP). For comparison, we have added the corresponding data for random 3-SAT.

Note that the polynomial time-complexity of model 1/4 is accidental and is due to the existence of a global algorithm, whereas the walk-SAT peak grows exponentially with N. To

Model	α_c (order, type)	α_{ws}	Complexity
$p_{0,1,2} = 1/7$	NO	5.10	P
$p_0 = 0,\ p_{1,2} = 1/6$	3.74 (2nd, ferro)	4.10	P
$p_0 \in [0.077, 1/4)$	4.27 (1st, ferro)	4.27	EXP
$p_{0,2} = 1/4,\ p_1 = 0$	4.27 (1st, ferro)	4.27	P
Random 3-SAT	4.27 (SAT/UNSAT)	4.27	EXP

corroborate this picture, we also performed simulated annealing experiments. We easily find solutions in models 1/7 and 1/6, but get stuck in the vicinity of model 1/4.

7.4 Conclusion

As a conclusion, we have first given an overview of the SAT/UNSAT transition in random 3-SAT formulas and the insight gained by using tools from statistical mechanics. Relying on these ideas, we have presented an efficient generator of hard, but satisfiable problem instances, by hiding one forced solution in a multitude of metastable states. Within the class of unbiased generators proposed, we conjecture the hardest instances to be generated with p_0-values close to 1/4. This conjecture is supported by numerical tests with various complete and incomplete SAT-solvers, including walk-SAT, simulated annealing and the Davis–Putnam backtracking algorithm. Resolution times are clearly found to be exponential in all of the ferromagnetic phase ($\alpha > 4.27$). Moreover we checked that resolution times in the paramagnetic phase ($\alpha < 4.27$) coincide, up to finite-size effects, with those of random 3-SAT.

We have also seen that a naive generator, namely model 1/7, could be ruled out immediately because algorithms could easily exploit local structures being correlated to the forced solution. In the case of model 1/7, the simplest such structure was given by local fields resulting from an unbalance in the appearance of unnegated and negated variables. Being random variables, these fields point on average in the direction of the forced solution – which consequently was not well-hidden. Balancing these fields in models 1/6 and 1/4 removes this simple kind of local structure.

This is exactly the main idea behind a further optimization of the presented generator. The less local structure a formula has, the less it can be exploited by any solver. Still, our generator has some local structure, e.g., the number of appearances of single variables in the clauses is distributed according to a Poissonian of mean 3α. Performing some numerical experiments, one can see that suppressing these fluctuations further increases the computational hardness of the generator. On the other hand, one has to keep in mind that the proposed generator is extremely fast and therefore well-suited for practical solver tests.

A last remark concerns the generation of hard, but solvable SAT instances with clause lengths exceeding 3. In the more general case of K-SAT, i.e., where all clauses contain exactly K variables, the corresponding statistical-mechanics Hamiltonians contains interaction terms ranging from external fields over two-spin interactions up to K-spin interactions. In this case, the model corresponding to model 1/6, i.e., the one where two opposite solutions are forced, contains also more-spin interactions and is therefore hard [3]. On the other hand, we again conjecture that the hardest formulas are those where the K-spin interactions dominate all lower ones, and the selection weights for different clause types can be optimized accordingly.

Acknowledgments

I am grateful to W. Barthel, A.K. Hartmann, M. Leone, F. Ricci-Tersenghi, and R. Zecchina for discussion concerning the presented work and the manuscript. Special thanks to W. Barthel for supplying most of the figures.

References

[1] D. Achlioptas, *Lower Bounds for Random 3-SAT via Differential Equations*, Theor. Comp. Sci. **265**, 159 (2001).

[2] D. Achlioptas, C. Gomes, H. Kautz, and B. Selman, *Generating Satisfiable Problem Instances*, Proc. AAAI-00 (2000); H. Kautz, Y. Ruan, D. Achlioptas, C. Gomes, B. Selman, and M. Stickel, *Balance and Filtering in Structured Satisfiable Problems*, Proc. IJCAI-01, 351 (2001).

[3] D. Achlioptas, H. Jia, and C. Moore, preprint (2003).

[4] Artificial Intelligence **81**, special issue (1996).

[5] Y. Asahiro, K. Iwama, and E. Miyano, *Random Generation of Test Instances with Controlled Attributes*, in *Cliques, Coloring, and Satisfiability*, ed. by D.S. Johnson and M.A. Trick, DIMACS series vol. 26 (AMS, Providence, 1996).

[6] W. Barthel, A.K. Hartmann, M. Leone, F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, *Hiding Solutions in Random Satisfiability Problems: A Statistical Mechanics Approach*, Phys. Rev. Lett. **88**, 188701 (2002).

[7] G. Biroli, R. Monasson, and M. Weigt, *A Variational Description of the Ground State Structure in Random Satisfiability Problems*, Eur. Phys. J. B **14**, 551 (2000).

[8] S. Cocco, O. Dubois, J. Mandler, and R. Monasson, *Rigorous Decimation-based Construction of Ground Pure States for Spin Glass Models on Random Lattices*, Phys. Rev. Lett. **90**, 047205 (2003).

[9] S.A. Cook, *The Complexity of Theorem-proving Procedures*, Proc. 3rd ACM Symp. on the Theory of Computing, 151 (Association for Computer Machinery, New York 1971).

[10] O. Dubois, *Upper Bounds on the Satisfiability Threshold*, Theor. Comp. Sci. **265**, 187 (2001).

[11] J. Franco, *Results Related to Threshold Phenomena Research in Satisfiability: Lower Bounds*, Theor. Comp. Sci. **265**, 147 (2001).

[12] E. Friedgut, *Sharp Thresholds of Graph Properties, and the k-sat Problem*, J. Amer. Math. Soc. **12**, 1017 (1999).

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability* (Freeman, New York, 1979).

[14] H. Hoos and T. Stützle, *Local Search Algorithms for SAT: An Empirical Evaluation*, J. Autom. Res. **24**, 421 (2000).

[15] S. Horie and O. Watanabe, *Hard Instance Generation for SAT*, Lecture Notes in Computer Science **1350**, 22 (1996).

[16] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, *Optimization by Simmulated Annealing*, Science **220**, 671 (1983).

[17] D.L. Mammen and T. Hogg, *A New Look at the Easy-Hard-Easy Pattern of Combinatorial Search Difficulty*, J. Art. Int. Res. **7**, 47 (1997).

[18] M. Mézard, G. Parisi, and R. Zecchina, *Analytic and Algorithmic Solution of Random Satisfiability Problems*, Science (Washington DC) **297**, 812 (2002).

[19] M. Mézard, F. Ricci-Tersenghi, and R. Zecchina, *Alternative Solutions to Diluted p-spin Models and XORSAT Problems*, J. Stat. Phys. **111**, 505 (2003).

[20] M. Mézard and R. Zecchina, *The Random K-satisfiability Problem: From an Analytic Solution to an Efficient Algorithm*, Phys. Rev. E **66**, 056126 (2002).

[21] D. Mitchell, B. Selman, and H. Levesque, *Hard and Easy Distributions of SAT Problems*, Proc. AAAI-92, 459 (1992).

[22] R. Monasson and R. Zecchina, *Entropy of the K-satisfiability Problem*, Phys. Rev. Lett. **76**, 3881 (1996); *Statistical mechanics of the random K-SAT model*, Phys. Rev. E **56**, 1357 (1997).

[23] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Determining Computational Complexity from Characteristic 'Phase Transitions'*, Nature (London) **400**, 133 (1999).

[24] M. Motoki and R. Uehara, *Unique Solution Instance Generation for the 3-Satisfiability (3SAT) Problem*, Technical Report TR-C129 (Tokyo Inst. of Technology, 1999).

[25] F. Ricci-Tersenghi, M. Weigt, R. Zecchina, *The Simplest Random Satisfiability Problem*, Phys. Rev. E **63**, 026702 (2001).

[26] B. Selman and S. Kirkpatrick, *Critical Behavior in the Computational Cost of Satisfiability Testing*, Science (Washington DC) **264**, 1297 (1994).

[27] SATLIB, `http://www.satlib.org`.

[28] B. Selman, H. Kautz, and B. Cohen, in *Cliques, Coloring, and Satisfiability*, ed. by D.S. Johnson and M.A. Trick, DIMACS series vol. 26 (AMS, Providence, 1996).

[29] Theor. Comp. Sci. **265**, special issue (2001).

[30] D. Welsh, *Codes and Cryptography* (Clarendon Press, 1988).

8 Analysis of Backtracking Procedures for Random Decision Problems

Simona Cocco, Liat Ein-Dor, Rémi Monasson

Complete search algorithms are procedures capable of deciding whether or not a decision problem has a solution. Among these are the ubiquitous backtracking-like algorithms, where a decision is reached through a sequence of trials and errors. Analysis of the performances of these procedures is difficult but can be done, to some extent, using statistical physics ideas and techniques. Here, this approach is presented and illustrated on the random Satisfiability (SAT) and Graph Coloring (COL) problems.

8.1 Introduction

The wide variety of practical problems that can be mapped onto NP-complete problems, together with the challenge in finding an answer to one of the most important open questions in theoretical computer science, 'Does $\mathrm{NP} = \mathrm{P}$?', have led to intensive studies over the past decades. Despite intense efforts, the worst-case running times of all currently known algorithms grow exponentially with the size of the inputs to these problems. However, NP-complete problems are not always hard. They might even be easy to solve on average [10, 25, 44], i.e., when their resolution complexity is measured with respect to some underlying probability distribution of instances. This 'average-case' behavior depends, of course, on the input distribution.

The average-case analysis of algorithms is a well defined branch of theoretical computer science, with close connections to probability theory and combinatorics [33, 42]. It was recently suggested that these connections could extend up to out-of-equilibrium statistical physics [15]. Indeed, scientists working in the fields of the analysis of algorithms and of statistical physics have common goals. They all aim to understand to the properties of dynamical processes involving an exponentially large (in the size of the input) set of configurations. The differences between the two disciplines mainly lie in the methods of investigation. In contrast to theoretical computer scientists, physicists rarely provide exact results. But concepts and tools developed over the past decades may prove useful in tackling the study of complex algorithms that mathematical approaches have so far failed to solve.

There is a huge variety of algorithms designed to cope with combinatorial problems [30]. Briefly speaking, one can distinguish between complete and incomplete search procedures. The latter are capable of finding solutions quickly but are unable to prove their absence[1] while

[1] This statement is true from a deterministic point of view, but incomplete procedures may be able to disprove the existence of the solution in a probabilistic manner, see [40].

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

the former will always output the right answer (existence or absence of solution) however long it takes to find it. While incomplete search procedures are sometimes related to dynamical processes inspired from physics, e.g., simulated annealing, the operation of complete procedures rely on very different principles, with no *a priori* physical motivations. In addition, their study has a long and rich history in theoretical computer science [34]. This makes the analysis of complete algorithms with theoretical physics tools all the more fascinating. In this chapter, we shall focus upon an ubiquitous complete procedure, the so-called Davis–Putnam–Logemann–Loveland (DPLL) algorithm [19,30], at the root of branch-and-bound procedures widely used in 'practical' optimization. The reader is referred to [15] and references therein for a review on recent progress in the analysis of incomplete procedures from the physics point of view.

Virtually all decision problems can be solved with DPLL. Two widely known examples of such problems which we shall focus on throughout this chapter are:

- *Satisfiability of Boolean constraints (SAT).* In K-SAT one is given an instance, that is, a set of M logical constraints (clauses) among N boolean variables, and one wants to know if there exists a truth assignment for the variables which fulfill all the constraints. Each clause is the logical OR of K literals, a literal being one of the N variables or its negation, e.g., $(x_1 \vee x_4 \vee \overline{x_5})$ for 3-SAT.

- *Coloring of graphs (COL).* An input instance of the K-COL decision problem consists of a graph G. The problem involves finding a mapping from the set of vertices to the set of K colors such that no edge links vertices with the same color, or else proving there are none.

We now illustrate the operation of DPLL on an input F of the K-SAT problem defined over a set V of variables. Call Γ_j the set of clauses including j variables, L the set of literals, U the set of unassigned variables, and $depth$ the number of further backtrackings available to DPLL. Initially, $\Gamma_j = \emptyset$ for all $j < K$, $\Gamma_K = F$, $L = \emptyset$, $U = V$, and $depth = 0$. The procedure is defined as follows:

```
algorithm DPLL[$\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_K; L; U; depth$]
begin
  for $j = 1$ to $K$ {updating of clause sets};
  begin
    for all $c = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_j \in \Gamma_j$;
    begin
      if $\exists i \in [1; j]$ such that $\ell_i \in L$ then {clause to be removed?}
      begin
        $\Gamma_j := \Gamma_j \setminus \{c\}$;
      end;
      if $\exists i \in [1; j]$ such that $\bar{\ell}_i \in L$ then {clause to be reduced?}
      begin
        $\Gamma_j := \Gamma_j \setminus \{c\}$;
        $\Gamma_{j-1} := \Gamma_{j-1} \cup \{c \setminus \ell_i\}$;
      end;
    end;
  end;
```

```
    **if** $\Gamma_0 = \Gamma_1 = \Gamma_2 = \ldots = \Gamma_K = \emptyset$ **then** {all clauses are satisfied?}
    **begin**
        **print** ‘SATISFIABLE (by literals in $L$)’ ;
        **stop**
    **end**;
    **if** $\Gamma_0 = \emptyset$ **then**
    **begin** {there is no violated clause}
        **if** $\Gamma_1 \neq \emptyset$ **then**
        **begin** {there is some unitary clause i.e. with a unique literal}
            $\ell$ := a literal (unitary clause) randomly chosen in $\Gamma_1$;
            $x$ := the variable associated to literal $\ell$;
            **DPLL**$[\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_K; L \cup \{\ell\}; U \setminus \{x\}; depth]$;
        **end**;
        **else** {there is no unitary clause};
            $x$ := a variable chosen in $U$ according to some heuristic rule;
            $\ell$ := a literal equal to $x$, or to $\bar{x}$ depending on some heuristic rule;
            **DPLL**$[\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_K; L \cup \{\ell\}; U \setminus \{x\}; depth + 1]$;
            **DPLL**$[\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_K; L \cup \{\bar{\ell}\}; U \setminus \{x\}; depth]$;
        **end**;
    **end**
    **else** {there is some violated clause};
        **if** $depth = 0$ **then**
        **begin** {further backtracking is impossible}
            **print** ‘UNSATISFIABLE’;
            **stop**;
        **end**;
    **end**;
**end**
```

The first part of the algorithm consists of updating the sets of clauses after a variable has been assigned at a previous step, e.g., $x = T$. Some clauses are satisfied, e.g., $c = x \vee y \vee z$ and eliminated, other are reduced, e.g., $c = \bar{x} \vee y \vee z \rightarrow c = y \vee z$. The procedure is such that, if some clauses include one variable only, e.g., $c = y$, the corresponding variable is automatically fixed to satisfy the clause ($y = T$). This unitary propagation is repeated up to the exhaustion of all unit clauses. If there is no unit clause ($\Gamma_1 = \emptyset$), a heuristic rule indicates which variable should be selected and which value it should be assigned to. In the presence of a contradiction, that is, two opposite unitary clauses, e.g., $c = y$, $c' = \bar{y}$, DPLL backtracks to the last assignment of literal and tries the opposite value for the attached variable. At the end of the process, a solution is found if no clauses are left, or no backtracking is possible and a proof of unsatisfiability is obtained.

It is convenient to represent the history of the search process, that is, the sequence of trials and errors generated by DPLL by a search tree. Examples of search trees are given in Figure 8.1. Nodes in the tree are attached to the assignment of variables, while edges represent the logical consequences (elimination of satisfied constraints, simplification of other constraints) resulting from these assignments. Branch extremities are marked with contradictions C, or by a solution S. A good computer-independent measure of the complexity of resolution is the size of the search tree generated by DPLL. This search tree varies with the input of the problem under consideration and the sequence of assignments made by the search procedure.

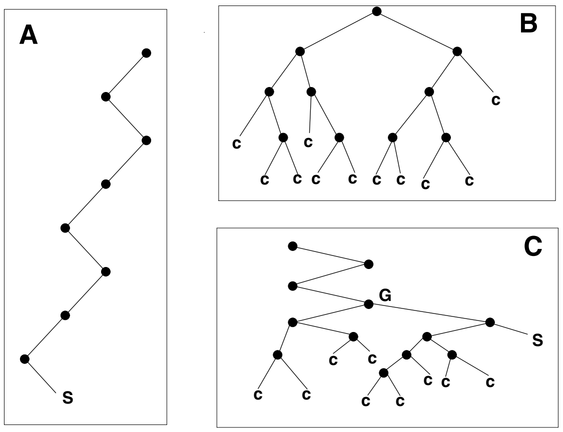


Figure 8.1: Types of search trees generated by the DPLL solving procedure on K-SAT. (A) *Simple branch:* the algorithm easily finds a solution without ever backtracking. (B) *Dense tree:* in the absence of solution, DPLL builds a tree, including many branches ending with contradictory leaves, before stopping. (C) *Mixed case, branch + tree:* if many contradictions arise before reaching a solution, the resulting search tree can be decomposed into a single branch followed by a dense tree. G is the highest node in the tree reached by DPLL through backtracking.

Analysis of the average-case performance of DPLL, that is, of the average of the search tree size for a given decision problem, e.g., SAT, requires a definition of its input distribution. Such distributions are usually unrealistic compared to structured instances from the real world, but are simple enough to allow for some analytical treatment. Current popular input distributions are:

- *Random K-SAT* is the K-SAT problem supplied with a distribution of inputs uniform over all instances having fixed values of N and M. The limit of interest is $N, M \to \infty$ at fixed ratio $\alpha = M/N$ of clauses per variable [17, 27, 37].

- *Random K-COL* inputs are random graphs G after Erdős-Rényi, i.e., drawn with uniform probability among all the graphs having N vertices and E edges. The limit of interest is $N, E \to \infty$ at fixed ratio $c = 2E/N$ of edges per vertex [4, 5, 18].

Random SAT and COL both exhibit a phase transition phenomenon [23]. For small values of their control parameter π ($= \alpha$ or c), and for large input sizes, the answer to the decision problem (existence of an assignment satisfying the constraints, or of a proper coloring) is almost definitely yes. This holds as long as π remains smaller than a (K dependent) critical value π_C called threshold. Above threshold, the answer is no with high probability. The behavior of DPLL is, to some extent, related to this phase transition as shown in Figure 8.2.

Three regimes can be identified:

1. For low control parameter $\pi < \pi_L(< \pi_C)$, the answer is almost definitely yes. There is a finite probability that a solution is found by DPLL with essentially no backtracking [8, 9, 24]. Search trees look like Figure 8.1(A). Their size grows polynomially (linearly) with the input size N (number of variables for SAT, of vertices for COL). In the following we will refer to this regime as low SAT phase.

2. For $\pi > \pi_C$, that is, when the answer is no with high probability, proving the absence of a solution requires the building up of a search tree like the one in Figure 8.1(B), whose size is exponentially large (in N) [11]. This regime will be called the UNSAT phase.

3. In the intermediate regime i.e. $\pi_L < \pi < \pi_C$, there are solutions solutions but finding them is hard (Figure 8.1(C)), and requires exponential effort [3, 12, 13]. This regime will be referred to as the upper SAT phase.

Notice that the location π_L of the easy/hard crossover depends upon the algorithm, in contrast to π_C. The purpose of this chapter is the presentation of techniques allowing the reader to reach a quantitative understanding of Figure 8.2.

This text is organized as follows. For the sake of simplicity, we focus in the first sections on the SAT problem. We start by introducing the useful notions of phase diagram, search trajectories, etc., in Section 8.2, and use these to analyze the search process in the low SAT phase where backtracking is essentially irrelevant. We turn to the opposite case of the UNSAT phase in Section 8.3, and develop tools necessary for the study of the action of DPLL in the presence of massive backtracking. Section 8.4 is devoted to the analysis of the upper SAT phase, that is, of the average-case complexity and large deviations from the latter. We show how all techniques presented in those Sections for SAT can be applied to COL in Section 8.5. Conclusions are presented in Section 8.6. Notice that, though most of the material presented in this chapter was previously published by the authors in various articles [15, 16, 20], Section 8.3.4 contains a new (and exact!) solution of the partial differential equation modeling the search tree growth.

8.2 Phase Diagram, Search Trajectories and the Easy SAT Phase

In this section, we present some useful concepts for the understanding of DPLL dynamics of search, and we apply them to investigate the low-ratio α regime, where a solution is rapidly found and the search tree essentially reduces to a single branch as shown in Figure 8.1(A). We start with some general comments on the dynamics induced by DPLL, and introduce the notion of mixed 2+p-SAT instance distribution. These concepts are made more precise and illustrated through the analysis of the single-branch trajectory, strongly inspired from some previous works by Chao and Franco [9] which the reader is referred to for more details. In the last Section 8.2.4, our numerical and analytical results for the solving complexity in the polynomial regime are presented.

Hereafter, the ratio of the 3-SAT instance to be solved will be denoted by α_0.

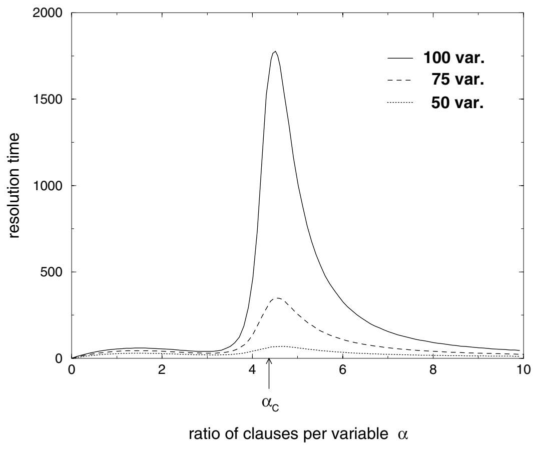


Figure 8.2: Resolution time of 3-SAT random instances by DPLL as a function of the ratio of clauses per variable α and for three different input sizes. Data correspond to the median resolution time of 10 000 instances; the average time may be somewhat larger due to the presence of rare, exceptionally hard instances. The computational complexity is maximal at α_c. It is exponential in the vicinity of the threshold and in the unsatisfiable phase, but less and less as α increases. A similar curve is obtained for random COL with α substituted with the ratio c of edges per vertex.

8.2.1 Overview of Concepts Useful to DPLL Analysis

The action of DPLL on an instance of 3-SAT causes changes to the overall numbers of variables and clauses, and thus of the ratio α. Furthermore, DPLL reduces some 3-clauses to 2-clauses. A mixed 2+p-SAT distribution, where p is the fraction of 3-clauses, can be used to model what remains of the input instance at a node of the search tree. Using experiments and methods from statistical mechanics [38], the threshold line $\alpha_C(p)$, separating SAT from UNSAT phases, may be estimated with the results shown in Figure 8.3. For $p \leq p_0 = 2/5$, i.e., to the left of point T, the threshold line is given by $\alpha_C(p) = 1/(1-p)$, as rigorously confirmed by [1], and coincides with the upper bound for the satisfaction of 2-clauses. Above p_0, no exact value for $\alpha_C(p)$ is known.

The phase diagram of 2+p-SAT is the natural space in which DPLL dynamics takes place. An input 3-SAT instance with ratio α shows up on the right vertical boundary of Figure 8.3 as a point of coordinates $(p = 1, \alpha)$. Under the action of DPLL, the representative point moves aside from the 3-SAT axis and follows a trajectory, very much like real-space renormalization. This trajectory obviously depends on the heuristic of the split followed by DPLL. Possible simple heuristics are [8, 9],

- *Unit-Clause (UC):* randomly pick up a literal among a unit clause if any, or any unset variable otherwise.

- *Generalized Unit-Clause (GUC):* randomly pick up a literal among the shortest available clauses.

- *Short Clause With Majority (SCWM):* randomly pick up a literal among unit clauses if any; otherwise randomly pick up an unset variable v, count the numbers of occurrences $\ell, \bar{\ell}$ of $v, \bar{v}$ in 3-clauses, and choose v (respectively $\bar{v}$) if $\ell > \bar{\ell}$ (resp. $\ell < \bar{\ell}$). When $\ell = \bar{\ell}$, v and $\bar{v}$ are equally likely to be chosen.

Rigorous mathematical analysis, undertaken to provide bounds to the critical threshold α_C, have so far been restricted to the action of DPLL prior to any backtracking, that is, to the first descent of the algorithm in the search tree[2]. The corresponding search branch is drawn on Figure 8.1(A). These studies rely on the two following facts.

First, the representative point of the instance treated by DPLL does not "leave" the 2+p-SAT phase diagram. In other words, the instance is, at any stage of the search process, uniformly distributed from the 2+p-SAT distribution conditioned to its clause per variable ratio α and fraction of 3-clauses p. This assumption is not true for all heuristics of split, but holds for the above examples (*UC*, *GUC*, *SCWM*) [8]. Analysis of more sophisticated heuristics requires the handling of more complex instance distributions [32].

Secondly, the trajectory followed by an instance in the course of resolution is a stochastic object, due to the randomness of the instance and of the assignments done by DPLL. In the large size limit ($N \to \infty$), this trajectory gets concentrated around its average locus in the 2+p-SAT phase diagram. This concentration phenomenon results from general properties of Markov chains [2, 46].

8.2.2 Clause Populations: Flows, Averages and Fluctuations

As pointed out above, under the action of DPLL, some clauses are eliminated while other ones are reduced. Let us call $C_j(T)$ the number of clauses of length j (including j variables), once T variables have been assigned by the solving procedure. T will be called hereafter "time", not to be confused with the computational effort necessary to solve a given instance. At time $T = 0$, we obviously have $C_3(0) = \alpha_0 N$, $C_2(0) = C_1(0) = 0$. As Boolean variables are assigned, T increases and clauses of length one or two are produced. A sketchy picture of DPLL dynamics at some instant T is proposed in Figure 8.4.

We call e_1, e_2, e_3 and w_2, w_1 the flows of clauses represented in Figure 8.4 when time increases from T to $T+1$, that is, when one more variable is chosen by DPLL after T have already been assigned. The evolution equations for the three populations of clauses read,

$$\begin{aligned} C_3(T+1) &= C_3(T) - e_3(T) - w_2(T) \\ C_2(T+1) &= C_2(T) - e_2(T) + w_2(T) - w_1(T) \\ C_1(T+1) &= C_1(T) - e_1(T) + w_1(T) \; . \end{aligned} \tag{8.1}$$

The flows e_j and w_j are, of course, random variables that depend on the instance under consideration at time T, and on the choice of the variable (label and value) done by DPLL. For a single descent, i.e., in the absence of backtracking, and for the GUC heuristic, the evolution process (8.1) is Markovian and unbiased. The distribution of instances generated by DPLL at time T is uniform over the set of all the instances having $C_j(T)$ clauses of length $j = 1, 2, 3$ and drawn from a set of $N - T$ variables [9].

[2] The analysis of [24] however includes a very limited version of backtracking, see Section 8.2.2.

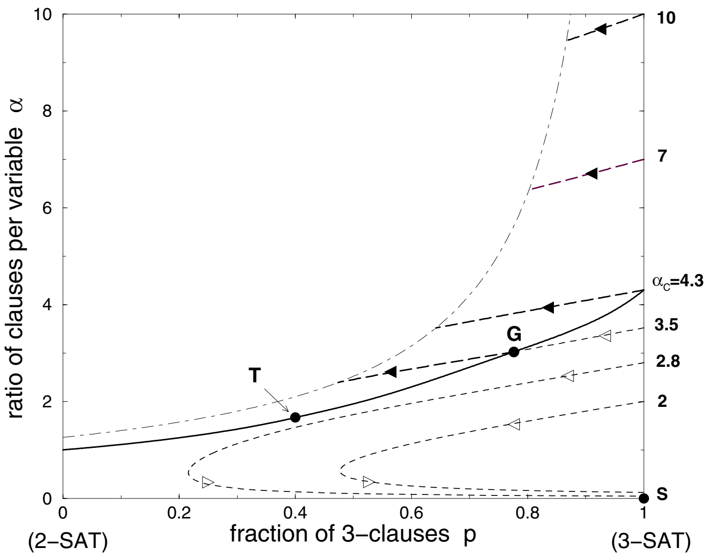


Figure 8.3: Phase diagram of 2+p-SAT and dynamical trajectories of DPLL. The threshold line $\alpha_C(p)$ (bold full line) separates SAT (lower part of the plane) from UNSAT (upper part) phases. Extremities lie on the vertical 2-SAT (left) and 3-SAT (right) axis at coordinates $(p = 0, \alpha_C = 1)$ and $(p = 1, \alpha_C \simeq 4.3)$ respectively. Departure points for DPLL trajectories are located on the 3-SAT vertical axis and the corresponding values of α are explicitly given. Dashed curves represent tree trajectories in the UNSAT region (thick lines, black arrows) and branch trajectories in the SAT phase (thin lines, empty arrows). Arrows indicate the direction of "motion" along trajectories parameterized by the fraction t of variables set by DPLL. For small ratios $\alpha < \alpha_L$, branch trajectories remain confined in the SAT phase, end in S of coordinates $(1, 0)$, where a solution is found. At α_L ($\simeq 3.003$ for the GUC heuristic), the single branch trajectory hits tangentially the threshold line in T of coordinates $(2/5, 5/3)$. In the intermediate range $\alpha_L < \alpha < \alpha_C$, the branch trajectory intersects the threshold line at some point G (which depends on α). A dense tree then grows in the UNSAT phase, as happens when 3-SAT departure ratios are above threshold $\alpha > \alpha_C \simeq 4.3$. The tree trajectory halts on the dot-dashed curve $\alpha \simeq 1.259/(1 - p)$ where the tree growth process stops. Once the dense tree is built, DPLL reaches the highest backtracking node in the search tree, that is, the first node when $\alpha > \alpha_C$, or node G for $\alpha_L < \alpha < \alpha_C$. In the latter case, a solution can be reached from a new descending branch while, in the former case, unsatisfiability is proven, see Figure 8.1.

As a result of the additivity of (8.1), some concentration phenomenon takes place in the large size limit. The numbers of clauses of lengths 2 and 3, *a priori* extensive in N, do not fluctuate too much,

$$C_j(T) = N\, c_j\left(\frac{T}{N}\right) + o(N) \qquad (j = 2, 3)\ . \tag{8.2}$$

where the c_j are the densities of clauses of length j averaged over the instance (quenched disorder) and the choices of variables ("thermal" disorder). In other words, the densities of 2- and 3-clauses are self-averaging quantities and we shall attempt to calculate their mean values only. Note that, in order to prevent the occurrence of contradictions, the number of unitary clauses must remain small and the density c_1 of unitary clauses has to vanish.

Formula (8.2) also illustrates another essential feature of the dynamics of clause populations. Two time scales are at play. The short time scale, of the order of unity, corresponds to the fast variations in the numbers of clauses $C_j(T)$ ($j = 1, 2, 3$). When time increases from T to $T + O(1)$ (with respect to the size N), all C_j's vary by $O(1)$ amounts. Consequently, the densities c_j of clauses, that is, their numbers divided by N, are changed by $O(1/N)$ only. The densities c_j evolve on a long time scale of the order of N and depend on the reduced time $t = T/N$ only.

Due to the concentration phenomenon underlined above, the densities $c_j(t)$ will evolve deterministically with the reduced time t. We shall see below how Chao and Franco calculated their values. On the short time scale, the relative numbers of clauses $D_j(T) = C_j(T) - Nc_j(T/N)$ fluctuate (with amplitude $\ll N$) and are stochastic variables. As above noted, the evolution process for these relative numbers of clauses is Markovian and the probability rates (master equation) are functions of slow variables only, i.e., of the reduced time t and of the densities c_2 and c_3. As a consequence, on intermediary time scales, much larger than unity and much smaller than N, the D_j may reach some stationary distribution that depend upon the slow variables.

This situation is best exemplified in the case $j = 1$ where $c_1(t) = 0$ as long as no contradiction occurs and $D_1(T) = C_1(T)$. Consider, for instance, a time delay $1 \ll \Delta T \ll N$, e.g., $\Delta T = \sqrt{N}$. For times T lying in between $T_0 = t\,N$ and $T_1 = T_0 + \Delta T = t\,N + \sqrt{N}$, the numbers of 2- and 3-clauses fluctuate but their densities are left unchanged and equal to $c_2(t)$ and $c_3(t)$. The average number of 1-clauses, called unitary clauses above, fluctuates and follows some master equation whose transition rates (from $C_1' = C_1(T)$ to $C_1 = C_1(T + 1)$) define a matrix $\mathbf{H}(C_1, C_1')$ and depend on t, c_2, c_3 only. $\mathbf{H}$ has a single eigenvector $\bar{\mu}(C_1)$ with eigenvalue unity, called equilibrium distribution, and other eigenvectors with smaller eigenvalues (in modulus). Therefore, at time T_1, C_1 has forgotten the "initial condition" $C_1(T_0)$ and is distributed according to the equilibrium distribution $\bar{\mu}(C_1)$ of the master equation. Calculation of the equilibrium distribution $\bar{\mu}(C_1)$ of unit clauses will be sketched in Section 8.2.4.

To sum up, the dynamical evolution of the clause populations may be seen as a slow and deterministic evolution of the clause densities on which are superimposed fast, small fluctuations. The equilibrium distribution of the latter adiabatically follows the slow trajectory.

8.2.3 Average-case Analysis in the Absence of Backtracking

In this section, we explain Chao and Franco's calculation of the densities of 2- and 3-clauses. Consider first the evolution equation (8.1) for the number of 3-clauses. This can be rewritten in terms of the average density c_3 of 3-clauses and of the reduced time t,

$$\frac{dc_3(t)}{dt} = -z_3(t) \, , \tag{8.3}$$

where $z_3 = \langle e_3 + w_2 \rangle$ denotes the averaged total outflow of 3-clauses (Section 8.2.2).

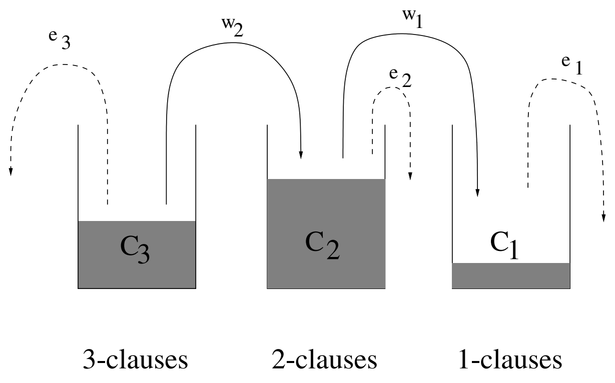


Figure 8.4: Schematic view of the dynamics of clauses. Clauses are sorted into three containers according to their lengths, i.e., the number of variables they include. Each time a variable is assigned by DPLL, clauses are modified, resulting in a dynamics of the container populations (lines with arrows). Dashed lines indicate the elimination of (satisfied) clauses of lengths 1, 2 or 3. Bold lines represent the reduction of 3-clauses into 2-clauses, or 2-clauses into 1-clauses. The flows of clauses are denoted by e_1, e_2, e_3 and w_2, w_1, respectively. A solution is found when all containers are empty. The level of the rightmost container coincides with the number of unitary clauses. If this level is low (i.e., $O(1)$), the probability that two contradictory clauses x and $\bar{x}$ are present in the container is vanishingly small. When the level is high (i.e., $O(\sqrt{N})$), contradictions will occur with high probability.

At some time step $T \to T+1$, 3-clauses are eliminated or reduced if and only if they contain the variable chosen by DPLL. Let us first suppose that the variable is chosen in some 1- or 2-clauses. A 3-clause will include this variable or its negation with probability $3/(N-T)$ and disappear with the same probability. Due to the uncorrelation of clauses, we obtain $z_3(t) = 3c_3(t)/(1-t)$. If the literal assigned by DPLL is chosen among some 3-clause, this expression for z_3 has to be increased by one (since this clause will necessarily be eliminated) in the large-N limit.

Let us call $\rho_j(t)$ the probability that a literal is chosen by DPLL in a clause of length j $(= 1, 2, 3)$. Note that the sum of these probabilities is smaller than or equal to one, since we are free to choose the literal irrespective of the clause content (see UC case below). Extending the above discussion to 2-clauses, we obtain

$$\begin{aligned} \frac{dc_3(t)}{dt} &= -\frac{3}{1-t}c_3(t) - \rho_3(t) \\ \frac{dc_2(t)}{dt} &= \frac{3}{2(1-t)}c_3(t) - \frac{2}{1-t}c_2(t) - \rho_2(t)\ . \end{aligned} \tag{8.4}$$

In order to solve the above set of coupled differential equations, we need to know the probabilities ρ_j. As we shall see below, the values of the ρ_j depend on the heuristic of choice followed by DPLL and explained in Section 8.2.1. The solutions of the differential Equations (8.4) will then be expressed in terms of the fraction p of 3-clauses and the ratio α of clauses per variable using the identities

$$p(t) = \frac{c_3(t)}{c_2(t)+c_3(t)} \,, \qquad \alpha(t) = \frac{c_2(t)+c_3(t)}{1-t} \,. \tag{8.5}$$

8.2.3.1 Case of the GUC Heuristic

When DPLL is launched, 2-clauses are created with an initial flow $\langle w_2(0)\rangle = 3\,\alpha_0/2$. Let us first suppose that $\alpha_0 \leq 2/3$, i.e., $w_2(0) \leq 1$. In other words, less than one 2-clause is created each time a variable is assigned. Since the GUC rule compels DPLL to look for literals in the smallest available clauses, 2-clauses are immediately removed just after creation and do not accumulate in their container ($c_2 = 0$). Unitary clauses are almost absent and we have

$$\rho_1(t) = 0 \,; \quad \rho_2(t) = \frac{3c_3(t)}{2(1-t)} \,; \quad \rho_3(t) = 1-\rho_2(t) \qquad (\alpha_0 < 2/3) \,. \tag{8.6}$$

The solutions of (8.4) with the initial condition $p(0) = 1, \alpha(0) = \alpha_0$ read

$$\begin{aligned} p(t) &= 1 \,, \\ \alpha(t) &= (\alpha_0+2)(1-t)^{1/2} - 2 \,. \end{aligned} \tag{8.7}$$

Solution (8.7) confirms that the instance never contains an extensive number of 2-clauses. At some final time t_{end}, depending on the initial ratio, $\alpha(t_{end})$ vanishes: no clause is left and a solution is found.

We now assume that $\alpha_0 > 2/3$, i.e., $\langle w_2(0)\rangle > 1$. In other words, more than one 2-clause is created each time a variable is assigned. 2-clauses now accumulate, and give rise to unitary clauses. Due to the GUC prescription, in the presence of 1- or 2-clauses, a literal is never chosen in a 3-clause. We show in Section 8.2.4 that the probability that there is no 1-clause at some stage of the procedure equals $1 - c_2(t)/(1-t)$. This probability coincides with $\rho_2(t)$ and, thus, we have

$$\rho_1(t) = \frac{c_2(t)}{1-t} \,; \quad \rho_2(t) = 1-\rho_1(t) \,; \quad \rho_3(t) = 0 \qquad (\alpha_0 > 2/3) \,, \tag{8.8}$$

as soon as $t > 0$. The solutions of (8.4) now read

$$\begin{aligned} p(t) &= \frac{4\alpha_0(1-t)^2}{\alpha_0(1-t)^2 + 3\alpha_0 + 4\ln(1-t)} \,, \\ \alpha(t) &= \frac{\alpha_0}{4}(1-t)^2 + \frac{3\alpha_0}{4} + \ln(1-t) \,. \end{aligned} \tag{8.9}$$

Solution (8.9) requires that the instance contains an extensive number of 2-clauses. This is true at small times since $p'(0) = 1/\alpha_0 - 3/2 < 0$. At some time $t^* > 0$, depending on the initial ratio, $p(t^*)$ reaches back unity: no 2-clause are left and hypothesis (8.8) breaks down. DPLL has therefore reduced the initial formula to a smaller 3-SAT instance with a ratio $\alpha^* = \alpha(t^*)$. It can be shown that $\alpha^* < 2/3$. Thus, as the dynamical process is Markovian, the further evolution of the instance reduces to the $\alpha_0 < 2/3$ case.

We show in Figure 8.3 the trajectories obtained for initial ratios $\alpha_0 = 0.6$, $\alpha_0 = 2$ and $\alpha_0 = 2.8$. When $\alpha_0 > 2/3$, the trajectory first heads to the left (creation of 2-clauses), and then reverses to the right (2-clause destruction results from splits) until reaching a point on the 3-SAT axis at small ratio $\alpha^*(< 2/3)$ without ever leaving the SAT region. Further action of DPLL leads to a rapid elimination of the remaining clauses and the trajectory ends up at the right lower corner S, where a solution is achieved (Section 8.2.3.1). As α_0 increases up to α_L, the trajectory gets closer and closer to the threshold line $\alpha_C(p)$. Finally, at $\alpha_L \simeq 3.003$, the trajectory touches the threshold curve tangentially at point T with coordinates $(p_T = 2/5, \alpha_T = 5/3)$. Note the identity $\alpha_T = 1/(1-p_T)$.

8.2.3.2 Case of UC and SCWM Heuristics

The above study can be extended to the other heuristics presented in Section 8.2.1. For UC and SCWM, the probability $\rho_3(t)$ that a variable is chosen from a 3-clause vanishes for all positive times. The set of ODEs (8.4) is thus entirely defined from the expression of the probability ρ_2 that a variable is set through splitting of a clause,

$$\rho_2(t) = \left[1 - \frac{c_2(t)}{1-t}\right] \, h(t) \; . \tag{8.10}$$

Function h depends upon the heuristic:

- $h_{UC}(t) = 0$;
- $h_{SCWM}(t) = 3a_3\, e^{-3a_3}\,(I_0(3a_3) + I_1(3a_3))/2$ where $a_3 \equiv c_3(t)/(1-t)$ and I_ℓ is the ℓ^{th} modified Bessel function.

The reader is referred to Ref. [2, 22] for additional information.

8.2.4 Occurrence of Contradictions and Polynomial SAT Phase

In this section, we compute the computational complexity in the range $0 \le \alpha_0 \le \alpha_L$ from the previous results. To avoid unnecessary repetitions, we specialize to the case of the GUC heuristic.

The trajectories obtained in Section 8.2.3 represent the deterministic evolution of the densities of 2- and 3-clauses when more and more variables are assigned. Below, we briefly present the calculation[3] of the distribution $\bar{\mu}(C_1, t)$ of the number C_1 of 1-clauses at reduced time t done by Frieze and Suen [24]. Call $\mathbf{H}(C_1, C_1')$ the probability that the number of unit-clauses goes from C_1' to C_1 once a variable is fixed by DPLL, see Section 8.2.2. In the limit of large size N, the entries of matrix $\mathbf{H}$ depend on the reduced time t and the average density c_2 of 2-clauses only,

$$\mathbf{H}(C_1, C_1') = \sum_{w_1 \ge 0} e^{-a_2} \frac{a_2^{w_1}}{w_1!} \, \delta_{C_1 - C_1' + w_1 - \sigma(C_1')} \tag{8.11}$$

where $a_2 \equiv c_2/(1-t)$, w_1 is the creation flow of unit-clauses represented in Figure 8.4, and $\sigma(C_1') = 1$ if $C_1' \ge 1$, 0 if $C_1' = 0$. The evolution matrix $\mathbf{H}$ ensures probability conservation

[3] This calculation, combined with the study of the large deviations of the densities of 2- and 3-clauses, is closely related to the more complex analysis of the UNSAT phase presented in Section 8.3.

since entries along any column sum up to one. $\bar{\mu}(C_1)$, the eigenvector associated to the largest eigenvalue (equal to unity), represents the stationary distribution of the number C_1 of unit-clauses. In terms of the generating function μ of $\bar{\mu}$ [21],

$$\mu(y_1) = \sum_{C_1 \geq 0} \bar{\mu}(C_1)\, e^{y_1 C_1} \ , \tag{8.12}$$

the eigenvalue equation reads

$$\mu(y_1) = (1 - a_2)\, \frac{e^{\nu(y_1)}\,(1 - e^{y_1})}{e^{\nu(y_1)} - 1} \ , \quad \text{where } \nu(y_1) \equiv -y_1 - a_2 + a_2\, e^{y_1} \ . \tag{8.13}$$

Sending $y_1 \to -\infty$ in (8.13) permits us to find the probability that there is no unitary clause, $\bar{\mu}(C_1 = 0) = 1 - a_2$. This probability gives the value of $\rho_2(t)$ used in the derivation of the branch trajectory of Section 8.2.3 in the $\alpha_0 > 2/3$ case.

The pole Y_1 of μ, that is, the non-vanishing zero of ν, controls the asymptotic behavior of the probability of the number of unit-clauses: $\bar{\mu}(C_1) \asymp e^{-Y_1 C_1}$ when $C_1 \to \infty$. As long as $a_2 < 1$, Y_1 is positive, and $\bar{\mu}$ is localized. The average number of unit-clauses,

$$\langle C_1 \rangle = \sum_{C_1 \geq 0} \bar{\mu}(C_1)\, C_1 = \frac{d\mu}{dy_1}(y_1 = 0) = a_2\, \frac{2 - a_2}{2(1 - a_2)} \tag{8.14}$$

is finite. As a result, unit-clauses do not accumulate too much, and the probability that a contradiction occurs when a new variable is assigned is $O(1/N)$ only.

To calculate this probability, we consider step number T of DPLL. There are $V = N - T = N(1-t)$ not-yet-assigned variables, and C_1 unit-clauses, a stochastic number drawn from distribution $\bar{\mu}$ with $a_2 = c_2(t)/(1-t)$. In the absence of a unit-clause, the next variable is assigned through splitting of a clause and no immediate contradiction is possible. Otherwise, DPLL picks up a unit-clause and satisfies it. The probability that another given unit-clause is contradictory is $p = 1/(2V)$. Since clauses are independent, the probability that no contradiction emerges during step T is,

$$\mathrm{Prob}\,(\mathrm{T\ to\ T+1}) = \left(1 - \frac{1}{2(N-T)}\right)^{C_1 - 1} \ , \tag{8.15}$$

The probability that a contradiction never occurs till step $T = t\,N$ is therefore [24]:

$$\mathrm{Prob}\,(0\ \mathrm{to\ T = t\,N}) = \exp\left(- \int_0^t dt\, \frac{\langle \max(C_1 - 1, 0) \rangle (t)}{2\,(1-t)} \right) \ . \tag{8.16}$$

This expression, combined with (8.14) gives the probability that a contradiction arises along the branch trajectory calculated in Section 8.2.3. Two cases can be distinguished:

- If the ratio α_0 of clauses per variable is smaller than $\alpha_L \simeq 3.003$, $a_2(t)$ remains strictly smaller than unity, and the probability of success (8.16) is positive. Frieze and Suen have shown that contradictions have no dramatic consequences. The number of total backtrackings necessary to find a solution is bounded from above by a power of $\log N$. The final trajectory in the p, α plane is identical to the one shown in Section 8.2.3, and the increase in complexity is negligible with respect to $O(N)$.

- When $\alpha_0 > \alpha_L$, the trajectory intersects the $\alpha = 1/(1-p)$ line at some time t. At this point, $a_2(t) = \alpha(1-p) = 1$: the average number of unit-clauses, $\langle C_1 \rangle$, diverges. Unit-clauses accumulate, and contradictions unavoidably arise. Backtracking enters massively into play, signaling the crossover to the exponential regime.

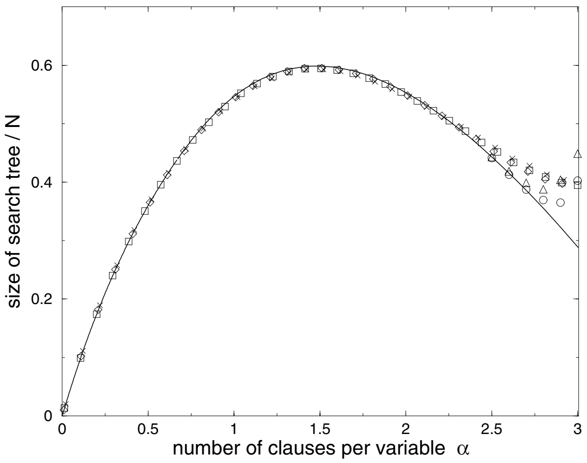


Figure 8.5: Complexity of solution in the SAT region for $\alpha < \alpha_L \simeq 3.003$, divided by the size N of the instances. Numerical data are for sizes $N = 50$ (crosses), 75 (squares), 100 (diamonds), 500 (triangles) and 1000 (circles). For the two biggest sizes, simulations have been carried out for ratios larger than 2.5 only. Data for different N collapse onto the same curve, proving that complexity scales linearly with N. The bold continuous curve is the analytical prediction $\gamma(\alpha)$ from Section 8.2.4. Note the perfect agreement with numerics except at large ratios where finite size effects are important, due to the crossover to the exponential regime above $\alpha_L \simeq 3.003$.

From the above discussion, it appears that a solution is found by DPLL essentially at the end of a single descent (Figure 8.1(A)) when $\alpha_0 < \alpha_L$ (lower SAT phase). Complexity thus scales linearly with N with a proportionality coefficient $\gamma(\alpha_0)$ smaller than unity.

For $\alpha_0 < 2/3$, clauses of length unity are never created by DPLL. Thus, DPLL assigns the overwhelming majority of variables through splittings. $\gamma(\alpha_0)$ simply equals the total fraction t_{end} of variables chosen by DPLL. From (8.7), we obtain

$$\gamma(\alpha_0) = 1 - \frac{4}{(\alpha_0+2)^2} \qquad (\alpha_0 \le 2/3) \,. \tag{8.17}$$

For larger ratios, i.e., $\alpha_0 > 2/3$, the trajectory must be decomposed into two successive portions. During the first portion, for times $0 < t < t^*$, 2-clauses are present with a non-vanishing density $c_2(t)$. Some of these 2-clauses are reduced to 1-clauses that have to be eliminated next. Consequently, when DPLL assigns an infinitesimal fraction dt of variables, a fraction $\rho_1(t) = \alpha(t)(1-p(t))dt$ are fixed by unit-propagation only. The number of nodes (divided by N) along the first part of the branch thus reads,

$$\gamma_1 = t^* - \int_0^{t^*} dt\, \alpha(t)(1-p(t)) \ . \tag{8.18}$$

At time t^*, the trajectory touches the 3-SAT axis $p = 1$ at ratio $\alpha^* \equiv \alpha(t^*) < 2/3$. The initial instance is then reduced to a smaller and smaller 3-SAT formula, with a ratio $\alpha(t)$ vanishing at t_{end}. According to the above discussion, the length of this second part of the trajectory equals

$$\gamma_2 = t_{end} - t^* \ . \tag{8.19}$$

It proves convenient to plot the total complexity $\gamma = \gamma_1 + \gamma_2$ in a parametric way. To do so, we express the initial ratio α_0 and the complexity γ in terms of the end time t^* of the first part of the branch. A simple calculation from (8.9) leads to

$$\begin{aligned} \alpha(t^*) &= -\frac{4\ln(1-t^*)}{3t^*(2-t^*)} \\ \gamma(t^*) &= 1 - \frac{4(1-t^*)}{(2+(1-t^*)^2\alpha_0(t^*))^2} + t^* + (1-t^*)\ln(1-t^*) \\ &\quad -\frac{1}{4}\,\alpha(t^*)\,(t^*)^2\,(3-t^*) \ . \end{aligned} \tag{8.20}$$

As t^* grows from zero to $t^*_L \simeq 0.892$, the initial ratio α_0 spans the range $[2/3; \alpha_L]$. The complexity coefficient $\gamma(\alpha_0)$ can be computed from (8.17) and (8.20) with the results shown in Figure 8.5. The agreement with numerical data is excellent.

8.3 Analysis of the Search Tree Growth in the UNSAT Phase

In this section, we present an analysis of search trees corresponding to UNSAT instances, that is, in the presence of massive backtracking. We first report results from numerical experiments, then explain our analytical approach for computing the complexity of resolution (size of search tree).

8.3.1 Numerical Experiments

For ratios above threshold ($\alpha_0 > \alpha_C \simeq 4.3$), instances almost never have a solution, but a considerable amount of backtracking is necessary before proving that clauses are incompatible. Figure 8.1(B) shows a generic UNSAT, or refutation, tree. In contrast to the previous section, the sequence of points (p, α) attached to the nodes of the search tree are not arranged along a line any longer, but rather form a cloud with a finite extension in the phase diagram of Figure 8.3. Examples of clouds are provided in Figure 8.6.

The number of points in a cloud, i.e., the size Q of its associated search tree, grows exponentially with N [11]. It is thus convenient to define its logarithm ω through $Q = 2^{N\omega}$. We directly counted Q experimentally, and averaged the corresponding logarithm ω over a

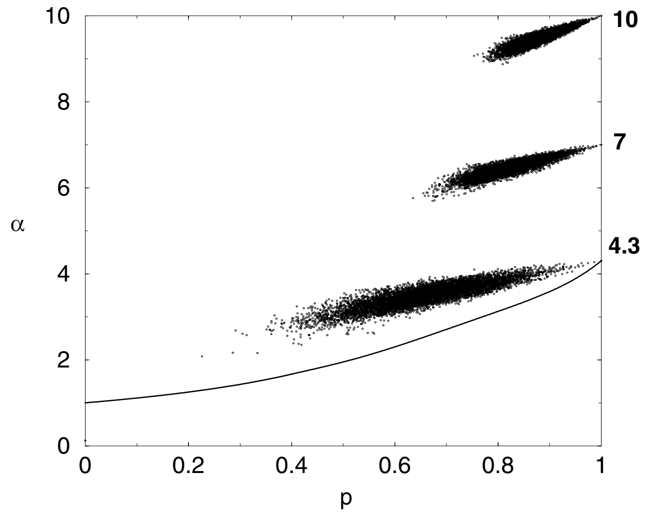


Figure 8.6: Clouds associated to search trees obtained from the resolution of three UNSAT instances with initial ratios $\alpha_0 = 4.3, 7$ and 10 respectively. Each point in the cloud corresponds to a splitting node in the search tree. Sizes of instances and search trees are $N = 120, Q = 7597$ for $\alpha_0 = 4.3$, $N = 200, Q = 6335$ for $\alpha_0 = 7$, and $N = 300, Q = 6610$ for $\alpha_0 = 10$.

Table 8.1: Logarithm of the complexity ω from experiments (EXP), theory (THE) from Section 8.3.4 and former linearization approximation (LIN) [13], as a function of the ratio α_0 of clauses per variable of the 3-SAT instance. Ratios above 4.3 correspond to UNSAT instances; the rightmost ratio lies in the upper SAT phase.

α_0	4.3	7	10	15	20	3.5
ω_{EXP}	0.089 ± 0.001	0.0477 ± 0.0005	0.0320 ± 0.0005	0.0207 ± 0.0002	0.0153 ± 0.0002	0.034 ± 0.003
ω_{THE}	0.0916	0.0486	0.0323	0.0207	0.0153	0.035
ω_{LIN}	0.0875	0.0477	0.0319	0.0206	0.0152	0.035

large number of instances. Results have then been extrapolated to the $N \to \infty$ limit [13] and are reported in Table 8.1. ω is a decreasing function of α_0 [7]: the larger α_0, the larger the number of clauses affected by a split, and the earlier a contradiction is detected. We will use the word "branch" to denote a path in the refutation tree which joins the top node (root) to a contradiction (leaf). The number of branches, B, is related to the number of nodes, Q, through the relation $Q = B - 1$, valid for any complete binary tree. As far as exponential (in N) scalings are concerned, the logarithm of B (divided by N) equals ω. In the following paragraph, we show how B can be estimated through the use of a matrix formalism.

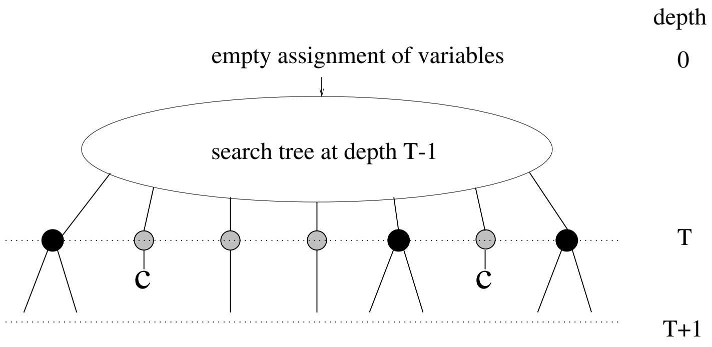


Figure 8.7: Imaginary, parallel growth process of an UNSAT search tree used in the theoretical analysis. Variables are fixed through unit propagation, or by the splitting heuristic as in the DPLL procedure, but branches evolve in parallel. T denotes the depth in the tree, that is the number of variables assigned by DPLL along each branch. At depth T, one literal is chosen on each branch among 1-clauses (unit propagation, grey circles not represented on Figure 8.1), or 2,3-clauses (splitting, black circles as in Figure 8.1). If a contradiction occurs as a result of unit propagation, the branch gets marked with C and dies out. The growth of the tree proceeds until all branches carry C leaves. The resulting tree is identical to the one built through the usual, sequential operation of DPLL.

8.3.2 Parallel Growth Process and Markovian Evolution Matrix

The probabilistic analysis of DPLL in the UNSAT regime appears to be a formidable task since the search tree of Figure 8.1(B) is the output of a complex, sequential process: nodes and edges are added by DPLL through successive descents and backtrackings (depth-first search). We have imagined a different, breadth-first building up of the refutation tree, which results in the same complete tree but can be mathematically analyzed. In our imaginary process, the tree grows in parallel, layer after layer (Figure 8.7). At time $T = 0$, the tree reduces to a root node, to which is attached the empty assignment of variables (nothing is known at the beginning of the search process), and an attached outgoing edge. At time T, that is, after having assigned T variables in the instance attached to each branch, the tree is made of $B(T)$ ($\leq 2^T$) branches, each one carrying a partial assignment of variables. At next time step $T \to T+1$, a new layer is added by assigning, according to DPLL heuristic, one more variable along every branch. As a result, a branch may keep growing through unitary propagation, get hit by a contradiction and die out, or split if the partial assignment does not induce unit clauses.

This parallel growth process is Markovian, and can be encoded in an instance-dependent evolution operator $\mathbf{H}$. A detailed definition and construction of $\mathbf{H}$ is presented in [16]. We limit ourselves to explaining hereafter the main steps:

- A 3^N dimensional-vector space is introduced. Each vector $|S\rangle$ in the spanning basis is in one-to-one correspondence with a partial assignment $S = (s_1, s_2, \ldots, s_N)$ of the N variables ($s_i = t$, f, u if variable x_i is, respectively, True, False, or Undetermined, i.e., not-yet-assigned).

- Let S be a partial assignment which does not violate the (unsatisfiable) instance **I** under consideration, and $S^{(j,x)}$, with $j = 1, \ldots, N$ and $x = t, f$, the partial assignment obtained from S by replacing s_j with x. Call $h_n(j|S)$ and $h_v(x|S,j)$ the probabilities that the heuristic (UC, GUC, ...) respectively chooses to assign variable x_j, and to fix it to $x \,(= t, f)$.
- The evolution operator $\mathbf{H}$ encodes the action of DPLL on **I**. Its matrix elements in the spanning basis are, see Figure 8.8,

  1. If S violates **I**, $\langle S'|\mathbf{H}|S\rangle = 1$ if $S' = S$, 0 otherwise.
  2. If S does not violate **I**, $\langle S'|\mathbf{H}|S\rangle = h_n(j|S) \times h_v(x|S,j)$ if $C_1(S) \geq 1$ and $S' = S^{(j,x)}$, $h_n(j|S)$ if $C_1(S) = 0$ and ($S' = S^{(j,x)}$ or $S' = S^{(j,\bar{x})}$), 0 otherwise. Here S, S' are the partial assignments corresponding to $|S\rangle, |S'\rangle$, and $C_1(S)$ the number of undetermined clauses of type 1 (unitary clauses) for partial assignment S.

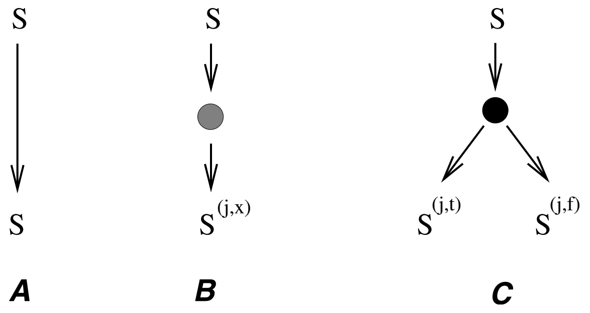


Figure 8.8: Transitions allowed by the heuristic-induced evolution operator. Grey and black nodes correspond to variables assigned through unit-propagation and split respectively, as in Figure 8.7. A. If partial assignment S already violates the instance **I**, it is left unchanged. B. If the partial assignment does not violate **I** and there is at least one unitary clause, a variable is fixed through unit propagation (grey node) e.g. $x_j = x$. The output partial assignment is $S^{j,x}$. C. If the partial assignment does not violate **I** and there is no unitary clause, a variable x_j is fixed through splitting (black node). Two partial assignments are generated, $S^{j,t}$ and $S^{j,f}$.

Then, the expectation value over the random assignments of variables of the size (number of leaves) of the search tree produced by DPLL to refute **I**, is equal to

$$B = \sum_S \langle S|\mathbf{H}^N|u, u, \ldots, u\rangle \ , \tag{8.21}$$

where $\mathbf{H}^N$ denotes the N^{th} (matrical) power of $\mathbf{H}$, the sum runs over all 3^N partial assignments S, and the rightmost vector corresponds to the initial, fully undetermined assignment of variables [16].

Calculation of the expectation value of the N^{th} power of $\mathbf{H}$, and of its average over the instance distribution is a hard task. We therefore turned to a simplifying approximation, called dynamical annealing. Call clause vector $\vec{C}(S)$ of a partial assignment S the three-dimensional vector $\vec{C} = (C_1, C_2, C_3)$ where C_j is the number of undetermined clauses of length j. The quantity we focus on is $\bar{B}(\vec{C}; T+1)$, the expectation number of branches at depth T in the search tree (Figure 8.7) carrying partial assignments with clause vector $\vec{C} = (C_1, C_2, C_3)$. Within the dynamical annealing approximation, the evolution of the $\bar{B}$s is Markovian,

$$\bar{B}(\vec{C}; T+1) = \sum_{\vec{C}'} \bar{\mathbf{H}}\,[\vec{C}, \vec{C}'; T]\; \bar{B}(\vec{C}'; T)\,. \tag{8.22}$$

The entries of the evolution matrix $\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T]$ can be calculated from the definition of the evolution matrix $\mathbf{H}$ [16]. They can be interpreted as the average number of branches with clause vector $\vec{C}$ that DPLL will generate through the assignment of one variable from a partial assignment of variables with clause vector $\vec{C}'$.

For the GUC heuristic, we find [13],

$$\begin{aligned}
\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T] = & \binom{C_3'}{C_3' - C_3} \left(\frac{3}{N-T}\right)^{C_3'-C_3} \left(1 - \frac{3}{N-T}\right)^{C_3} \times \\
& \sum_{w_2=0}^{C_3'-C_3} \left(\frac{1}{2}\right)^{C_3'-C_3} \binom{C_3' - C_3}{w_2} \times \\
& \Bigg\{ (1 - \delta_{C_1'}) \left(1 - \frac{1}{2(N-T)}\right)^{C_1'-1} \sum_{z_2=0}^{C_2'} \binom{C_2'}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \times \\
& \left(1 - \frac{2}{N-T}\right)^{C_2'-z_2} \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2 - C_2' - w_2 + z_2}\, \delta_{C_1 - C_1' - w_1 + 1} + \\
& \delta_{C_1'} \sum_{z_2=0}^{C_2'-1} \binom{C_2' - 1}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \left(1 - \frac{2}{N-T}\right)^{C_2'-1-z_2} \times \\
& \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2 - C_2' - w_2 + z_2 + 1} \left[\delta_{C_1 - w_1} + \delta_{C_1 - 1 - w_1}\right] \Bigg\}\,,
\end{aligned} \tag{8.23}$$

where δ_X denotes the Kronecker delta function over integers X: $\delta_X = 1$ if $X = 0$, $\delta_X = 0$ otherwise. Expression (8.23) is easy to obtain from the interpretation following Eq. (8.22), and the picture of containers in Figure 8.4 [13]. Among the $C_3' - C_3$ clauses that flow out from the leftmost 3-clauses container, w_2 clauses are reduced and go into the 2-clauses container, while the remaining $C_3' - C_3 - w_2$ are eliminated. w_2 is a random variable in the range $0 \le w_2 \le C_3' - C_3$ and drawn from a binomial distribution of parameter $1/2$, which represents the probability that the chosen literal is the negation of the one in the clause. It is assumed that the algorithm never chooses the variable among 3-clauses. This hypothesis is justified *a posteriori* because in the UNSAT region, there is always (except at the initial time $t = 0$) an extensive number of 2-clauses. Variables are chosen among 1-clauses or, in the absence of the latter, among 2-clauses. The term on the r.h.s. of Eqn. (8.23) beginning with $\delta_{C_1'}$ (respectively $1 - \delta_{C_1'}$) corresponds to the latter (resp. former) case. z_2 is the number of clauses (other than the one from which the variable is chosen) flowing out from the second container; it obeys a binomial distribution with parameter $2/(N-T)$, equal to the probability that the chosen variable appears in a 2-clause. The 2-clause container is, at the same time, poured with w_2 clauses. In an analogous way, the unitary clause container welcomes w_1 new clauses if it was empty at the previous step. If not, a 1-clause is eliminated by fixing the corresponding literal. The branch keeps growing as long as the level C_1 of the unit clauses container remains low, i.e., C_1 remains of the order of unity and the probability to have two, or more, 1-clauses with opposite literals can be neglected. This probability enters as a multiplicative factor in the third line of (8.23). Finally, we sum over all possible flow values w_2, z_2, w_1 that satisfy the conservation laws $C_2 - C_2' = w_2 - z_2$, $C_1 - C_1' = w_1 - 1$ when $C_1' \neq 0$ or, when $C_1' = 0$, $C_2 - C_2' = w_2 - z_2 - 1$, $C_1 = w_1$ if the literal is the same as the one in the clause or $C_1 = w_1 + 1$ if the literal is the negation of the one in the clause. The presence of two δ is responsible for the growth in the number of branches. In the real sequential DPLL dynamics, the inversion of a literal at a node requires backtracking; here, the two edges grow in parallel at each node according to Section 8.3.2.

8.3.3 Generating Function and Large-size Scaling

Let us introduce the generating function $G(\vec{y};T)$ of the average number of branches $\bar{B}(\vec{C};T)$ where $\vec{y} \equiv (y_1, y_2, y_3)$, through

$$G(\vec{y};T) = \sum_{\vec{C}} e^{\vec{y}\cdot\vec{C}}\, \bar{B}(\vec{C},T)\,, \qquad \vec{y}\cdot\vec{C} \equiv \sum_{j=1}^{3} y_j\, C_j\,, \tag{8.24}$$

where the first sum runs over all triplets of positive clause numbers. The evolution equation (8.22) for the $\bar{B}$s can be rewritten in term of the generating function G,

$$\begin{aligned} G(\vec{y};T+1) &= e^{-\gamma_1(\vec{y})}\, G\big(\vec{\gamma}(\vec{y});T\big) \\ &\quad + \left(e^{-\gamma_2(\vec{y})}(e^{y_1}+1) - e^{-\gamma_1(\vec{y})}\right)\, G\big(-\infty,\, \gamma_2(\vec{y}),\, \gamma_3(\vec{y});T\big) \end{aligned} \tag{8.25}$$

where $\vec{\gamma}$ is a vectorial function of argument $\vec{y}$ whose components read

$$\begin{aligned} \gamma_1(\vec{y}) &= y_1 + \ln\left[1 - \frac{1}{2(N-T)}\right]\,, \\ \gamma_2(\vec{y}) &= y_2 + \ln\left[1 + \frac{2}{N-T}\left(\frac{e^{-y_2}}{2}\,(1+e^{y_1}) - 1\right)\right]\,, \\ \gamma_3(\vec{y}) &= y_3 + \ln\left[1 + \frac{3}{N-T}\left(\frac{e^{-y_3}}{2}\,(1+e^{y_2}) - 1\right)\right]\,. \end{aligned} \tag{8.26}$$

To solve (8.25), we infer the large-N behavior of G from the following remarks:

1. Each time DPLL assigns variables through splitting or unit propagation, the numbers C_j of clauses of length j undergo $O(1)$ changes. It is thus sensible to assume that, when the number of assigned variables increases from $T_1 = t\,N$ to $T_2 = t\,N + \Delta T$ with ΔT very large but $o(N)$, e.g., $\Delta T = \sqrt{N}$, the densities $c_2 = C_2/N$ and $c_3 = C_3/N$ of 2- and 3-clauses have been modified by $o(1)$.

2. On the same time interval $T_1 < T < T_2$, we expect the number of unit-clauses C_1 to vary at each time step. But its distribution $\rho(C_1|c_2, c_3; t)$, conditioned to the densities c_2, c_3 and the reduced time t, should reach some well defined limit distribution. This claim is a generalization of the result obtained by [24] for the analysis of the GUC heuristic in the absence of backtracking.

3. As long as a partial assignment does not violate the instance, very few unit-clauses are generated, and splitting frequently occurs. In other words, the probability that $C_1 = 0$ is strictly positive as N becomes large.

The above arguments entice us to make the following claim. For large N, T at fixed ratio $t = T/N$, the generating function (8.24) of the average numbers $\bar{B}$ of branches is expected[4] to behave as

$$G(y_1, y_2, y_3; t\,N) = \exp\left[N\,\varphi(y_2, y_3; t) + \psi(y_1, y_2, y_3; t) + o(1)\right] . \tag{8.27}$$

Hypothesis (8.27) expresses in a concise way some important information on the distribution of clause populations during the search process that we now extract. Call ω the Legendre transform of φ,

$$\omega(\,c_2, c_3\,; t\,) = \min_{y_2, y_3}\left[\varphi(\,y_2, y_3\,; t\,) - y_2\,c_2 - y_3\,c_3\right] . \tag{8.28}$$

Then, combining equations (8.24), (8.27) and (8.28), we obtain

$$\lim_{N\to\infty}\frac{1}{N}\ln\;\bar{B}(C_1, c_2\,N, c_3\,N; t\,N) = \omega(c_2, c_3; t\,) , \tag{8.29}$$

independently of the (finite) number C_1 of unit clauses. In other words, the expectation value of the number of branches carrying partial assignments with $(1-t)\,N$ undetermined variables and $c_j\,N$ j-clauses ($j = 2, 3$) scales exponentially with N, with a growth function $\omega(c_2, c_3; t)$ related to $\varphi(y_2, y_3; t)$ through identity (8.28). Moreover, $\varphi(0, 0; t)$ is the logarithm of the number of branches (divided by N) after a fraction t of variables have been assigned. The most probable values of the densities $c_j(t)$ of j-clauses are then obtained from the partial derivatives of φ: $c_j(t) = \partial\varphi/\partial y_j(0, 0)$ for $j = 2, 3$. Let us emphasize that φ in (8.27) does not depend on y_1. This hypothesis simply expresses that, as far as non violating partial assignments are concerned, both terms on the right-hand side of (8.25) are of the same order, and that the density of unit-clauses, $c_1 = \partial\varphi/\partial y_1$, identically vanishes.

Similarly, function $\psi(y_1, y_2, y_3; t)$ is related to the generating function of the equilibrium distribution $\bar{\mu}(C_1|c_2, c_3, t)$ of unit-clause at fixed c_2, c_3, t, extending the definition and calculation of Section 8.2.4 valid in the absence of backtracking to the UNSAT regime,

$$e^{\psi(y_1,y_2,y_3;t)-\psi(0,y_2,y_3;t)} = \sum_{C_1\geq 0}\bar{\mu}(C_1|c_2, c_3, t)\;e^{\,y_1\,C_1} , \tag{8.30}$$

where $c_j = \partial\varphi/\partial y_j(y_2, y_3; t)$ ($j = 2, 3$) on the right-hand side of the above formula.

[4] See [29] for a similar large deviation ansatz in the context of the relaxation dynamics of the mean-field Ising model.

Inserting expression (8.27) into the evolution equation, (8.25), we find

$$\begin{aligned}\frac{\partial\varphi}{\partial t}(y_2,y_3;t) &= -y_1+\frac{2}{1-t}\left[e^{-y_2}\left(\frac{1+e^{y_1}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_2}(y_2,y_3;t)\\ &+ \frac{3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_3}(y_2,y_3;t)\\ &+ \ln\left[1+\mathcal{K}(y_1,y_2)\,e^{\psi(-\infty,y_2,y_3;t)-\psi(y_1,y_2,y_3;t)}\right] \end{aligned}\qquad(8.31)$$

where $\mathcal{K}(y_1,y_2)=e^{-y_2}(e^{2\,y_1}+e^{y_1})-1$. As φ does not depend upon y_1, the latter may be chosen at our convenience, e.g., to cancel $\mathcal{K}$ and the contribution from the last term in (8.31),

$$y_1=Y_1(y_2)\equiv y_2-\ln\left(\frac{1+\sqrt{1+4\,e^{y_2}}}{2}\right)\ .\qquad(8.32)$$

Such a procedure, similar to the kernel method [33], is correct in the major part of the y_2,y_3 space and, in particular, in the vicinity of $(0,0)$ which we focus on in this paper[5]. We end up with the following partial differential equation (PDE) for φ,

$$\frac{\partial\varphi}{\partial t}(y_2,y_3;t)=H\left[\frac{\partial\varphi}{\partial y_2},\frac{\partial\varphi}{\partial y_3},y_2,y_3,t\right]\ ,\qquad(8.33)$$

where H incorporates the details of the splitting heuristic[6],

$$\begin{aligned}H_{GUC}[c_2,c_3,y_2,y_3,t] &= -Y_1(y_2)+\frac{3\,c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]\\ &+ \frac{c_2}{1-t}\left(e^{-Y_1(y_2)}-2\right)\ .\end{aligned}\qquad(8.35)$$

We must therefore solve the partial differential equation (PDE) (8.33) with the initial condition,

$$\varphi(y_2,y_3,t=0)=\alpha_0\;y_3\ ,\qquad(8.36)$$

obtained through inverse Legendre transform (8.28) of the initial condition over $\bar{B}$, or equivalently over ω,

$$\omega(c_2,c_3;t=0)=\begin{cases}0 & \text{if}\quad c_3=\alpha_0\ ,\\ -\infty & \text{if}\quad c_3\neq\alpha_0\ .\end{cases}$$

5 It has, however, to be to modified in a small region of the y_2,y_3 space; a complete analysis of this case was carried out in [13].

6 For the UC heuristic,

$$H_{UC}=\ln 2+\frac{3\,c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]+\frac{c_2}{1-t}\left(\frac{3}{2}e^{-y_2}-2\right)\ .\qquad(8.34)$$

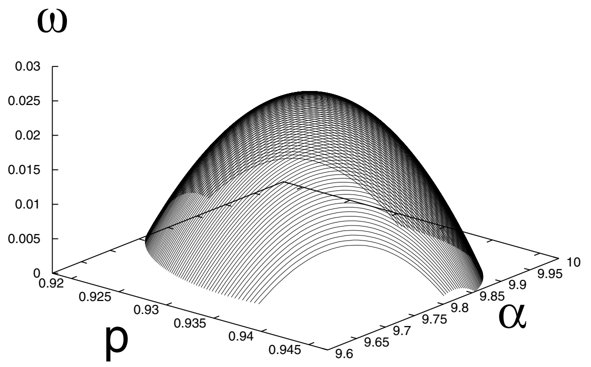


Figure 8.9: Snapshot of the surface $\omega(p, \alpha; t)$ for $\alpha_0 = 10$ at time (depth in the tree) $t = 0.05$. The height $\omega^*(t)$ of the top of the surface, with coordinates $p^*(t), \alpha^*(t)$, is the logarithm (divided by N) of the number of branches. The coordinates $(p^*(t), \alpha^*(t))$ define the tree trajectory shown in Figure 8.3. The halt line is hit at $t_h \simeq 0.094$.

8.3.4 Interpretation in Terms of Growth Process

We can interpret the dynamical annealing approximation made in the previous paragraphs, and the resulting PDE (8.33) as a description of the growth process of the search tree resulting from DPLL operation. Using Legendre transform (8.28), PDE (8.33) can be written as an evolution equation for the logarithm $\omega(c_2, c_3, t)$ of the average number of branches with parameters c_2, c_3 as the depth $t = T/N$ increases,

$$\frac{\partial \omega}{\partial t}(c_2, c_3, t) = H\left[c_2, c_3, -\frac{\partial \omega}{\partial c_2}, -\frac{\partial \omega}{\partial c_3}, t\right] . \tag{8.37}$$

Partial differential equation (PDE) (8.37) is analogous to growth processes encountered in statistical physics [36]. The surface ω, growing with "time" t above the plane c_2, c_3, or equivalently from (8.5), above the plane p, α (Figure 8.9), describes the whole distribution of branches. The average number of branches at depth t in the tree equals

$$B(t) = \int_0^1 dp \int_0 d\alpha \; e^{N\,\omega(p,\alpha;t)} \asymp e^{N\,\omega^*(t)} , \tag{8.38}$$

where $\omega^*(t)$ is the maximum over p, α of $\omega(p, \alpha; t)$ reached in $p^*(t), \alpha^*(t)$. In other words, the exponentially dominant contribution to $B(t)$ comes from branches carrying 2+p-SAT instances with parameters $p^*(t), \alpha^*(t)$, that is clause densities $c_2^*(t) = \alpha^*(t)(1 - p^*(t))$, $c_3^*(t) = \alpha^*(t)p^*(t)$. Along the tree trajectory, $\omega^*(t)$ grows thus from 0, on the right vertical axis, up to some halting time t_h, at which dominant branches almost surely get hit by contradictions. $\omega_{THE} = \omega^*(t_h)$ is our theoretical prediction for the logarithm of the complexity (divided by N)[7].

The hyperbolic line in Figure 8.3 indicates the halt points, where contradictions prevent dominant branches from further growing [13]. To obtain this curve, we calculate the probability $\bar{\mu}^*(t) \equiv \bar{\mu}(C_1 = 0|c_2^*(t), c_3^*(t), t)$ that a split occurs when a variable is assigned by DPLL,

$$\bar{\mu}^*(t) = \exp\left(\frac{\partial \varphi}{\partial t}(0, 0; t)\right) - 1 \ , \tag{8.39}$$

from (8.30) with $y_1 = -\infty, y_2 = y_3 = 0$ and (8.31) with $y_1 = y_2 = y_3 = 0$, respectively. The probability of split vanishes, and unit-clauses accumulate until a contradiction is obtained, when the tree stops growing,

$$\bar{\mu}^*(t) = 0 \to \frac{\partial \varphi}{\partial t}(0, 0; t) = \frac{\partial \omega}{\partial t}\big(c_2^*(t), c_3^*(t); t\big) = 0 \ . \tag{8.40}$$

From PDEs (8.33) or (8.37), this halting condition corresponds to crossing of

$$\alpha = \left(\frac{3+\sqrt{5}}{2}\right) \ln\left[\frac{1+\sqrt{5}}{2}\right] \frac{1}{1-p} \ . \tag{8.41}$$

Notice that the halt line in the UNSAT phase differs from the halt line $\alpha = 1/(1-p)$ calculated in Section 8.2.4 in the absence of backtracking.

Equation (8.37) is a first-order PDE[8] and can be solved using the characteristics method [35]. The idea is to describe the surface $\omega(c_2, c_3; t)$ as the union of curves, representing the evolution of a 'particle' in a 3-dimensional space with coordinates $c_2(t), c_3(t), \omega(t)$. Denoting by $p_j(t)$ the partial derivative $\partial\omega/\partial c_j(c_2(t), c_3(t); t)$ $(j = 2, 3)$, we write the conditions fulfilled by the particle to sit on the surface at any time as a set of five first-order ordinary coupled differential equations,

$$\begin{aligned}
\frac{dp_j}{dt}(t) &= -\frac{\partial H}{\partial c_j}\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] \ , \quad (j = 2, 3) \\
\frac{dc_j}{dt}(t) &= \frac{\partial H}{\partial p_j}\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] \ , \quad (j = 2, 3) \\
\frac{d\omega}{dt}(t) &= H\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] + \sum_{j=2,3} p_j(t)\, \frac{dc_j}{dt}(t) \ .
\end{aligned} \tag{8.42}$$

Assume now that we focus on dominant branches only, and want to calculate the coordinates c_2^*, c_3^*, ω^* of the top of the surface as a function of time, say t', positive and smaller than the halt time. To do this, we need to solve (8.42) for $0 < t < t'$ with boundary conditions,

$$c_2(0) = 0 \ , \quad c_3(0) = \alpha_0 \ , \quad \omega(0) = 0 \ , \tag{8.43}$$

[7] Notice that we have to divide the theoretical value by $\ln 2$ to match the definition used for numerical experiments; this is done in Table 8.1.

[8] This statement is correct in the large-size limit only. Finite-size corrections would introduce second-derivative terms with $1/N$ multiplicative coefficients. See [29] for a similar situation.

which expresses that all trajectories describe resolution of a 3-SAT instance with ratio α_0, and

$$p_2(t') = p_3(t') = 0 \, , \tag{8.44}$$

to match the end of the trajectory with the top of the surface ω at time t'. Numerical resolution of equations (8.42) with boundary conditions (8.43,8.44) is shown in Figure 8.10. Clause densities c_2, c_3 keep positive at any time as expected. The parametric plot of the final coordinates, $p^*(t'), \alpha^*(t')$, as a function of t' defines the tree trajectories on Figure 8.3. Values of ω obtained for various initial ratios α_0 are listed in Table 8.1, and compared to the linearization approximation developed in [13].

We have plotted the surface ω at different times, with the results shown in Figure 8.9 for $\alpha_0 = 10$. Values of ω_{THE}, obtained for $4.3 < \alpha < 20$ by solving (8.37) compare very well with the numerical results (Table 8.1). Although our calculation is not rigorous, it provides a very good quantitative estimate of the complexity. It is therefore expected that our dynamical annealing approximation is quantitatively accurate. It is a reasonable conjecture that it becomes exact at large ratios α_0, where PDE (8.33) can be exactly solved.

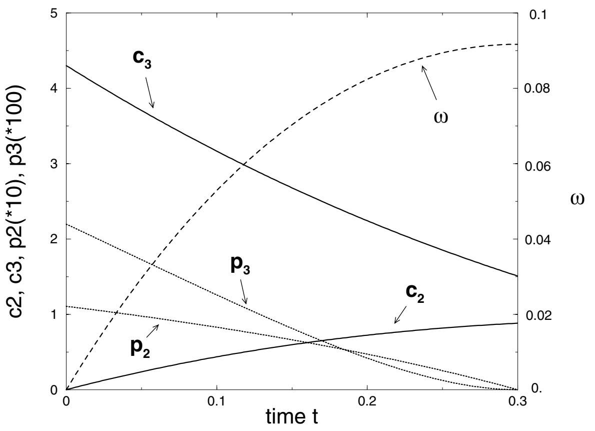


Figure 8.10: Clause densities $c_2(t), c_3(t)$, logarithm $\omega(t)$ of the number of branches, local derivatives $p_2(t), p_3(t)$ as a function of time t for the characteristic curve reaching the top surface at halt time $t_h \simeq 0.299$ corresponding to ratio $\alpha_0 = 4.3$. Left axis scale corresponds to c_2, c_3 and $p_2 \times 10$, $p_3 \times 100$; right axis scale shows values of ω. Boundary conditions (8.43,8.44) can be seen from the curves. The vanishing of the derivative of ω at $t = t_h$ stems from the halt condition (8.40).

8.3.4.1 Asymptotic Equivalent of ω for Large Ratios

Resolution of PDE (8.37) in the large ratio α_0 limit gives (for the GUC heuristic),

$$\omega_{THE}(\alpha_0) \asymp \frac{3+\sqrt{5}}{6 \ln 2} \left[\ln \left(\frac{1+\sqrt{5}}{2} \right) \right]^2 \frac{1}{\alpha_0} \, . \tag{8.45}$$

This result exhibits the $1/\alpha_0$ scaling proven by [7], and is conjectured to be exact. As α_0 increases, search trees become smaller and smaller, and correlations between branches, weaker and weaker, making dynamical annealing increasingly accurate.

8.4 Hard SAT Phase: Average Case and Fluctuations

8.4.1 Mixed Branch and Tree Trajectories

The main interest of the trajectory framework proposed in this paper is best seen in the upper SAT phase, that is, for ratios α_0 ranging from α_L to α_C. This intermediate region juxtaposes branch and tree behaviors [14], see search tree in Figures 8.1(C) and 8.11.

The branch trajectory, started from the point $(p = 1, \alpha_0)$ corresponding to the initial 3-SAT instance, hits the critical line $\alpha_c(p)$ at some point G with coordinates (p_G, α_G) after $N\ t_G$ variables have been assigned by DPLL, see Figure 8.12. The algorithm then enters the UNSAT phase and, with high probability, generates a 2+p-SAT instance with no solution. A dense subtree, that DPLL has to go through entirely, forms beyond G up until the halt line (left subtree in Figure 8.11). The size of this subtree can be analytically predicted from the theory exposed in Section 8.3. All calculations are identical, except initial condition (8.36) which has to be changed into

$$\varphi(y_2, y_3, t = 0) = \alpha_G\ (1 - p_G)\ y_2 + \alpha_G\ p_G\ y_3\ . \tag{8.46}$$

As a result we obtain the size $2^{N_G\ \omega_G}$ of the unsatisfiable subtree to be backtracked (leftmost subtree in Figure 8.11). $N_G = N\,(1 - t_G)$ denotes the number of undetermined variables at point G.

G is the highest backtracking node in the tree (Figures 8.1(C) and 8.11) reached by DPLL, since nodes above G are located in the SAT phase and carry 2+p-SAT instances with solutions. DPLL will eventually reach a solution. The corresponding branch (rightmost path in Figure 8.1(C)) is highly non-typical and does not contribute to the complexity, since almost all branches in the search tree are described by the tree trajectory issued from G (Figures 8.3,8.12). We expect that the computational effort DPLL requires to find a solution will, to exponential order in N, be given by the size of the left unsatisfiable subtree of Figure 8.11. In other words, massive backtracking will certainly be present in the right subtree (the one leading to the solution), and no significant statistical difference is expected between both subtrees.

We have experimentally checked this scenario for $\alpha_0 = 3.5$. The average coordinates of the highest backtracking node, $(p_G \simeq 0.78, \alpha_G \simeq 3.02)$, coincide with the computed intersection of the single branch trajectory (Section 8.2.2) and the estimated critical line $\alpha_c(p)$ [13]. As for complexity, experimental measures of ω from 3-SAT instances at $\alpha_0 = 3.5$, and of ω_G from 2+0.78-SAT instances at $\alpha_G = 3.02$, obey the expected identity

$$\omega_{THE} = \omega_G \times (1 - t_G)\ , \tag{8.47}$$

and are in very good agreement with theory (Table 8.1). Therefore, the structure of search trees corresponding to instances of 3-SAT in the upper SAT regime reflects the existence of a critical line for 2+p-SAT instances. The exponential scaling of the complexity ($\omega > 0$) in this upper SAT regime was recently rigorously established [3].

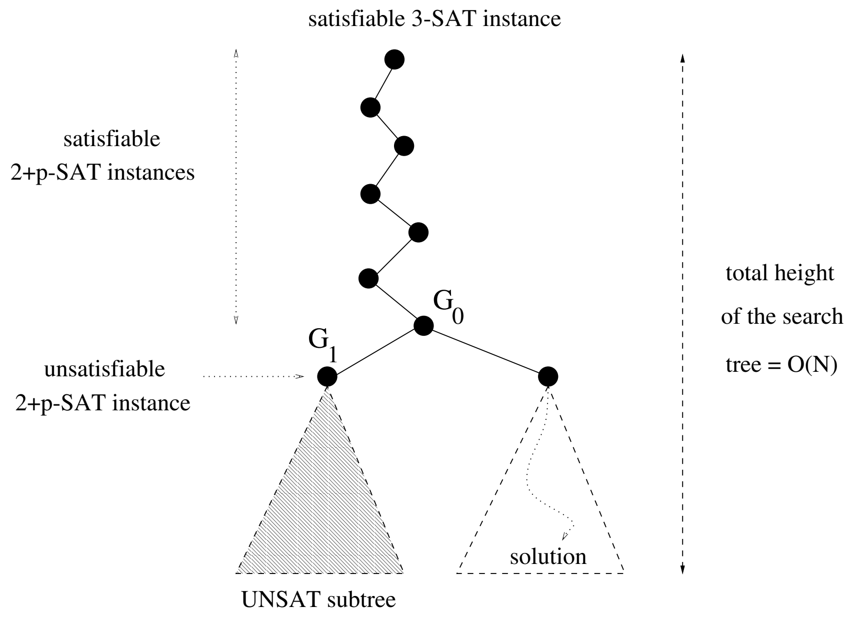

satisfiable 3-SAT instance
satisfiable 2+p-SAT instances
unsatisfiable 2+p-SAT instance
G0
G1
total height of the search tree = O(N)
solution
UNSAT subtree

Figure 8.11: Detailed structure of the search tree in the upper SAT phase ($\alpha_L < \alpha < \alpha_C$). DPLL starts with a satisfiable 3-SAT instance and transforms it into a sequence of 2+p-SAT instances. The leftmost branch in the tree symbolizes the first descent made by DPLL. Above node G_0, instances are satisfiable while below G_1, instances have no solutions. A grey triangle accounts for the (exponentially) large refutation subtree that DPLL has to go through before backtracking above G_1 and reaching G_0. By definition, the highest node reached back by DPLL is G_0. Further backtracking, below G_0, will be necessary but a solution will be eventually found (right subtree), see Figure 8.1(C).

8.4.2 Distribution of Running Times

While in the upper SAT phase, search trees almost always look like Figure 8.1(B), they may sometimes consist of a single branch (Figure 8.1(A)). Figure 8.13 shows the normalized histogram of the logarithm ω of the solving times of instances with ratio $\alpha_0 = 3.5$ and various sizes N [14, 31, 43]. The histogram is made of a narrow peak (left side) followed by a wider bump (right side). As N grows, the right peak acquires more and more weight, while the left peak progressively disappears. The abscissa of the center of the right peak reaches a finite value $\omega^* \simeq 0.035$ as $N \to \infty$. This right peaks thus corresponds to the core of exponentially hard resolutions: with high probability resolutions of instances requiring a time scaling as $2^{N\omega^*}$ as the size of the instance gets larger and larger, in agreement with Section 8.4.1.

On the contrary, the location of the maximum of the left peak vanishes as $\log_2(N)/N$ when the size N increases, indicating that the left peak accounts for polynomial (linear) resolutions. We have thus replotted the data shown in Figure 8.13, changing the scale of the horizontal axis $\omega = \log_2(Q)/N$ into Q/N. Results are shown in Figure 8.14. We have lim-

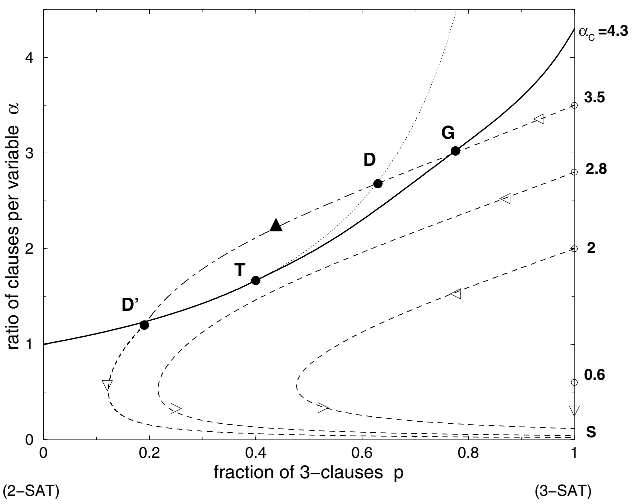


Figure 8.12: Phase diagram of 2+p-SAT and dynamical trajectories of DPLL for satisfiable instances. See caption of Figure 8.3 for definitions of critical lines and and trajectories. When the initial ratio lies in the $\alpha_L < \alpha_0 < \alpha_C$ range, with high probability, a contradiction arises before the trajectory crosses the dotted curve $\alpha = 1/(1-p)$ (point D). Through extensive backtracking, DPLL later reaches back to the highest backtracking node in the search tree (G) and finds a solution at the end of a new descending branch, see Figure 8.1(B). With exponentially small probability, the trajectory (dot-dashed curve, full arrow) is able to cross the "dangerous" region where contradictions are likely to occur; it then exits from this contradictory region (point D') and ends up with a solution (lowest dashed curve, open arrow).

ited ourselves to $Q/N < 1$, as the range of interest for analyzing the left peak of Figure 8.13. The maximum of the distribution is located at $Q/N \simeq 0.2 - 0.25$, with weak dependence upon N. The cumulative probability P_{lin} to have a complexity Q less than, or equal to N, i.e., the integral of Figure 8.14 over $0 < Q/N < 1$, decreases very quickly with N. We find an exponential decrease, $P_{lin} = 2^{-N\zeta}$, see inset of Figure 8.14. The rate $\zeta \simeq 0.011 \pm 0.001$ is determined from the slope of the logarithm of the probability shown in the inset.

The existence of rare but fast resolutions suggests the use of a systematic restart heuristic to speed up resolution [28]: if a solution is not found before N splits, DPLL is stopped and launched again after some random permutations of the variables and clauses. Intuitively, the expected number of restarts necessary to find a solution should indeed be equal to the inverse of the weight of the linear complexity peak in Figure 8.13, with a resulting total complexity scaling as $N\ 2^{0.011\,N}$, and much smaller than the one-run complexity $2^{0.035\,N}$ of DPLL. We check the above reasoning by measuring the number N_{rest} of restarts performed before a solution is finally reached with the restart heuristic, and averaging $\log_2(N_{res})$ over a large number of random instances. Results are reports in the inset of Figure 8.14. The typical number $N_{\text{rest}} = 2^{N\bar{\zeta}}$ of required restarts clearly grows exponentially as a function of the size N with a rate $\bar{\zeta} = 0.012 \pm 0.001$. Within the accuracy of the experiments, ζ and $\bar{\zeta}$ coincide as expected.

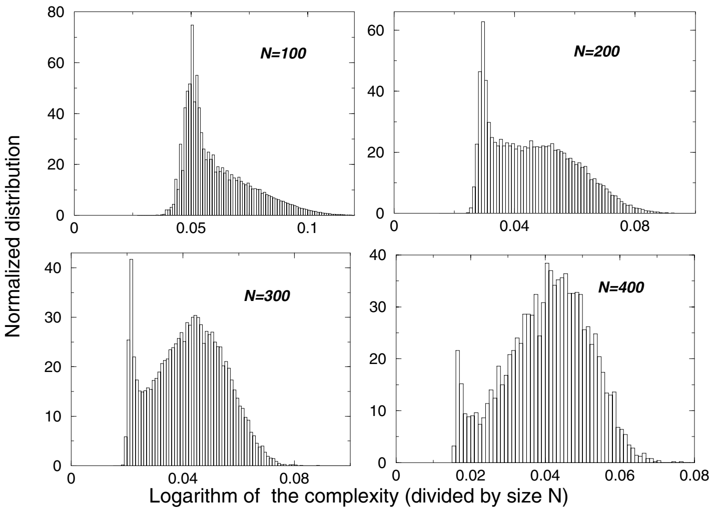


Figure 8.13: Probability distribution of the logarithm ω of the complexity (base 2, and divided by N) for $\alpha = 3.5$ and for different sizes N. Histograms are normalized to unity and obtained from 400 000 ($N = 100$), 50 000 ($N = 200$), 20 000 ($N = 300$), and 5 000 ($N = 400$) samples.

Experiments provide intuition about runs that are able to find a solution without backtracking. These are typically runs in which, although the 2-SAT subformula is supercritical ($\alpha > 1/(1-p)$) and many ($O(N)$) unitary clauses are present, no pair of these unitary clauses is contradictory (Figure 8.12). Such a "miracle" occurs with an exponentially small probability as calculated in Section 8.4.3.

This statement is supported by the analysis of the number of unit-clauses generated during easy resolutions. We have measured the maximal number $(C_1)_{\text{max}}$ of unit-clauses generated along the last branch in the tree, leading to the solution S (Figure 8.1(B)). We found that $(C_1)_{\text{max}}$ scales linearly with N with an extrapolated ratio $(C_1)_{\text{max}}/N \simeq 0.022$ for $\alpha = 3.5$. This linear scaling of the number of unit-clauses is an additional proof of the trajectory entering the "dangerous" region $\alpha > 1/(1-p)$ of the phase diagram where unit-clauses accumulate. In the presence of a $O(N)$ number of 1-clauses, the probability of survival of the branch (absence of contradictory literals among the unit-clauses) will be exponentially small in N, in agreement with the scaling of the left peak weight in Figure 8.13.

8.4.3 Large Deviation Analysis of the First Branch in the Tree

We give, in the following, a lower bound to the probability that DPLL finds a solution without ever backtracking. Due to the absence of backtracking, the same probabilistic setting as in Section 8.2 may be applied: the search heuristic defines a Markov chain for the evolution of subformulas as more and more variables are set. The major difference is that we are now considering initial densities above α_L, which means that the probability of the events we

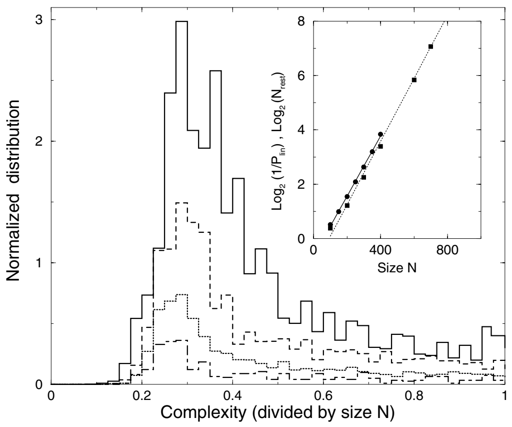


Figure 8.14: Probability distributions of the complexity Q (divided by the size N) for sizes $N = 100$ (full line), $N = 200$ (dashed line), $N = 300$ (dotted line), $N = 400$ (dashed-dotted line). Distributions are not shown for complexities larger than N. Inset: Minus logarithm of the cumulative probability of complexities smaller or equal to N as a function of N, for sizes ranging from 100 to 400 (full line); logarithm of the number of restarts necessary to find a solution for sizes ranging from 100 to 1000 (dotted line). Slopes are equal to $\zeta = 0.0011$ and $\bar{\zeta} = 0.00115$ respectively.

look for are exponentially small (in the number N of variables). In other words, rather than considering the mean resolution trajectory (which unavoidably leads to a contradiction and backtracking), we need to look at large deviations from this trajectory. Notice that, though we focus on a specific algorithm, namely GUC, our approach and the spirit of our results should hold for other heuristics.

The probability $\bar{B}(\vec{C};T)$ that the first branch of the tree carries an instance with C_j j-clauses ($j = 1, 2, 3$) after T variables have been assigned (and no contradiction has occurred) obeys the Markovian evolution equation (8.22). The entries of the transition matrix $\bar{H}$ are given (for the GUC heuristic) by (8.23) where $\delta_{C_1-w_1} + \delta_{C_1-w_1+1}$ replaced with $\delta_{C_1-w_1+1}$ in the last line.

The generating function G associated to probability $\bar{B}$ (8.24) obeys equation (8.25) with $(e^{y_1} + 1)$ replaced with 1, and $\gamma_1(\vec{y})$ in Eqn. (8.26) changed into

$$\gamma_1(\mathbf{y}) = y_1 + \ln\left[1 + \frac{1}{N-T}\left(\frac{e^{-y_1}}{2} - 1\right)\right] . \tag{8.48}$$

We now present the partial differential equations (PDE) obeyed by φ. Two cases must be distinguished: the number C_1 of unit-clauses may be bounded ($C_1 = O(1), c_1 = o(1)$), or of the order of the instance size ($C_1 = \Theta(N), c_1 = \Theta(1)$).

8.4.3.1 $C_1 = O(1)$: A Large Deviation Analysis Around Frieze and Suen's Result

When DPLL starts running on a 3-SAT instance, very few unit-clauses are generated and splittings occur frequently. In other words, the probability that $C_1 = 0$ is strictly positive when N becomes large. Consequently, both terms on the right-hand side of (8.25) are of the same order, and we make the hypothesis that φ does not depend on y_1: $\varphi(y_1, y_2, y_3; t) = \varphi(y_2, y_3; t)$. This hypothesis simply expresses that $c_1 = \partial\varphi/\partial y_1$ identically vanishes. Inserting expression (8.27) into the evolution equation (8.25), we find[9]

$$\frac{\partial \varphi}{\partial t} = -y_2 + 2\, g(y_2, y_2; t)\, \frac{\partial \varphi}{\partial y_2} + 3\, g(y_2, y_3; t)\, \frac{\partial \varphi}{\partial y_3} \ , \tag{8.49}$$

where function g is defined

$$g(u, v; t) = \frac{1}{1-t} \left(\frac{e^{-v}}{2} \left(1 + e^{u}\right) - 1 \right) \ . \tag{8.50}$$

PDE (8.49) together with initial condition $\varphi(\mathbf{y}; t = 0) = \alpha_0\, y_3$ (where α_0 is the ratio of clauses per variable of the 3-SAT instance) can be solved exactly with the resulting expression,

$$\begin{aligned} \varphi(y_2, y_3; t) \quad &= \quad \alpha_0 \ln \left[1 + (1-t)^3 \left(e^{y_3} - \frac{3}{4} e^{y_2} - \frac{1}{4} \right) + \frac{3(1-t)}{4} (e^{y_2} - 1) \right] \\ &+ \quad (1-t)\, y_2\, e^{y_2} + (1-t)(e^{y_2} - 1) \ln(1-t) \\ &- \quad (e^{y_2} + t - t\, e^{y_2}) \ln\left(e^{y_2} + t - t\, e^{y_2}\right) \ . \end{aligned} \tag{8.51}$$

Chao and Franco, and Frieze and Suen's analysis of the GUC heuristic may be recovered when $y_2 = y_3 = 0$ as expected. It is an easy check that $\varphi(y_2 = 0, y_3 = 0; t) = 0$, i.e., the probability of survival of the branch is not exponentially small in N [24], and that the derivatives $c_2(t), c_3(t)$ of $\varphi(y_2, y_3; t)$ with respect to y_2 and y_3 coincide with the solutions of (8.4).

In addition, (8.51) also provides a complete description of rare deviations of the resolution trajectory from its highly probable locus shown in Figure 8.3. As a simple numerical example, consider DPLL acting on a 3-SAT instance of ratio $\alpha_0 = 3.5$. Once, e.g., $t = 20\%$ of variables have been assigned, the densities of 2- and 3-clauses are with high probability equal to $c_2 \simeq 0.577$ and $c_3 \simeq 1.792$ respectively. Expression (8.51) gives access to the exponentially small probabilities that c_2 and c_3 differ from their most probable values. For instance, choosing $y_2 = -0.1, y_3 = 0.05$, we find from (8.51) and (8.28) that there is a probability $e^{-0.00567N}$ that $c_2 = 0.504$ and $c_3 = 1.873$ for the same fraction $t = 0.2$ of eliminated variables. By scanning all the values of y_2, y_3 we can obtain a complete description of large deviations from Frieze and Suen's result[10].

The assumption $C_1 = O(1)$ breaks down for the most probable trajectory at some fraction t_D, e.g., $t_D \simeq 0.308$ for $\alpha_0 = 3.5$ at which the trajectory hits point D on Figure 8.12. Beyond

[9] PDE (8.49) is correct in the major part of the y_1, y_2, y_3 space and, in particular, in the vicinity of $\mathbf{y} = \mathbf{0}$ which we focus on in this paper. It has, however, to be to modified in a small region of the y_1, y_2, y_3 space; a complete analysis of this case is not reported here but may be easily reconstructed along the lines of Appendix A in [13].

[10] Though we are not concerned here with sub-exponential (in N) corrections to probabilities, we mention that it is possible to calculate the probability of split, $\bar{\mu}(C_1 = 0)$, extending the calculation of Section 8.2.4 to $\mathbf{y} \neq \mathbf{0}$.

D, 1-clauses accumulate and the probability of survival of the first branch is exponentially small in N.

8.4.3.2 Case $C_1 = O(N)$: Passing Through the "Dangerous" Region

When the number of unit-clauses becomes of the order of N, variables are almost surely assigned through unit-propagation. The first term on the right-hand side of equation (8.25) is now exponentially dominant with respect to the second one. The density of 1-clauses is strictly positive, and φ depends on y_1. We then obtain the following PDE,

$$\frac{\partial \varphi}{\partial t} = -y_1 + g(-\infty, y_1; t)\,\frac{\partial \varphi}{\partial y_1} + 2\,g(y_1, y_2; t)\,\frac{\partial \varphi}{\partial y_2} + 3\,g(y_2, y_3; t)\,\frac{\partial \varphi}{\partial y_3}\,, \tag{8.52}$$

with $g(u, v; t)$ given by (8.50). When $y_1 = y_2 = y_3 = 0$, (8.52) simplifies to

$$\frac{dz}{dt}(t) = -\frac{c_1(t)}{2(1-t)}\,, \tag{8.53}$$

where $c_1(t)$ is the most probable value of the density of unit-clauses, and $z(t)$ is the logarithm of the probability that the branch has not encountered any contradiction (divided by N). The interpretation of (8.53) is transparent. Each time a literal is assigned through unit-propagation, there is a probability $(1 - 1/2/(N-T))^{C_1 - 1} \simeq e^{-c_1/2/(1-t)}$ that no contradiction occurs. The right-hand side of (8.53) thus corresponds to the rate of decay of z with "time" t.

PDE (8.52) can be solved numerically [14], with results as shown in Figure 8.15. The calculated values of $\zeta \simeq 0.01$, $(c_1)_{\max} \simeq 0.022$ and $\gamma \simeq 0.21$ are in very good agreement with numerical experiments (Section 8.4.2). This agreement extends over the whole range $\alpha_L \leq \alpha_0 \leq \alpha_C$ [14].

8.4.3.3 More on Restarts and Cut-off

This study suggests that the cut-off time, at which the search is halted and restarted, need not be precisely tuned but is simply given by the size of the instance. This conclusion could be generic and apply to other combinatorial decision problems and other heuristics. More precisely, if a combinatorial problem admits some efficient (polynomial) search heuristic for some values of control parameter (e.g., the ratio α here, or the average adjacency degree for the coloring problem of random graphs), there might be an exponentially small probability that the heuristic is still successful (in polynomial time) in the range of parameters where resolution almost surely requires massive backtracking and exponential effort. When the decay rate of the polynomial time resolution probability ζ is smaller than the growth rate ω of the typical exponential resolution time, restart procedures with a cut-off in the search equal to a polynomial of the instance size will lead to an exponential speed-up of resolutions.

In principle, one could not rule out the existence of even luckier runs than linear ones. For instance, there could exist exponentially long (complexity $2^{\omega' N}$ with $0 < \omega' < \omega$) and rare (probability $2^{-\zeta' N}$ with $0 < \zeta' < \zeta$) runs with $\omega' + \zeta' < \zeta$. If so, $2^{\omega' N}$ would be a better cut-off for restart than N. A recent analysis of the distribution of exponentially long resolutions indicates this is not so for the problem of the vertex covering of random graphs, and that the optimal cut-off for restarts is indeed the instance size itself [39].

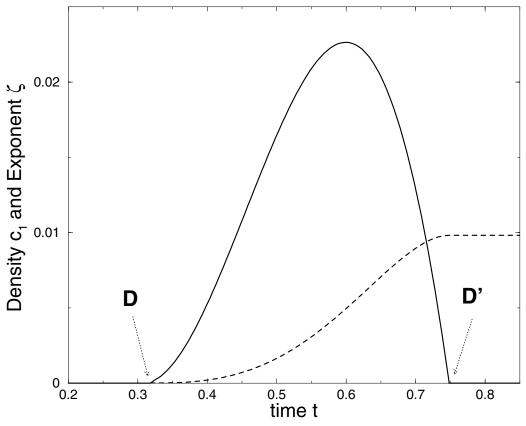


Figure 8.15: Density c_1 of unitary clauses (full line) and logarithm z (8.53) of the probability of the absence of contradiction (dashed line) along the first search branch as a function of time t (fraction of assigned variables) for an initial ratio $\alpha = 3.5$. The density of unit clauses is positive between points D and D$'$ along the branch trajectory of Figure 8.12; z is null before the trajectory reaches D, and constant and equal to the exponent ζ beyond D$'$.

8.5 The Random Graph Coloring Problem

In this section we apply the approach described above for random SAT to random COL. More precisely, we analyze the performances of a complete DPLL algorithm capable of determining whether a given graph is 3-colorable or not [20]. The algorithm is based on a combination of a coloring heuristic, 3-GREEDY-LIST (3-GL), and of backtracking steps. We first present the algorithm and then analyze the dynamics of its resolution time.

8.5.1 Description of DPLL Algorithm for Coloring

The action of the coloring procedure is described as follows:

- Necessary Information: while running, the algorithm maintains for each uncolored vertices, a list of available colors, which consists of all the colors that can be assigned to this vertex, given the colors already assigned to surrounding vertices.

- Coloring Order: the order in which the vertices are colored, is such that the most constrained vertices, i.e., with the least number of available colors, are colored first. At each time step, a vertex is chosen among the most constrained vertices, and its color is selected from the list of its available colors. Both choices are done according to some heuristic rule, which can be unbiased (no preference is made between colors), or biased (following a hierarchy between colors), see next section.

- List Updating: to ensure that no adjacent vertices have the same color, whenever a vertex is assigned a color, this color is removed from the lists (if present) which are attached to each of the uncolored neighbors.

- Contradictions and Backtracking: a contradiction occurs as soon as one of the lists becomes empty. Then, the algorithm backtracks to the most recently chosen vertex, which has more than one available color (the closest node in the search tree – see definition below).

- Termination Condition: the algorithm stops when all vertices are colored, or when all coloring possibilities have been tried.

A search tree can describe the action of the algorithm as for the SAT problem. A node in the tree represents a vertex chosen by the algorithm, which has more than one color in its available-colors list. An edge which comes out of a node, corresponds to a possible color of the chosen vertex. A leaf is either a solution (S) or a contradiction (denoted by C), see Figure 8.1.

8.5.2 Coloring in the Absence of Backtracking

Let us call the 3-GL heuristic the incomplete version of the above algorithm, obtained when the algorithm stops if a coloring is found (and outputs "Colorable"), or just after the first contradiction, instead of backtracking (and outputs "Don't know if colorable or not"). In contrast to the 3-GL algorithm with backtracking, the 3-GL heuristic is not able to prove the absence of a solution, and is amenable to rigorous analysis [5, 6].

In the simplest case, vertices and colors are chosen purely randomly without any bias between colors (Coloring Order step described above). This "symmetric" 3-GL heuristic verifies two key properties on which our analysis relies. The first one is a statistical invariance called the R-property. Throughout the execution of the algorithm, the uncolored part of the graph is distributed as $G((1-t)N, p)$ where t is the number of colored vertices divided by N. The second property is color symmetry. The search heuristic is symmetric with respect to the different colors, and the initial conditions are symmetric as well. Hence, the evolution of the algorithm can be exactly monitored by tracking of the three numbers $N_j(T)$ of j-color nodes ($j = 1, 2, 3$) only, without distinction between the colors available to each of these nodes.

The analysis of the evolution of these numbers in the course of the coloring was carried out by Achlioptas and Molloy [6]. In a way very similar to Figure 8.4 and due to the R-property, the average flows of vertices, $w_2(T)$ from $N_3(T)$ to $N_2(T)$, and $w_1(T)$ from $N_2(T)$ to $N_1(T)$ are $c\,N_3(T)/N$ and $2\,c\,N_2(T)/(3\,N)$, respectively. Note that the last factor is due to the fact that $2/3$ of the 2-color nodes adjacent to the vertex just colored have the used color as one of their two available colors. Hence, the evolution equations for the three populations of vertices read,

$$\begin{aligned} N_3(T+1) &= N_3(T) - w_2(T)\ , \\ N_2(T+1) &= N_2(T) + w_2(T) - w_1(T) - \delta N_1(T)\ , \\ N_1(T+1) &= N_1(T) + w_1(T) - (1 - \delta N_1(T))\ . \end{aligned} \tag{8.54}$$

where $\delta N_1(T) = 1$ if $N_1(T) = 0$ (a 2-color vertex is colored) and $\delta N_1(T) = 0$ if $N_1(T) \neq 0$ (a 1-color vertex is colored). For $c > 1$, both $N_2(T)$ and $N_3(T)$ are extensive in N, and can be written as

$$N_i(T) = n_i(T/N)\ N + o(N)\ . \tag{8.55}$$

The appearance of the reduced time, $t = T/N$, means that population densities $n_i(T/N)$ change by $O(1)$ over $O(N)$ time intervals. To avoid the appearance of contradictions, the number of 1-color vertices must remain of $O(1)$ throughout the execution of the algorithm. From queuing theory, this requires $w_1(t) < 1$, that is

$$\frac{2}{3}\, c\, n_2(t) < 1 \tag{8.56}$$

which means that 1-color nodes are created slowly enough to color them and do not accumulate. Thus, in the absence of backtracking, the evolution equations for the densities are

$$\frac{dn_3(t)}{dt} = -c\, n_3(t)\,, \qquad \frac{dn_2(t)}{dt} = c\, n_3(t) - 1\,. \tag{8.57}$$

The solution of these differential equations, with initial conditions $n_3(0) = 1$, $n_2(0) = 0$, is $n_3(t) = e^{-c\,t}\,, \quad n_2(t) = 1 - t - e^{-c\,t}$. Equations (8.57) were obtained under the assumption that $n_2(t) > 0$ and hold until $t = t_2$ at which the density n_2 of 2-color nodes vanishes. For $t > t_2$, 2-color vertices no longer accumulate. They are colored as soon as they are created. 1-color vertices are almost never created, and the vertices colored by the algorithm are either 2-, or 3-color vertices. Thus, when $t_2 < t < 1$, $n_2(t) = 0$, and $n_3(t) = 1 - t$ decreases to zero. A proper coloring is found at $t = 1$, i.e., when all nodes have been colored.

These equations define the trajectory of the algorithm in phase space in the absence of contradictions, i.e., as long as condition (8.56) is fulfilled. The trajectory corresponding to $c = 3$ is plotted on Figure 8.16. For $c < c_L \approx 3.847$, condition (8.56) is never violated, and the probability that the algorithm succeeds in finding an appropriate coloring without backtracking is positive. The complexity $\gamma(c)\, N$ of the algorithm in the absence of backtracking is linear with N, and equals the number of nodes in the single branch of the search tree.

$$\gamma(c) = 1 - \frac{2}{3}\, c \int_0^{t_2} dt\; n_2(t)\,, \tag{8.58}$$

where $t_2 > 0$ is the first time (after $t = 0$) that $n_2(t)$ becomes 0.

For $c > c_L$ condition (8.56) is violated at $t = t_d(c)$ which depends on c, and 1-color vertices start to accumulate. As a result, the probability for contradictions becomes large, and backtracking enters into play.

8.5.3 Coloring in the Presence of Massive Backtracking

The analytical study of the complexity in the presence of backtracking is inspired by the analysis of the DPLL algorithm acting onto random 3-SAT (see Section 8.3). In the absence of solution, DPLL builds up a complete search tree before stopping. Obviously, the order in which the available colors of a vertex are tried does not affect the final shape of the tree. This allows us to study the evolution of the parallel (instead of sequential) growth process of the search tree (see Section 8.3.2 for detailed explanations).

As was pointed out before, due to color symmetry, the three-dimensional vector $\vec{N} = (N_1, N_2, N_3)$ describes the state of a graph under the action of the algorithm. Denoting by

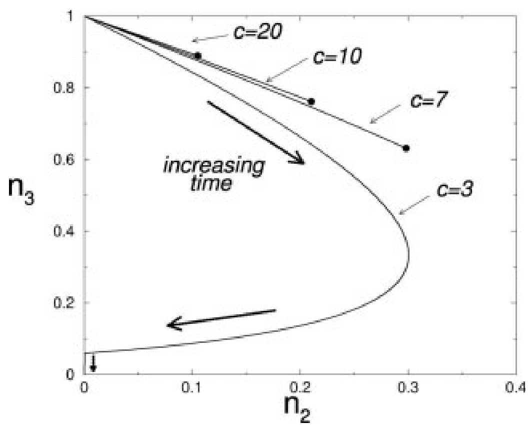


Figure 8.16: Trajectories of dominant search branches generated by DPLL in the uncolorable phase ($c > c_3 \simeq 4.7$ [18, 41]) compared to a search trajectory in the easy colorable phase ($c < c_L \simeq 3.85$). Horizontal and vertical axes represent the densities n_2 and n_3 of 2- and 3-color nodes respectively. Trajectories are depicted by solid curves, and the arrows indicate the direction of motion (increasing depth of the search tree); they originate from the left top corner, with coordinates $(n_2 = 0, n_3 = 1)$, since all nodes in the initial graph are 3-color nodes. Dots at the end of the uncol trajectories ($c = 7, 10, 20$) symbolize the halt point at which condition $n_2 < 3\ln 2/c$ ceases to be fulfilled, and the search tree stops growing. Note that as the initial connectivity increases, the trajectories halt at an earlier stage, implying the early appearance of contradictions as the problem becomes overconstrained (large connectivity values). The col trajectory (shown here for $c = 3$) represents the under-constrained region of the problem, where the very first search branch is able to find a proper coloring (bottom left corner with coordinates $(n_2 = 0, n_3 = 0)$).

$\bar{B}(\vec{N};T)$ the number of branches at time T with N_i $(i = 1, 2, 3)$ i-color vertices (the components of $\vec{N}$), the growth process of the search tree can be described by the evolution of $\tilde{B}(\vec{N};T)$ with time. Following the procedure exhibited in Section 8.3.2, we consider the evolution matrix

$$\bar{\mathbf{H}}(\vec{N},\vec{N}';T) = \sum_{w_2=0}^{N_3'} \binom{N_3'}{w_2} (\frac{c}{N})^{w_2} (1-\frac{c}{N})^{N_3} \delta_{N_3'-N_3-w_2} \Bigg\{ \{(1-\delta_{N_1'}) \sum_{w_1=0}^{N_2'} \binom{N_2'}{w_1} (\frac{2c}{3N})^{w_1} (1-\frac{2c}{3N})^{N_2'-w_1} \delta_{N_2-N_2'-(w_2-w_1)} \delta_{N_1-N_1'-w_1+1} + 2\delta_{N_1'} \sum_{w_1=0}^{N_2'-1} \binom{N_2'-1}{w_1} (\frac{2c}{3N})^{w_1} (1-\frac{2c}{3N})^{N_2'-w_1-1} \delta_{N_2-N_2'-(w_2-w_1-1)} \times \delta_{N_1-N_1'-w_1} \Bigg\} \tag{8.59}$$

where δ_N is the Kronecker delta function. The matrix describes the average number of branches with color vector $\vec{N}$ coming out from one branch with color vector $\vec{N}'$, as a result of the coloring of one vertex at step T. Note that (8.59) is written under the assumption that no 3-color nodes are chosen by the algorithm throughout the growth process. This assumption is consistent with the resultant solution which shows that in the uncolorable (uncol) region, $n_2(t)$, namely the number of 2-color vertices divided by N, keeps positive for all $t > 0$.

The generating function $G(\vec{y};T)$ of the number $\bar{B}(\vec{N};T)$ of branches satisfies an evolution equation similar to (8.22),

$$G(\vec{y};T+1) = e^{-y_1}\, G\big(\vec{\gamma}(\vec{y});T\big) + \big(2\, e^{-y_2} - e^{-y_1}\big)\, G\big(-\infty, \gamma_2(\vec{y}), \gamma_3(\vec{y});T\big) \quad (8.60)$$

where

$$\begin{aligned}
\gamma_1(\vec{y}) &= y_1\,, \\
\gamma_2(\vec{y}) &= y_2 + \frac{2c}{3N}\big(e^{y_1-y_2}-1\big)\,, \\
\gamma_3(\vec{y}) &= y_3 + \frac{c}{N}\big(e^{y_2-y_3}-1\big)\,.
\end{aligned} \quad (8.61)$$

To solve (8.60), we make scaling hypotheses for $\bar{B}$ and G, similar to those made in Section 8.3.3. Namely,

$$\bar{B}(\vec{N};T) = e^{N\,\omega(\vec{n};t)+o(N)}, \qquad G(\vec{y};T) = e^{N\,\varphi(\vec{y};t)+o(N)}, \quad (8.62)$$

where $\omega(\vec{n};t)$ is the logarithm of the number of branches $\bar{B}(\vec{N};T)$ divided by N and $\vec{n} = (n_1, n_2, n_3)$. As in Section 8.3.3, φ is the Legendre transform of ω. At the initial stage of the tree building up, there is a single outgoing branch from the root node, carrying a fully uncolored graph. Thus, $\bar{B}(\vec{N};T=0) = 1$ if $\vec{N} = (0,0,N)$, 0 otherwise, and $G(\vec{y},T=0) = e^{N\,y_3}$. The initial condition for function φ is simply, $\varphi(\vec{y};t=0) = y_3$. According to (8.55) both $N_2(T)$ and $N_3(T)$ are extensive in N; hence $n_2 > 0$ and $n_3 > 0$. Conversely, as soon as $N_1(T)$ becomes very large, contradictions are very likely to occur, and the growth process stops. Throughout the growth process, $N_1 = O(1)$ almost surely. Thus $n_1 = 0$ with high probability, and φ does not depend upon y_1. Independence of φ from y_1 allows us to choose the latter at our convenience, that is, as a function of y_2, y_3, t. Following the so-called kernel method [33], we see that equation (8.60) simplifies if $y_1 = y_2 - \ln 2$. Then, from ansatz (8.62), we obtain the following partial differential equation (PDE),

$$\frac{\partial\varphi}{\partial t}(y_2,y_3;t) = -y_2 + \ln 2 - \frac{c}{3}\frac{\partial\varphi}{\partial y_2}(y_2,y_3;t) + c\,\big(e^{y_2-y_3}-1\big)\frac{\partial\varphi}{\partial y_3}(y_2,y_3;t)\,. \quad (8.63)$$

This PDE can be interpreted as a description of the growth process of the search tree resulting from the algorithm operation. Through Legendre transformation, PDE (8.63) can be written as an evolution equation for the logarithm $\omega(n_2,n_3;t)$ of the average number of branches with densities n_2, n_3 of 2-, 3-colors nodes as the depth $t = T/N$ increases,

$$\frac{\partial\omega}{\partial t} = \frac{\partial\omega}{\partial n_2} + \ln 2 - \frac{c}{3}\,n_2 + c\,n_3\left[\exp\left(\frac{\partial\omega}{\partial n_3} - \frac{\partial\omega}{\partial n_2}\right) - 1\right]\,. \quad (8.64)$$

The surface ω, growing with "time" t above the plane n_2, n_3 describes the whole distribution of branches. Here, this distribution simplifies due to node conservation. The sum $n_2 + n_3$ of 2- and 3-color node densities necessarily equals the fraction $1 - t$ of not-yet colored nodes. Therefore, ω is a function of n_3 and t only, whose expression is obtained through exact resolution of PDE (8.63) with the above initial condition,

$$\begin{aligned}\omega(n_3;t) \quad &= \quad \frac{c}{6}\, t\,(1-2\,t-4\,n_3) - n_3\,\ln\,n_3 - (1-n_3)\,\ln\,(1-n_3) - \\ &\quad (1-t-n_3)\ln 2 + (1-n_3)\,\ln\left[3\,\left(1-e^{-\,2\,t\,c/3}\right)\right]. \qquad (8.65)\end{aligned}$$

Figure 8.17 exhibits $\omega(n_3, t)$ for $c = 10$.

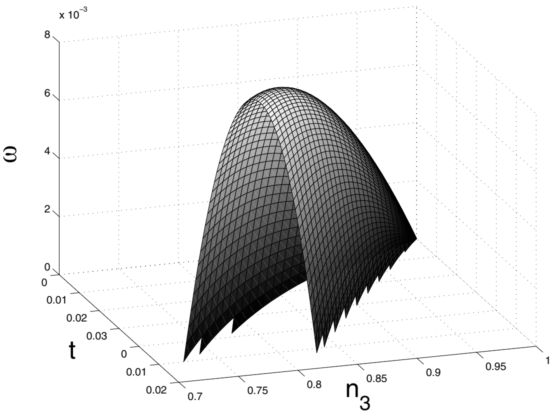


Figure 8.17: Function ω (log of number of branches with densities $n_2 = 1 - t - n_3$, n_3 of 2- and 3-color nodes at depth t in the search tree) as a function of n_3 and t for $c = 10$. The top of the curve at given time t, $\omega^*(t)$, is reached for the dominant branch 3-color density $n_3^*(t)$. The evolution of ω is shown until $t = t_h$ at which dominant branches in the search tree stop growing (die from the onset of contradictions). The maximal ω at t_h, $\omega^*(t_h)$, is our theoretical prediction for the complexity.

The maximum over n_2, n_3 of $\omega(n_2, n_3; t)$ at depth t in the tree

$$\omega^*(t) = \frac{c}{6}t^2 - \frac{c}{3}t - (1-t)\ln 2 + \ln\left[3 - \ e^{-2c\,t/3}\right] \qquad (8.66)$$

is reached at $n_3^*(t) = 2/(3\ e^{2\,c\,t/3} - 1)$, $n_2^*(t) = 1 - t - n_3^*(t)$, and gives the logarithm of the average number of branches at depth t divided by N (see Section 8.3.4 and explanations there). Under the action of the 3-GL algorithm, initially random 3-coloring instances become random mixed 2 and 3-coloring instances, where nodes can have either 2 or 3 colors at their disposal. This phenomenon indicates that the action of the 3-GL algorithm on random 3-coloring instances can be seen as an evolution in the n_2, n_3 phase-space (Figure 8.16). Each point (n_2, n_3) in this space, represents a random mixed 2 and 3-coloring instance, with an

Table 8.2: Analytical results and simulation results of the complexity ω for different connectivities c in the uncol phase. The analytical values of ω_{THE} are derived from theory; ω_{NOD} is obtained by measuring the average value of the search tree size.

c	ω_{THE}	ω_{NOD}
20	2.886×10^{-3}	$3 \times 10^{-3} \pm 3 \times 10^{-4}$
15	5.255×10^{-3}	$5.8 \times 10^{-3} \pm 5 \times 10^{-4}$
10	1.311×10^{-2}	$1.5 \times 10^{-2} \pm 1 \times 10^{-3}$
7	2.135×10^{-2}	$3. \times 10^{-2} \pm 3.6 \times 10^{-3}$

average number $(n_2 + n_3)N$ of nodes, and a fraction $n_3/(n_2 + n_3)$ of 3-color nodes. Parametric plot of $n_2^*(t), n_3^*(t)$ as a function of t represents the trajectories of dominant branches in Figure 8.16.

The halt condition, analogous to (8.41) for the DPLL algorithm, is $n_2^*(t) = 3 \ln 2/c$. It defines the endpoints of the dominant branch trajectories in the n_2, n_3 dynamical phase diagram of Figure 8.16. Call t_h the halt time at which the halt condition is fulfilled. The logarithm $\omega^*(t_h)$ of the number of dominant branches at $t = t_h$, when divided by $\ln 2$, yields our analytical estimate for the complexity of resolution, $\ln Q/N$.

To check our theory, we have run numerical experiments to estimate ω, the logarithm of the median solving time, as a function of the initial graph degree c. Table 8.2 presents results for ω as a function of the connectivity c in the uncol phase as found from numerical experiments and from the above theory. Note the significant decrease in the complexity as the initial connectivity increases. Agreement between theory and numerics is good but deteriorates at small c. However, the high computational complexity of the algorithm for small c values, does not allow us to obtain numerical results for large sizes N, and affects the quality of the large N extrapolation of ω.

In the uncol region, as c increases, contradictions emerge in an earlier stage of the algorithm, the probability that the same vertex appears in different branches reduces, and the analytical prediction becomes exact. As a consequence of the early appearance of contradictions, the complexity ω decreases with c. At very large c, we find

$$\omega(c) \asymp \frac{3 \ln 2}{2} \frac{1}{c^2} \simeq \frac{1.040}{c^2} , \tag{8.67}$$

and therefore that the (logarithm of the) complexity exhibits a power law decay with exponent 2 as a function of connectivity c.

8.6 Conclusions

In this chapter, we have explained a procedure for understanding and quantifying the complexity pattern of the backtrack resolution of the random decision problem, for which input distributions depend on a few control parameters. Under the action of the backtracking algorithm, the inputs are modified and additional control parameters must be introduced to model their distribution. The main steps in our approach are:

1. Identify the space of parameters in which the dynamical evolution takes place; this space will be generally larger than the initial parameter space since the algorithm modifies the instance structure. While the distribution of 3-SAT instances is characterized by the clause per variable ratio α only, another parameter p accounting for the emergence of 2-clauses has to be considered.

2. Divide the parameter space into different regions (phases) depending on the output of the resolution, e.g., SAT/UNSAT phases for 2+p-SAT.

3. Represent the action of the algorithm as trajectories in this phase diagram. Intersection of trajectories with the phase boundaries allow us to distinguish hard from easy regimes.

In addition, we have presented a quantitative study of the search tree growth, which allows us to accurately estimate the complexity of resolution in the presence of massive backtracking. From a mathematical point of view, it is worth noticing that monitoring the growth of the search tree requires a PDE, while ODEs are sufficient to account for the evolution of a single branch [2]. As shown in Section 8.4, the analysis of backtracking algorithms is not limited to the average-case complexity, but may also capture the distribution of resolution times [14,26].

Although the approach has been illustrated on the SAT and COL problems, it has already been applied to other decision problems, e.g., the vertex covering (VC) of random graphs [45]. In the VC problem, the parameter space is composed of the relative fraction of vertices which are allowed to be covered, x, and the average connectivity c of the graph. Following the three aforementioned steps, a complexity pattern of a branch-and-bound algorithm was obtained, yielding a distinction between exponential and linear regimes of the algorithm. The emerging pattern of complexity is similar to those of the DPLL algorithm for SAT and COL. The bound introduced in [45], was proved to significantly reduce the time consumption of the algorithm in the exponential regime, underlying the possibility of analyzing not only pure backtracking algorithms but also their improved bound-including versions.

In the light of the success of our method in investigating the performance of the DPLL algorithm, other not-yet studied backtracking algorithms, as well as more complicated heuristics, are future possibilities for effective analysis using this method. However, from a mathematical point of view, it would be very interesting to have better control of the dynamical annealing approximation underlying the analysis of the search tree in the presence of massive backtracking. The relative technical simplicity of the analysis of DPLL for the random 3-COL problem with respect to random 3-SAT, makes 3-COL a promising candidate for future rigorous studies [16]. The growth partial differential equation, monitoring the evolution of the search tree, is simpler than its 3-SAT counterpart, a consequence of the conservation law expressing that the sum of the numbers of colored and uncolored nodes remains constant throughout the search. We hope that progress towards greater rigor will be made in the near future.

Acknowledgments

Partial support from the ACI Jeunes Chercheurs “Algorithmes d’optimisation et systèmes désordonnés quantiques” is acknowledged. L.E. benefited from the post-doctoral funding program of the French Ministry of Research during year 2001/2002.

Notes added in Proofs:

- Section 8.2.4: the critical regime $\alpha \simeq \alpha_L$ has been recently investigated (C. Deroulers, R. Monasson, *Critical scaling of search heuristics and the unit-clause universality class*, preprint (2004)). The probability that UC or GUC succeeds in finding a solution (without ever backtracking) scaled as $\exp[-N^{1/6}\,\phi((\alpha-\alpha_L)\,N^{1/3})]$ where ϕ can be explicitly expressed in terms of the Airy function.

- Section 8.4.3: the power law behaviour of the complexity ω at large connectivities c depends on the number of colors ($Q = 3$ throughout Section 8.5). It is conjectured that ω decreases as $c^{-(Q-1)/(Q-2)}$ (R. Monasson, *On the analysis of backtrack procedures for the colouring of random graphs*, preprint (2004)).

References

[1] D. Achlioptas, L. Kirousis, E. Kranakis, and D. Krizanc, *Rigorous results for random (2+p)-SAT*, Theor. Comp. Sci. **265**, 109 (2001).

[2] D. Achlioptas, *Lower bounds for random 3-SAT via differential equations*, Theor. Comp. Sci. **265**, 159 (2001).

[3] D. Achlioptas, P. Beame, and M. Molloy, *A sharp threshold in proof complexity*, in *Proceedings of STOC 01*, 337 (2001).

[4] D. Achlioptas and E. Friedgut, *A sharp threshold for k-colorability*, Random Structures and Algorithms **14(1)**, 63 (1999).

[5] D. Achlioptas and C. Moore, *Almost all graphs with average degree 4 are 3-colorable*, Proc. on 34th Annual ACM Symposium on Theory of Computing, May 19-21, Montreal, Quebec, Canada, ACM, Montreal, 199 (2002)

[6] D. Achlioptas and M. Molloy, *Analysis of a list-colouring algorithm on a random graph*, Proc. of FOCS 97, 204 (1997).

[7] P. Beame, R. Karp, T. Pitassi, and M. Saks, *ACM symp. on theory of computing (STOC98)*, 561–571 Assoc. Comput. Mach., New York (1998).

[8] M.T. Chao and J. Franco, *Probabilistic analysis of two heuristics for the 3-satisfiability problem*, SIAM Journal on Computing **15**, 1106 (1986).

[9] M.T. Chao and J. Franco, *Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem*, Information Science **51**, 289 (1990).

[10] P. Cheeseman, B. Kanefsky, and M.W. Taylor, *Where the really hard problems are*, in J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th IJCAI*, 331, (Morgan Kaufmann Publishers, Inc., 1991).

[11] V. Chvàtal and E. Szmeredi, *Many hard examples for resolution*, Journal of the ACM **35**, 759 (1988).

[12] C. Coarfa, D.D. Dernopoulos, A. San Miguel Aguirre, D. Subramanian, and M.Y. Vardi, *Random 3-SAT: the plot thickens*, in R. Dechter, editor, *Proc. Principles and Practice of Constraint Programming (CP'2000)*, Lecture Notes in Computer Science **1894**, 143 (2000).

[13] S. Cocco and R. Monasson, *Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-Satisfiability problem*, Phys. Rev. Lett. **86**, 1654 (2001); *Analysis of the computational complexity of solving random satisfiability problems using branch and bound search algorithms*, Eur. Phys. J. B **22**, 505 (2001).

[14] S. Cocco and R. Monasson, *Exponentially hard problems are sometimes polynomial, a large deviation analysis of search algorithms for the random satisfiability problem, and its application to stop-and-restart resolutions*, Phys. Rev. E **66**, 037101 (2002); *Restarts and exponential acceleration of the Davis–Putnam–Loveland–Logemann algorithm: a large deviation analysis of the generalized unit clause heuristic for random 3-SAT*, to appear in Ann. Math. Artificial Intelligence (2003).

[15] S. Cocco, A. Montanari, R. Monasson, and G. Semerjian, *Approximate analysis of algorithms with physical methods*, preprint (2003).

[16] S. Cocco and R. Monasson, *Heuristic average-case analysis of backtrack resolution of random 3-Satisfiability instances*, to appear in Theor. Com. Sci. A (2004).

[17] J. Crawford and L. Auton, *Experimental results on the cross-over point in satisfiability problems*, *Proc. 11th Natl. Conference on Artificial Intelligence (AAAI-93),* 21–27, (The AAAI Press / MIT Press, Cambridge, MA, 1993); Artificial Intelligence **81** (1996).

[18] J.C. Culbersome and I.P. Gent, *Frozen development in graph coloring*, Theor. Comp. Sci. **265(1-2)**, 227 (2001).

[19] M. Davis, G. Logemann, and D. Loveland, *A machine program for theorem proving.* Communications of the ACM **5**, 394 (1962).

[20] L. Ein-Dor and R. Monasson, *The dynamics of proving uncolorability of large random graphs. I. symmetric colouring heuristic*, to appear in J. Phys. A (2003).

[21] R. Segdewick and P. Flajolet, *An introduction to the analysis of algorithms*, Chapter 3, (Addison-Wesley, Boston, 1995).

[22] J. Franco, *Results related to thresholds phenomena research in satisfiability: lower bounds*, Theor. Comp. Sci. **265**, 147 (2001).

[23] E. Friedgut, *Sharp thresholds of graph properties, and the k-sat problem*, Journal of the A.M.S. **12**, 1017 (1999).

[24] A. Frieze and S. Suen, *Analysis of two simple heuristics on a random instance of k-SAT*, Journal of Algorithms **20**, 312 (1996).

[25] M.R. Garey and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, (W.H. Freeman and Company, San Fransico, 1979).

[26] I.P. Gent and T. Walsh, *Easy problems are sometimes hard*, Artificial Intelligence **70**, 335 (1994).

[27] I. Gent, H. van Maaren, and T. Walsh, (eds). *SAT2000: Highlights of satisfiability research in the year 2000*, Frontiers in Artificial Intelligence and Applications, vol. **63**, (IOS Press, Amsterdam, 2000).

[28] C.P. Gomes, B. Selman, N. Crato, and H. Kautz, J. Automated Reasoning **24**, 67 (2000).

[29] R.B. Griffiths, C.-H. Weng, and J.S. Langer, *Relaxation times for metastable states in the mean-field model of a ferromagnet*, Phys. Rev. **149**, 301 (1966).

[30] J. Gu, P.W. Purdom, J. Franco, and B.W. Wah, *Algorithms for satisfiability (SAT) problem: a survey*. DIMACS Series on Discrete Mathematics and Theoretical Computer Science **35**, 19 (American Mathematical Society, 1997).

[31] T. Hogg and C.P. Williams, *The hardest constraint problems: a double phase transition*, Artificial Intelligence **69**, 359 (1994).

[32] A.C. Kaporis, L.M. Kirousis, and E.G. Lalas, *The probabilistic analysis of a greedy satisfiability algorithm*, Lecture Notes in Computer Science **2461**, 574 (2002).

[33] D.E. Knuth, *The Art of Computer Programming, vol. 1: Fundamental Algorithms*, (Addison-Wesley, New York, 1968).

[34] D.E. Knuth, *Selected Papers on Analysis of Algorithms*, Center for the Study of Language and Information, Lecture Notes **102**, Stanford CA (2000).

[35] G. Lopez, *Partial Differential Equations of First Order and Their Applications to Physics*, (World Scientific, Singapore, 1999).

[36] A. McKane, M. Droz, J. Vannimenus, and D. Wolf (eds), *Scale Invariance, Interfaces, and Non-equilibrium Dynamics*, Nato Asi Series B: Physics **344**, (Plenum Press, New York, 1995).

[37] D. Mitchell, B. Selman, and H. Levesque, *Hard and easy distributions of SAT problems, Proc. of the Tenth Natl. Conf. on Artificial Intelligence (AAAI-92)*, 440, (The AAAI Press / MIT Press, Cambridge, MA, 1992).

[38] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Determining computational complexity from characteristic 'phase transitions'*, Nature **400**, 133 (1999); *2+p-SAT: relation of typical-case complexity to the nature of the phase transition*, Random Structure and Algorithms **15**, 414 (1999).

[39] A. Montanari and R. Zecchina, *Boosting search by rare events*, Phys. Rev. Lett. **88**, 178701 (2002).

[40] R. Motwani and P. Raghavan, *Randomized Algorithms*, (Cambridge University Press, Cambridge, 1995).

[41] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, *Coloring random graphs*, Phys. Rev. Lett. **89**, 268701 (2002).

[42] R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*, (Addison-Wesley, New York, 1996).

[43] B. Selman and S. Kirkpatrick, *Critical behavior in the computational cost of satisfiability testing*, Artificial Intelligence **81**, 273 (1996).

[44] J.S. Turner, *Almost all k-colorable graphs are easy to color*, Journal of Algorithms **9**, 63 (1988).

[45] M. Weigt and A.K. Hartmann, *Typical solution time for a vertex-covering algorithm on finite-connectivity random graphs*, Phys. Rev. Lett. **86**, 1658 (2001).

[46] N. Wormald, *Differential equations for random processes and random graphs*, Ann. Appl. Probab. **5**, 1217 (1995).

9 New Iterative Algorithms for Hard Combinatorial Problems

Riccardo Zecchina

We provide an elementary introduction to a new class of iterative algorithm capable of dealing with the proliferation of metastable states in some hard optimization problems. Such algorithms can be viewed as a generalization of the message-passing algorithms used in error-correcting codes and are based on the single-sample implementation of the cavity equations in presence of multiple states.

9.1 Introduction

Phase transitions in random Constraint Satisfaction Problems (CSP) and the associated onset of exponential regimes in search algorithms are at the root of the modern typical-case theory of computational complexity [16]. CSP deal with an ensemble of n discrete variables which have to satisfy m constraints, all at the same time. Each constraint can take different forms depending on the problem under study: well known examples are the K-Satisfiability (K-SAT) problem in which constraints are an 'OR' function of K variables in the ensemble (or their negations) and the Graph Q-coloring problem in which constraints simply enforce the condition that the endpoints of the edges in the graph must not have the same color (among the Q possible ones). Quite generally a generic CSP can be written as the problem of finding a zero-energy ground state of an appropriate energy function, and its analysis amounts to performing a zero-temperature statistical physics study.

Problems which are of central interest for their intrinsic hardness are those belonging to the so called NP-complete class [11, 19]: by definition, the existence of an efficient algorithm for solving one of them, in its worst-case instances, would immediately lead to other algorithms for solving efficiently thousands of different hard combinatorial problems. During the last decade, much effort has been put into the study of random realizations of such hard CSP [12]. Numerical observations have shown that often NP-complete problems are easy to solve when generated at random or in real-world contexts. Running times of combinatorial algorithms display a huge variability and a theory for their typical behavior represents the natural complement to the worst-case analysis. Moreover, the connections between worst-case complexity and the average case one is the building block of modern cryptography: on the one hand the RSA system is based on factoring large integers, a problem which is believed to be hard on average. On the other hand, alternative cryptographic systems have recently been proposed which rely on a worst-case/average-case equivalence theorem of Ajtai [3, 4] for some hidden vector problems in high-dimensional lattices. Roughly speaking, the theorem states

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

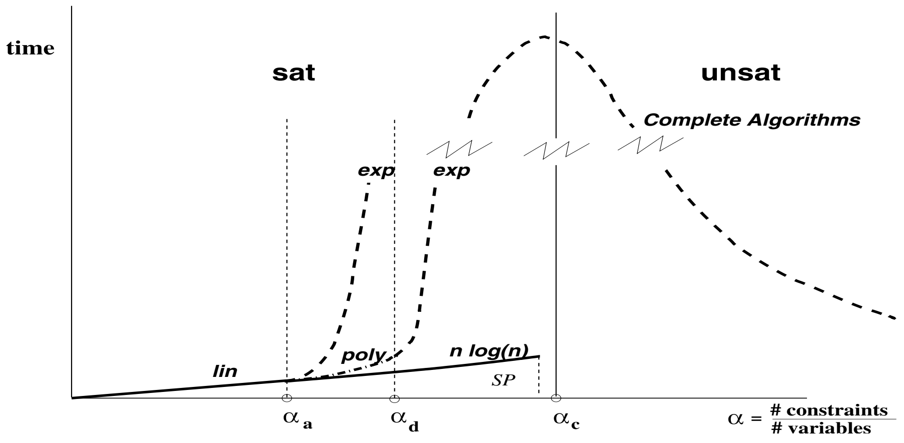


Figure 9.1: Typical behavior of algorithms: for low α most algorithms take linear time to find solutions. Next there appear transition points where the different algorithms start to behave exponentially. Finally below α_c ($\alpha \in [\alpha_d, \alpha_c]$) there appears a clustering transition and local search algorithms get trapped in the exponentially more numerous states of non-zero energy. The SP algorithm still finds solution in polynomial time.

that the existence of a polynomial probabilistic algorithm for random instances would lead to an efficient algorithm also for the worst-case instances which in turn are known to be hard.

The study of the random realization of CSP is known as "typical-case complexity theory" [6,12,16,21,25] and consists of specifying a probability measure over problem instances. Such a scenario perfectly fits within the framework of spin-glass theory and makes the tools of statistical physics of disordered systems a powerful instrument for the study of frustration effects in optimization problems [27,32,33].

Two main questions are in order [12]. The first is of algorithmic nature and asks for an algorithm which decides whether, for a given CSP instance, all the constraint can be simultaneously satisfied (in which case the problem is said to be SAT) or not (problem UNSAT). The second question is more theoretical and deals with large random instances, for which one wants to know the structure of the solution space and predict the typical behavior of classes of algorithms.

Numerical simulations have shown that a typical scenario arises in many hard random CSP (e.g. in K-SAT, Q-coloring [9,29,31]). First, a phase transition is found when the ratio $\alpha = m/n$ (for $n \to \infty$) of the number of random constraints to the number of variables is increased. For $\alpha < \alpha_c$ the generic random problem is satisfiable (SAT), for $\alpha > \alpha_c$ the generic problem is not satisfiable (UNSAT). Secondly, below α_c there exist regions in which algorithms start to behave exponentially [14] (see Figure 9.1).

Very recently it has been shown [31] that the statistical physics approach known as the "cavity method" [28] (which is the generalization to disordered systems of the iterative method used to solve exactly the Bethe lattice Ising problem with non-trivial boundary conditions [5]) could be applied to the study of single realizations of random problem instances. Such an analysis has allowed to develop, in Ref. [31], a new class of combinatorial algorithms called

Survey Propagation (SP) which are able to solve efficiently hard instances of random CSP. Further improvements of SP have been given in Ref. [8] and applications to K-SAT, Graph Coloring and Hyper-Graph Bi-coloring can be found in Refs. [9,10,29,31]. The SP algorithms are a generalization of the so called message-passing or Belief Propagation (BP) algorithms used in error correcting codes [40].

The basic reason why an iterative procedure such as SP may provide optimal results, resides in the fact that the kind of CSP that are studied have a cost-energy function whose interactions lie on a diluted random graph (or hyper-graph). In such structures loops are indeed present and are the source of frustration. However, their typical length slowly diverges with n (as $\log n$), a fact which allows to neglect the effects of correlated fluctuations and to set up a closed set of iterative equations for the marginal probabilities of the variables. Such probabilities are defined over the set of degenerate clusters of optimal configurations, that is in a 1-step replica symmetry-breaking phase [27].

In the following we shall briefly review the analytic results on the most studied random CSP problem, random K-SAT, and provide a reformulation of the SP algorithm for the random K-SAT problem in a general formalism which can be easily exported to other problems. The description of SP will be presented as a way of generalizing and improving the performance of the more traditional message-passing algorithms, such as BP, through a detailed comparison of the iterative equations. This will allow us to discuss how SP is capable of dealing with the appearance of clustering in the solution space of K-SAT. The decimation process which uses the information provided by the SP procedure to solve specific instances of random K-SAT completes the description of the algorithm.

9.2 Combinatorial Decision Problems, K-SAT and the Factor Graph Representation

NP-complete combinatorial decision problems can be described as discrete models with energy functions bounded below (say by $zero$) for which one wants to find the optimal ground states. A very simply stated problem is K-SAT:

Given a vector of $\{0,1\}$ Boolean variables $\mathbf{x} = \{x_i\}_{i\in I}$ where $I = \{1,\dots,n\}$, consider a SAT formula defined by

$$\mathcal{F}(\mathbf{x}) = \bigwedge_{a\in A} C_a(\mathbf{x})$$

where A is an arbitrary finite set (disjoint with I) labeling the clauses C_a; $C_a(\mathbf{x}) = \bigvee_{i\in I(a)} J_{a,i}(x_i)$; any *literal* $J_{a,i}(x_i)$ is either x_i or $\sim x_i$ ("not" x_i); and finally, $I(a) \subset I$ for every $a \in A$. Similarly to $I(a)$ we can define the set $A(i) \subset A$ as $A(i) = \{a : i \in I(a)\}$, that is the set of clauses containing variable x_i or its negation.

.

We will use the "factor graph" representation: given a formula $\mathcal{F}$, we will define its associated *factor graph* as a bipartite undirected graph $G = (V; E)$, having two types of nodes, and edges only between different types of nodes (see Figure 9.2):

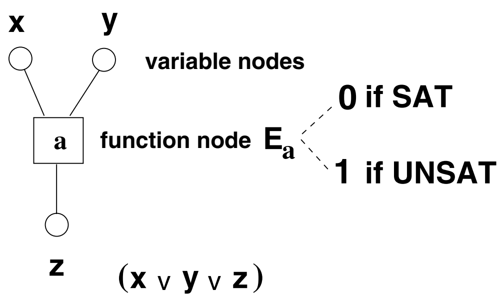


Figure 9.2: Variable nodes and function node corresponding to one clause. The energy is positive if the clause is UNSAT.

- Variable nodes, each one labeled by a variable index in $I = \{1, \ldots, n\}$.
- Function or factor nodes, each one labeled by a clause index $a \in A$.
- An edge (a, i) will belong to the graph if and only if $a \in A(i)$ or equivalently $i \in I(a)$.

In other words, $V = A \cup I$ and $E = \{(i, a) : i \in I, a \in A(i)\}$ or equivalently $\{(i, a) : a \in A, i \in I(a)\}$.

Given a formula $\mathcal{F}$, the problem of finding a variable assignment $\mathbf{s}$ such that $\mathcal{F}(\mathbf{s}) = 1$ if it exists, can also be written as a spin-glass problem as follows: if we consider a set of n Ising spins, $\sigma_i \in \{\pm 1\}$ in place of the Boolean variables ($\sigma_i = -1, 1 \leftrightarrow x_i = 0, 1$) we may write the energy function associate to each function node as follows:

$$E_a = \prod_{r=1}^{K} \frac{(1 + J_{a,i_r} \sigma_{i_r})}{2} . \tag{9.1}$$

where $J_{a,i} = -1$ (resp. $J_{a,i} = 1$) if x_i (resp. $\tilde{x}_i$) appears in clause a. The total energy of a configuration $\sigma_1, \ldots, \sigma_n$ $E = \sum_{a=1}^{|A|} E_a$ is nothing but a 3-spin spin-glass model.

K-SAT plays a central role in computer science, and a lot of effort has been devoted to this problem. As soon as there are clauses with $K \geq 3$ variables this problem is in fact NP-complete [11, 19].

9.2.1 Random K-SAT

A K-SAT formula is a SAT formula having $|I(a)| = K$ for each $a \in A$. The random K SAT formula distribution is obtained by picking a formula with $m \equiv |A|$ clauses, each one chosen independently and uniformly between all K-tuples of variables (no variable repetition), in which negations are distributed afterwards randomly with a fair coin. The factor graph which arises from this construction is a bipartite random graph with loops of typical length $O(\log n)$ and variable nodes with a Poisson connectivity distribution of mean $K\alpha$ (see Figure 9.3).

We will be interested in solving formulas from this distribution for large m, n, but $lim_{n \to \infty} \frac{m}{n} = \alpha$. This problem displays a very interesting threshold phenomenon when one

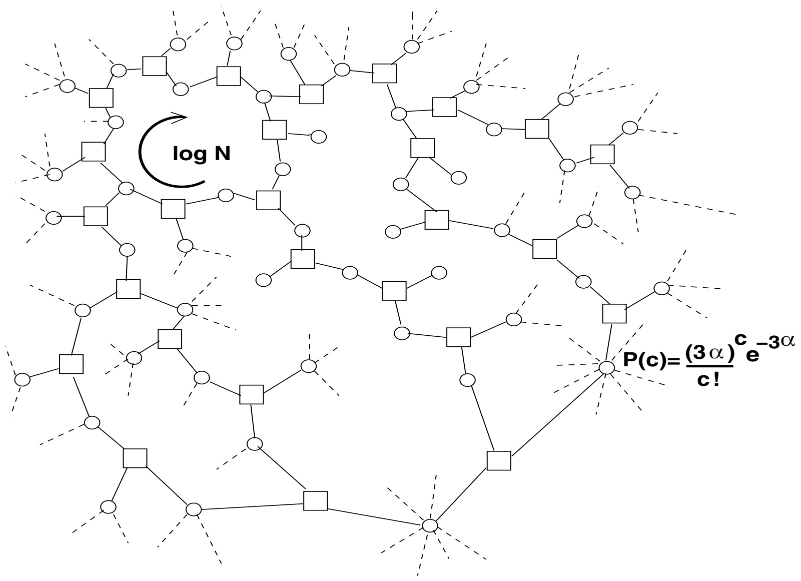


Figure 9.3: The factor graph of a random 3-SAT formula. The most numerous loops have length scaling as $\log n$ and the variable nodes have Poisson connectivity distribution.

takes the large-n limit. There exists a phase transition at a value $\alpha_c(K)$ of this ratio. For $\alpha < \alpha_c(K)$ the generic problem is satisfiable (SAT), for $\alpha > \alpha_c(K)$ the generic problem is not satisfiable (UNSAT).

This phase transition can be seen numerically [23] and plays an important algorithmic role in that difficult instances are found to accumulate close to α_c [34].

On the analytical side, there exists a proof that the threshold phenomenon exists at large n [17], although the fact that the corresponding α_c has a limit when $n \to \infty$ has not yet been established rigorously. Upper bounds $\alpha_{UB}(K)$ on α_c have been found using first-moment methods [15, 24] and variational interpolation methods [18, 20], and lower bounds $\alpha_{LB}(K)$ have been found using either explicit analysis of some algorithms [1], or some second moment methods [2]. The large-K limit is particularly interesting and one knows from [2, 15] that, at first order in the expansion in powers of 2^{-K}, the bounds scale as

$$\alpha_{LB}(K) = \log(2)2^K - \left(\frac{K+1}{2}\log(2) + 1 + o(1)\right) + \mathcal{O}(2^{-K}) \tag{9.2}$$

$$\alpha_{UB}(K) = \log(2)2^K - \frac{1+\log(2)}{2} + \mathcal{O}(2^{-K}) \tag{9.3}$$

where $o(1)$ is a term which goes to 0 at large K.

Recently, the techniques of statistical physics [27] have been applied to this problem [16, 29, 31–33] leading to a an increasingly clear understanding of the geometrical structure of the space of solutions in the vicinity of the phase boundary, where hard instances accumulate. Even more recently, a systematic use of the the cavity method has allowed to compute the

SAT/UNSAT thresholds for any K [26] and the detailed study of the stability of the cavity solutions [26, 35] has provided a more clear understanding of the internal structure of the hard-SAT phase (see Chapter 7 in this book for a brief review on the subject.)

The cavity calculations are non-rigorous, however, the self-consistency of its assumptions can be checked, and its predictions can be tested against numerical simulations. In simpler cases such as that of the random K-XORSAT problem, where a phase transition similar to the one of random K-SAT is found, the results of the cavity solution can be confirmed rigorously [13, 30]. Finally, the application to random K-SAT of the variational interpolation method of Guerra [20] has allowed to prove that, for even K, the cavity results for $\alpha_c(K)$ are a rigorous upper bound [18].

The generic phase diagram of the SAT region arising from the cavity calculation is as follows.

For $\alpha < \alpha_d$, the set of all satisfying assignments $S_{\mathcal{F}}$ is connected, that is, one can find a path to go from any assignment to any other assignment requiring short steps only (in Hamming distance). No variables are constrained to take the same value in all satisfying assignments and a solution can be found by simple greedy algorithms. For $\alpha_d < \alpha < \alpha_c$, $S_{\mathcal{F}}$ becomes divided into subsets which are far apart in Hamming distance and which are composed of the same number of solutions (up to sub-exponential corrections). The number of such clusters is exponential in n and its logarithm (divided by n) is known as the *complexity*. The lower α part of this region, $\alpha_d < \alpha < \alpha_S$ presents further structure: clusters are organized in *families* with a characteristic distribution of mutual distances [35], that is, there appears effects of multiple steps of replica symmetry breaking. α_S is the point below which the 1-step cavity solution becomes unstable with respect to two steps of replica symmetry breaking, i.e., further clustering effects. Such complicated structure is captured by more sophisticated cavity equations [35] which in turn could be directly transformed into an algorithmic tool, as is done by SP in the 1-step regime. However, for the sake of brevity, we shall not discuss this issue in this chapter.

Although in the $[\alpha_d, \alpha_c]$ region there exists an exponentially large number of clusters, each containing an exponentially large number of solutions, it is very difficult to find one solution because of the proliferation of “metastable” clusters [29, 31]. A metastable cluster is a cluster of assignments which all have the same fixed number C of violated clauses, and such that one cannot find at a small Hamming distance of this cluster an assignment which violates strictly less than C clauses.

The metastable clusters with $C > 0$ are exponentially (in n) more numerous than the satisfiable clusters. One can then expect that local search algorithms will generically get trapped in these much more numerous metastable clusters. We call the range $\alpha_d < \alpha < \alpha_c$ the **hard-sat** region.

At the SAT/UNSAT threshold $\alpha = \alpha_c$ the complexity vanishes ($\Sigma(\alpha_c) = 0$) and for $\alpha > \alpha_c$ the instances are almost always unsatisfiable.

It is worth mentioning that the numerical values of C for the metastable clusters is so small [35] that one needs n sufficiently large to detect their presence (e.g., for 3-SAT, $\frac{C}{n} \sim 10^{-4}$ at α_c). Finite size effects are indeed strong and are possibly responsible for some confusion in the recent literature reporting simulation results for search algorithms on random K-SAT.

A summary of the values of the different thresholds from the cavity calculations of Ref. [26] are given in Table 9.1.

Table 9.1: The values of different thresholds from the cavity calculations of Mertens et al. [26].

K	α_d	α_s	α_c
3	$\mathbf{3.927} \pm 0.004$	**4.15**	**4.267**
4	$\mathbf{8.297} \pm 0.008$	**9.08**	**9.931**
5	$\mathbf{16.12} \pm 0.02$	**17.8**	**21.117**
6	$\mathbf{30.50} \pm 0.03$	**33.6**	**43.37**
7	$\mathbf{57.22} \pm 0.06$	**62.5**	**87.79**

In the large-K limit, the random K-SAT problem simplifies and one can get more precise analytical results. The perturbative expansion in powers of $\varepsilon \equiv 2^{-K}$ gives, for the threshold value

$$\alpha_c = \ln(2)2^K - \frac{\ln(2)+1}{2} + \mathcal{O}(\varepsilon)\,. \tag{9.4}$$

Note that this result saturates the best known *rigorous upper bound* for α_c found in [15, 24]. Similarly one may compute the dynamical and the stability points. One finds [26]

$$\alpha_d = \frac{2^K}{K}\left(\ln K + d^\star\right) \mathrm{e}^{\frac{\mathrm{e}^{-d^\star}}{2}} \tag{9.5}$$

and

$$\alpha_S = \frac{2^K}{K}(\ln(2K) + d^\star(2K))\mathrm{e}^{\frac{1}{4}\mathrm{e}^{-[d^*(2K)]}} \tag{9.6}$$

where $d^\star$ denotes the solution of

$$\exp(d^\star) = \frac{1}{2}\left(\ln K + d^\star\right) \tag{9.7}$$

The relative behavior of α_S and α_d is given by:

$$\frac{\alpha_S}{\alpha_d} \simeq 1 + \frac{\ln 2 - 1/2}{\ln K} \tag{9.8}$$

For any finite K, there exists a region between α_d and α_S in which the 1-step solution is unstable, while the solution at $\alpha_S < \alpha \leq \alpha_C$ is stable. At large K the unstable region is small and $\lim_{K\to\infty} \frac{\alpha_d(K)}{\alpha_S(K)} = 1$.

The qualitative phase diagram of random K-SAT ($K > 2$) is displayed in Figure 9.4.

In the clustering region, there exist exponentially numerous metastable states at positive energy which are responsible for the slowing down in local search algorithms.

The case of 3-SAT is slightly more involved: in the low-α phase – the replica symmetric (RS) phase – looking carefully at the fluctuations of variables, one finds an additional clustering phenomenon at $\alpha = 3.87$ [37] where a continuous RSB transition appears. In the region $\alpha \in [3.87, 3.927]$ variables are never frozen (one may flip any variable and still find another satisfying assignment at a short distance) and yet the solution space presents a non-trivial distribution of typical mutual distances. At $\alpha = 3.927$ frozen variables set in and the unfrozen clustering is wiped away.

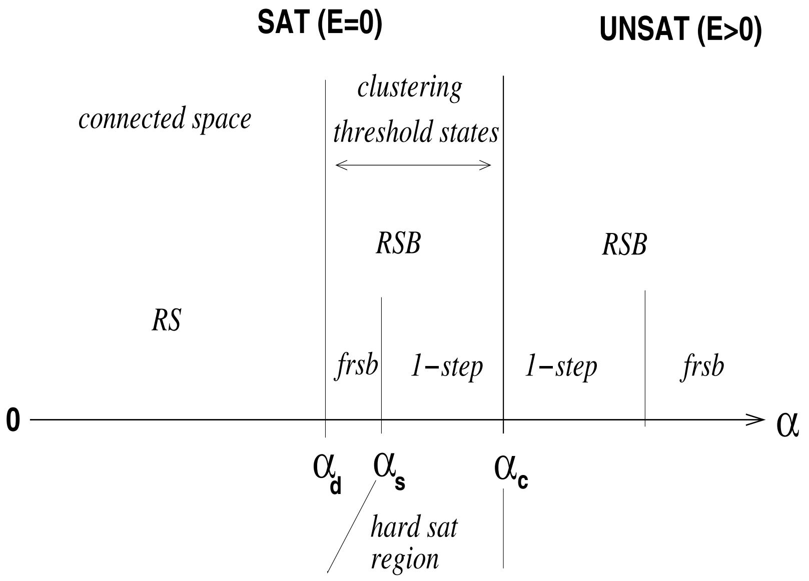


Figure 9.4: Qualitative phase diagram of random K-SAT

For $K > 3$ such a transition is absent and there is no numerical evidence that such a transition point plays any computational role in slowing down local search algorithms.

In the following we shall try to give an heuristic derivation of the SP equations without making any reference to the statistical physics derivation which can be found in many recent publications, e.g., [31]. We shall rather provide an intuitive probabilistic characterization based on clear, finite-n definitions.

9.3 Growth Process Algorithm: Probabilities, Messages and Their Statistics

In order to provide a simple justification of the SP (or BP) iterative probabilistic approach, we first discuss an exact enumerative (exponential) algorithm and next show how it can be approximated and simplified by computing marginal probabilities over satisfying assignments, leading to the BP and SP iterative equations.

Define $I_k = \{1, \ldots, k\}$, $A_k = \{a \in A : I(a) \subset I_k\}$ ($A_k \subset A$ labels the clauses involving the variables in $I_k \subset I$) and $\mathcal{F}_k, G_k$ its associated formula and factor graph.

Define $S_{\mathcal{F}} = \mathcal{F}^{-1}(\{1\}) \subset \{0,1\}^n$ as the set of satisfiable instances of $\mathcal{F}$ and consider the following inductive procedure to compute $S_k = S_{\mathcal{F}_k}$. Clearly $S_k \subset S_{k-1} \times \{0,1\}$, and

$$\begin{aligned} S_0 &= \emptyset \\ S_k &= \{(\mathbf{s}, s_k) : \mathbf{s} \in S_{k-1}, s_k \in \{0,1\} / C_a(\mathbf{s}, s_k) = 1 \text{ for } a \in A_k \setminus A_{k-1}\} \end{aligned} \tag{9.9}$$

Observe that $A_k \setminus A_{k-1} \subset I(k)$, so typically only a small number of checks have to be done for every s to take it from S_{k-1} to S_k. This procedure can help us compute $S_n = S$ iteratively: at each step we add a variable to the graph, and then "filter out" contradictory configurations (i.e., configurations which are UNSAT once the last var was added).

Of course this procedure is typically exponential (in time and space) in n, partially because the set S can be (and typically is) exponentially large.

A more efficient solution would be to carry on less information instead. Consider

$$P_S\left(s_i = v\right)$$

where P_S is the uniform probability measure over the set S and $v = 0, 1$. If we can compute these quantities, then we can certainly use this information to get a single $s \in S$ (we will see exactly how in Section 9.6).

Trying to mimic Eq. (9.9), we see that if we get to know the joint probability distribution of variables $\{s_j\}_{j \in I(A_i(i))\setminus\{i\}}$ on the space S_{i-1} (i is the variable added at the i-th step and $A_i(i)$ is the set of clauses touching that variable, hence j runs over the neighboring variables of i) then clearly we can trivially rebuild the state of these variables in any S_{i-1} configuration, and then compute the statistics for s_i on the extended space including this variable, i.e., $P_{S_i}(s_i)$.

The basic assumption behind the algorithm we are describing is that these variables $\{s_j\}$ ($j \in I(A_i(i)) \setminus \{i\}$) are weakly correlated (completely uncorrelated in the limit $n \to \infty$).

In this case, we only need to know $P_{S_{i-1}}\left(s_j = 0, 1\right)$ for $j \in I\left(A\left(i\right)\right) \setminus \{i\}$ to compute $P_{S_i}\left(s_i = 0, 1\right)$ and so we can write an explicit equation relating these quantities.

Let us assume that we have computed all $P_{S_{i-1}}$ (we will drop the S_{i-1} index for notational simplicity), and we want to compute the statistics for s_i in S_i (we will see later how this can be used to obtain a good recursion).

In order to do so explicitly, we will define the following quantities: given a configuration for $\{s_j\}_{j \in I(a)\setminus\{i\}}$ on S_{i-1} we will denote the set-valued variable $u_{a\to i} \subset \{0, 1\}$ as the subset of available (SAT) configurations for the last variable s_i, i.e.:

$$u_{a\to i}\left(\{s_j\}_{j\in I(a)\setminus\{i\}}\right) = \left\{s_i \in \{0,1\} : C_a\left(\{s_j\}_{j\in I(a)}\right) = 1\right\} \tag{9.10}$$

Note that given the particular structure of the SAT problem, any clause can be satisfied by a given participating variable (if chosen to an appropriate value), disregarding all other ones in that clause, i.e. $J_{a,i}(1) \in u_{a\to i}$ always, and that eliminates $\emptyset$ and $\{\sim J_{a,i}(1)\}$ from possible u outputs (meaning that effectively each u is a two-valued variable).

We will have that $u_{a\to i} = \{J_{a,i}\left(1\right)\}$ when the clause is not already satisfied by the remaining participating variables, and $u_{a\to i} = \{0, 1\}$ otherwise. Then

$$\begin{aligned} P\left(u_{a\to i} = \{J_{a,i}\left(1\right)\}\right) &= \prod_{j\in I_i(a)\setminus\{i\}} P\left(s_j =\sim J_{a,j}(1)\right) \\ P\left(u_{a\to i} = \{\sim J_{a,i}\left(1\right)\}\right) &= 0 \\ P\left(u_{a\to i} = \emptyset\right) &= 0 \\ P\left(u_{a\to i} = \{0,1\}\right) &= 1 - \prod_{j\in I_i(a)\setminus\{i\}} P\left(s_j =\sim J_{a,j}(1)\right) \end{aligned} \tag{9.11}$$

The available states for variable s_i will be given by the set

$$h_i = \bigcap_{a \in A_i(i)} u_{a \to i} \tag{9.12}$$

Following the above equation we may easily compute the statistics for h_i:

$$\begin{aligned}
P(h_i = \{0,1\}) &= \prod_{a \in A_i(i)} P(u_{a \to i} = \{0,1\}) \\
P(h_i = \{v\}) &= \prod_{a \in A_i(i)} [P(u_{a \to i} = \{0,1\}) + P(u_{a \to i} = \{v\})] \\
&\quad - \prod_{a \in A_i(i)} P(u_{a \to i} = \{0,1\}) \\
P(h_i = \emptyset) &= 1 - P(h_i = \{0,1\}) - P(h_i = \{0\}) - P(h_i = \{1\})
\end{aligned} \tag{9.13}$$

Once we have computed the statistics for h_i on S_{i-1}, it is straightforward to compute the statistics for s_i on S_i:

- each S_{i-1} configuration leading to $h_i = \{0,1\}$ will be split on two new S_i configurations with, respectively, $s_i = 0$ and 1;
- each S_{i-1} configuration leading to $h_i = \{v\}$ will correspond to a single S_i configuration with $s_i = v$, and finally:
- each configuration leading to $h_i = \emptyset$ will be eliminated or ignored.

Explicitly:

$$P_{S_i}(s_i = v) = \frac{P(h_i = \{v\}) + P(h_i = \{0,1\})}{P(h_i = \{0\}) + P(h_i = \{1\}) + 2P(h_i = \{0,1\})} \tag{9.14}$$

Note that the normalization is needed here because each configuration of the input variables can produce from zero up to two output configurations in the extended space. The above formula is the basic iteration of the Belief Propagation (BP) message-passing algorithm [38], used, for instance, as a decoding tool in low-density parity check codes [40].

The problem in using Eq. (9.14) to build up a recursion is that we have successfully computed $P_{S_i}(s_i)$ but we need also all other $P_{S_i}(s_j)$ for $j < i$. This problem is due to the fact that the quantities we can put together in an equation, namely $P_{S_{i-1}}(s_j)$ and $P_{S_i}(s_i)$ are rather different in nature when we observe them on the same set S_i: while the former are quantities "ignoring" variable i, the latter is a "complete" quantity.

9.4 Traditional Message-passing Algorithm: Belief Propagation as Simple Cavity Equations

Surprisingly enough, this problem is easy to solve almost without changing the obtained equation: it suffices to work always with "incomplete" quantities. We can define the *cavity* objects

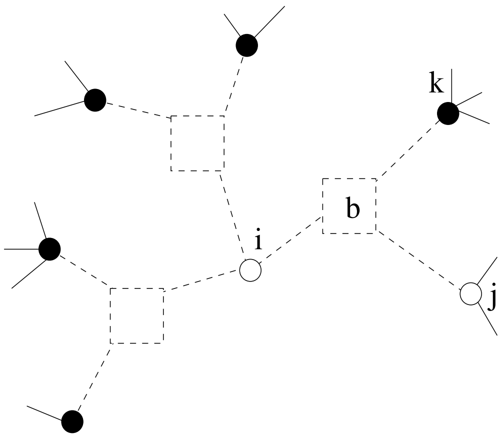


Figure 9.5: We use as inputs $P^{(i)}(\bullet)$, corresponding to cavity probabilities ignoring var i. The computed probabilities $P^{(j)}(i)$ will be the ones ignoring var j.

$I^{(i)} = \{1, \ldots, n\} \setminus \{i\}$, $A^{(i)} = \left\{a \in A : I(a) \subset I^{(i)}\right\}$, $I^{(i)}(a) = I(a) \cap I^{(i)}$ and $\mathcal{F}^{(i)}$, $S^{(i)}$ the associated formula and solution space, and try to define the equation relating statistics over $S^{(i)}$ (see Figure 9.5).

Now the "output" computed statistics for s_i would need also to be done with respect to some $S^{(j)}$ to obtain "closed" equations, and so we will have to single out another neighboring variable $j \in I(b)$ for $b \in A(i)$ (note that as all others $k \in I(b)$ will become disconnected from i, they will not enter into the equations, so we can simply eliminate b). Then Eqs. (9.11) become

$$\begin{aligned}
P^{(i)}\left(u_{a\to i} = \{J_{a,i}(1)\}\right) &= \prod_{h\in I(a)\setminus\{i\}} P^{(i)}\left(s_h =\sim J_{a,h}(1)\right)\\
P^{(i)}\left(u_{a\to i} = \{\sim J_{a,i}(1)\}\right) &= 0\\
P^{(i)}\left(u_{a\to i} = \emptyset\right) &= 0\\
P^{(i)}\left(u_{a\to i} = \{0,1\}\right) &= 1 - \prod_{h\in I(a)\setminus\{i\}} P^{(i)}\left(s_h =\sim J_{a,h}(1)\right) \qquad (9.15)
\end{aligned}$$

and Eqs. (9.13) become:

$$\begin{aligned}
P^{(j)}\left(h_i = \{0,1\}\right) &= \prod_{a\in A(i)\setminus\{b\}} P^{(i)}\left(u_{a\to i} = \{0,1\}\right)\\
P^{(j)}\left(h_i = \{v\}\right) &= \prod_{a\in A(i)\setminus\{b\}} \left[P^{(i)}\left(u_{a\to i} = \{0,1\}\right) + P^{(i)}\left(u_{a\to i} = \{v\}\right)\right]\\
&\quad - \prod_{a\in A(i)\setminus\{b\}} P^{(i)}\left(u_{a\to i} = \{0,1\}\right) \qquad (9.16)
\end{aligned}$$

$P^{(j)}\left(h_i = \emptyset\right)$ can be computed using the normalization condition.

It is worth noticing that, for n small enough, in the case of random K-SAT, formulas will have a considerable relative number of pairs of clauses sharing more than one variable and other types of short loops, e.g., for 3-SAT one finds on average for large n, $9\alpha^2$ pairs of clauses sharing two variables. The presence of such structures invalidates the cavity procedure which assumes independence of input pdfs. Correct equations can be recovered by considering effective function nodes in which the dangerous structures are substituted by single multi-variable constraints: for instance, a pair of 3-clauses sharing two variables can be glued in a single function node depending on the four distinct variables. The energy of the node is the sum of the energies of the two initial clauses taken together in the cavity procedure.

In order to actually use the above BP equations to find the required probability distributions we will parameterize $\eta_{i,j} = P^{(j)}(s_i)$ and use Eq. (9.16) together with (9.14) to obtain an operator $\Lambda : \{\eta_{i,j}\}_{i\in I, j\in I(A(i))} \mapsto \{\eta'_{i,j}\}_{i\in I, j\in I(A(i))}$. We will look for a fixed point for this operator, that can be reached by

$$\left\{\eta_{i,j}^{fix}\right\} = \lim_{t\to\infty} \overbrace{\Lambda \circ \cdots \circ \Lambda}^{t\,\text{times}} \left(\{\eta_{i,j}^0\}\right)$$

for some random initial $\{\eta^0\}$.

These equations can be easily implemented, and extensive experimentations and stability analysis show that they indeed converge in the low-α region and can be used to efficiently find satisfying assignments by the decimation procedure described in the inset below. However, in the hard-SAT region, the use of BP equations is limited. In $3 - SAT$, they stop to converge already at $\alpha = 3.87$ while for $K > 3$ they always converge below $\alpha_c(K)$ and yet the pdfs one obtains seem to be less efficient for obtaining SAT assignments with respect to the pdfs provided by SP. The reason for such limitation resides, most likely, in the clustering phenomenon and in the intra-cluster constraintness property of variables which are ignored by the BP procedure. SP overcomes this problem.

9.5 Survey Propagation Equations

We refer to Ref. [31] for the original statistical physics derivation of the SP equations, valid in the $n \to \infty$ limit. The chief difference with respect to the BP formalism of the previous section consists of taking care of clusters of solutions: within one cluster a variable can be either "frozen" to some value (that is, the variable always takes the same value for all SAT assignments within the cluster) or it may be "unfrozen" (that is it fluctuates from solution to solution within the cluster). The scope of the SP equations will be to properly describe the cluster-to-cluster fluctuations.

Going back to Eq. (9.14) in the previous section, we want to try also to propagate the "unfrozen" state of variables.

Then s_i will be a 3-valued variable: it can take values $0, 1$ and a new "joker" or "don't care" state $*$, corresponding to $h_i = \{0, 1\}$. In fact s_i will become nothing but a renormalized h_i (renormalization is still needed because we ignore configurations for which $h_i = \emptyset$, i.e. configurations which are incompatible with the addition of the new variable).

In order to take care of the new joker state, the Eqs. (9.14) are modified to

$$P^{(j)}(s_i = v) = \frac{P^{(j)}(h_i = \{v\})}{P^{(j)}(h_i = \{0\}) + P^{(j)}(h_i = \{1\}) + P^{(j)}(h_i = \{0,1\})} \tag{9.17}$$

$$P^{(j)}(s_i = *) = \frac{P^{(j)}(h_i = \{0,1\})}{P^{(j)}(h_i = \{0\}) + P^{(j)}(h_i = \{1\}) + P^{(j)}(h_i = \{0,1\})} \tag{9.18}$$

In these equations, every configuration in the restricted graph can be extended to: zero (canceled) or just one configuration. There is no longer any *splitting* into two configurations as happens in BP and this is most likely the cause of the convergence improvement of SP with respect in BP.

The above SP equations have first been derived [31] as $T = 0$ cavity equations confined to zero energy states. Their general version also can be used to analyze states of finite energy [29, 31].

9.6 Decimating Variables According to Their Statistical Bias

Once we have found a fixed point for Eqs. (9.16), (9.17), and (9.18) we need to get back "complete" probabilities in order to be able to solve the original SAT problem. Of course, this is simply a matter of *not* ignoring the additional variable j in Eq. (9.16) which becomes

$$\begin{aligned} P(h_i = \{0,1\}) &= \prod_{a \in A(i)} P^{(i)}(u_{a\to i} = \{0,1\}) \\ P(h_i = \{v\}) &= \prod_{a \in A(i)} P^{(i)}(u_{a\to i} = \{0,1\}) + P^{(i)}(u_{a\to i} = \{v\}) \\ &\quad - \prod_{a \in A(i)} P^{(i)}(u_{a\to i} = \{0,1\}) \end{aligned} \tag{9.19}$$

And $P(h_i = \emptyset)$ is obtained by normalization. Eqs. (9.17), (9.18) become

$$\begin{aligned} P(s_i = v) &= \frac{P(h_i = \{v\})}{P(h_i = \{0\}) + P(h_i = \{1\}) + P(h_i = \{0,1\})} \\ P(s_i = *) &= \frac{P(h_i = \{0,1\})}{P(h_i = \{0\}) + P(h_i = \{1\}) + P(h_i = \{0,1\})} \end{aligned} \tag{9.20}$$

With these quantities at our disposal, the following simple decimation procedure can been implemented:

```
algorithm SP & Decimation
begin
  ${\eta} \leftarrow$ random;   comment initialize surveys
  $B_i \leftarrow 1$;   comment initialize biases
  while $\max_i |B_i| > \epsilon$ do
    while $|\eta' - \eta| > \epsilon$ do
      Compute $\{\eta'\}$ from $\{\eta\}$ following Eqs. (9.15), (9.16) and (9.17), (9.18).;
      If $|\eta' - \eta| > \epsilon$, SET $\{\eta\} \leftarrow \{\eta'\}$ ;
    end
    Compute $P(s_i = 0)$, $P(s_i = *)$, $P(s_i = 1)$ following Eqs. (9.19)–(9.20) ;
    For $B_i = P(s_i = 1) - P(s_i = 0)$, find $i'$ such that $|B_{i'}|$ is maximum;
    If $|B_{i'}| < \epsilon$ STOP and output sub-formula.
    Fix $s_{i'} \leftarrow 1$ if $B_{i'} > 0$, $s_{i'} \leftarrow 0$ otherwise.
  end
end
```

The above algorithm is a very heuristic procedure, for which we have no proof of convergence. When it converges, SP provides a full set of marginal probabilities. However, all quantities have been defined in the large-n limit and what is the rigorous meaning of these quantities for finite n, in general, is an open question. The validity of the factorization approximation for the input probabilities in the SP iterations is difficult to assess. Even the notion of cluster is not easy to define for finite n. Roughly speaking, one can think that for large n, there might exist some effective "finite n clusters", such that the number of variables to flip in order to reach one cluster from another one is large, leading to a separation of scales in the number of variables involved between the intra-cluster moves and the inter-cluster moves. Such a situation would generally be difficult to handle for search algorithms, and this is where SP turns out to be quite effective.

In order to have a clear understanding of these questions for large but finite n, several numerical experiments have been run [8]. In the case of 3-SAT, with n up to 10^7, the results are summarized as follows.

- for low α ($\alpha < \alpha_d$), the variables turn out to be unfrozen,
- in the hard-sat region the output probabilities are non-trivial and the decimation procedure leads to sub-formulas which are under-constrained and easily solved by standard algorithms. Very close to α_c the decimation procedure may fail to find solutions in the first run.

Extensive numerical experiments on random 3-SAT instances at $\alpha = 4.2 - 4.25$ with sizes up to $N = 10^7$ have shown a remarkable efficiency of the decimation algorithm as shown in the table 9.2 from Ref. [8]. A basic complete version of the code which is intended to serve only for the study on random K-SAT instances is available at the web site [41]. The addition of more sophisticated heuristics in fixing variables can lead to further performance improvements very close to the critical point [7,36]. For generic K-SAT ($K > 3$) the behavior is similar.

In Figure 9.6 we report data (from Ref. [7]) obtained by running one of the most efficient heuristics, the so called WALKSAT algorithm [39], on a given random 3-SAT formula gener-

Table 9.2: Results obtained by solving with a single decimation run of the SP algorithm, 50 random instances of 3-SAT with $N = 10^5$ and 5 with $N = 10^6$ for each value of α. An attempt was first made to solve all samples by fixing variables in steps of $f = 4\%$ at a time, then the remaining (unsolved) samples where solved with $f = 1\%$. The maximal number of iterations was taken equal to 10^3 and the precision for convergence was taken equal to 10^{-2}. The table shows the fraction of instances which were solved and the fraction of variables which remained in the simplified instance. The computational cost is of the order of $n \log n$ (the largest instance was solved in less than 4000 seconds on a 2.4 GHz PC).

	% solved			N_{final}/N		
$\gamma \downarrow$	$f = 4$	$f = 1$	$f = 4$	$f = 4$	$f = 1$	$f = 4$
4.20	100%		100%	0.48		0.50
4.21	98%	100%	100%	0.43	0.53	0.46
4.22	82%	100%	100%	0.37	0.47	0.38
$N \rightarrow$	10^5		10^6	10^5		10^6

ated at $\alpha = 4.24$ with size $N = 10^5$. WALKSAT is an incomplete algorithm which works by combining a randomized local search procedure with deterministic greedy steps [39] and is known to perform drastically better than Simulated Annealing on hard random 3-SAT formulas. The dots represent the minimum number of violated clauses found by WALKSAT before and after SP decimation (that is on the subproblem produced after SP decimation).

While on the initial formula, WALKSAT gets stuck above the lower bound for the so called threshold states (that is those clusters of assignments which have a finite fraction of violated clauses and that are exponentially more numerous than the others clusters, having fewer violated clauses), the same algorithm run on the simplified sub-formula output by the SP decimation can quickly find the zero-violated clauses assignment.

9.7 Conclusions and Perspectives

When SP was first proposed and implemented [31] it gave results which looked promising. First, SP provided detailed information on the role of the individual degrees of freedom in a given random problem instance. Second, SP was able to reach states and configurations deep inside the glassy region. From the computer science viewpoint, SP was seen as an heuristic computational device which was able to provide new conjectures for the SAT/UNSAT phase transitions and – at the same time – allowed the designing of a concrete algorithm for solving problems which were considered out of reach.

The SP results stimulate several conceptual as well as concrete issues, the main one being to understand the range of applicability in contexts different from those of random spin-glass-like problems.

A partial list of open problems and research lines is as follows.

1. Matching rigorous mathematics: both the $n \rightarrow \infty$ results on the phase diagram and the analysis of the SP equations for finite n over loopy graphs require a rigorous proof.

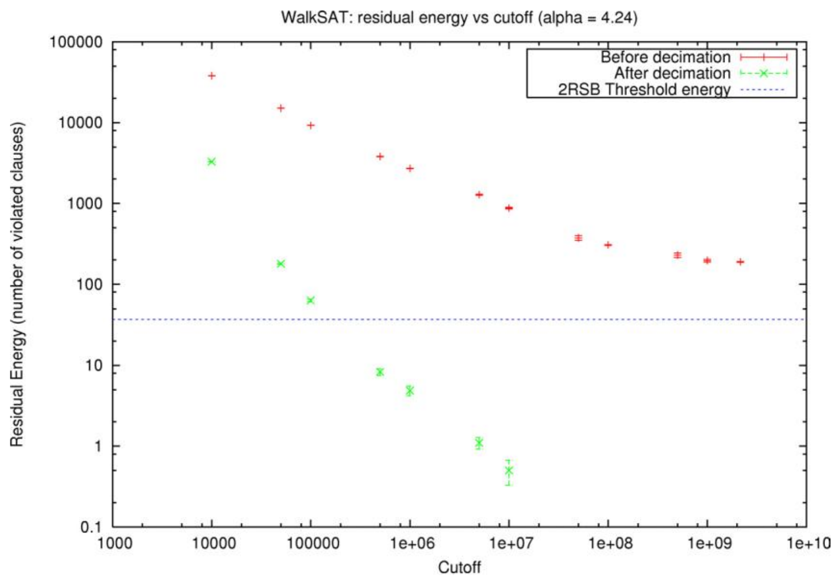


Figure 9.6: Fraction of violated clauses found by repeated runs WALKSAT versus the number of variable flips made, on a formula with $N = 10^5$ and $\alpha = 4.24$ (from Ref. [7]). The upper dots are the results of WALKSAT on the initial formula, while the lower dots are the data obtained on the simplified sub-formula. The horizontal line is the 2-RSB predicted lower bound for the number of violated clauses in the dominating threshold clusters given in Ref. [35]

2. Correlation among variables arises from the topological structure of the underlying factor graph and is obviously present in real-world problems. The SP method needs to be generalized to take care of such structures following the so called Cluster Variation Method [22], as has been done for Belief Propagation [40].

3. The decimation procedure is possibly the most elementary and arbitrary part of the algorithm. Error correction mechanisms such as backtracking should indeed be of great help in improving SP performance.

4. Having approximate information about the marginal probability distribution of the variables in a given CSP could be of great help also as a variable selection criterion in proving unsatisfiability. Coupling SP with a complete algorithm looks to be an interesting new strategy.

5. SP can be used as an optimization device in the UNSAT phase. One simply needs to resort to the general formalism [31] which allows to deal with states of positive finite energy.

Finally, it would be of great interest to apply SP to real-world CSP problems. Examples range from error-correcting codes, data compression and number theory to the more classical problems such as circuit verification and optimization. The study of hard real-world CSP is a concrete way not only to tackle unsolved problems but also to find new conceptual and methodological challenges for statistical physics.

References

[1] D. Achlioptas, *Lower bounds for random 3-SAT via differential equations*, Theoretical Computer Science **265**, 159 (2001).

[2] D. Achlioptas and Y. Peres, *The threshold for random k-SAT is* $2^k(ln2 + o(1))$, preprint (2003), http://arXiv.org/abs/cs.CC/0305009.

[3] M. Ajtai, *Generating hard instances of lattice problems*, in Proc. 28-th ACM Symposium on Theory of Computing, 99 (1996).

[4] M. Ajtai and C. Dwork, *A public-key cyptosystem with a worst-case/average-case equivalence*, in Proc. 29-th ACM Symposium on Theory of Computing, 284 (1997).

[5] R.J. Baxter, *Exactly Solved Models in Statistical Mechanics*, (Academic Press, New York, 1982).

[6] S. Ben-David, B. Chor, O. Goldreich, and M. Luby, *On the theory of average case complexity*, JCSS **44**, 193 (1992).

[7] A. Braunstein, D. Battaglia, M. Kolar, and R. Zecchina, unpublished (2003).

[8] A. Braunstein, M. Mezard, and R. Zecchina, *Survey propagation: an algorithm for satisfiability*, preprint (2003), URL: `http://lanl.arXiv.org/cs.CC/0212002`.

[9] A. Braunstein, R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, *Polynomial iterative algorithms for coloring and analyzing random graphs*, Phys. Rev. E, in press (2003); cond-mat/0304558.

[10] T. Castellani, V. Napolano, F. Ricci-Tersenghi, and R. Zecchina, *Coloring random hypergraphs*, in press, J. Phys. A, cond-mat/0306369.

[11] S. Cook, *The complexity of theorem-proving procedures*, in: Proc. 3rd Ann. ACM Symp. on Theory of Computing, p. 151, (Assoc. Comput. Mach., New York, 1971).

[12] S.A. Cook and D.G. Mitchell, *Finding hard instances of the satisfiability problem: A Survey*, In: *Satisfiability Problem: Theory and Applications*, Du, Gu and Pardalos (Eds). DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 35, (1997).

[13] S. Cocco, O. Dubois, J. Mandler, and R. Monasson, *Rigorous decimation-based construction of ground pure states for spin glass models on random lattices*, Phys. Rev. Lett. **90**, 047205 (2003).

[14] S. Cocco and R. Monasson, Phys. Rev. Lett. **86**, 1654 (2001); M. Weigt and A.K. Hartmann, Phys. Rev. Lett. **86**, 1658 (2001); A. Montanari and R. Zecchina, Phys. Rev. Lett. **88**, 178701 (2002).

[15] O. Dubois and Y. Bouhkhad, *A general upper bound for the satisfiability threshold of random r-SAT formulae*, J. Algorithms **24**, 395 (1997).

[16] O. Dubois, R. Monasson, B. Selman, and R. Zecchina (Eds.), *Phase transitions in combinatorial problems*, special issue Theoret. Comp. Sci. **265** (2001).

[17] E. Friedgut, *Sharp thresholds of graph properties, and the k-SAT problem*, J. Amer. Math. Soc. **12**, 1017 (1999).

[18] S. Franz and M. Leone, *Replica bounds for optimization problems and diluted spin systems*, J. Stat. Phys. **111**, 535 (2003).

[19] M. Garey and D.S. Johnson, *Computers and Intractability; A guide to the theory of NP-completeness* (Freeman, San Francisco, 1979); C.H. Papadimitriou, *Computational Complexity* (Addison-Wesley, 1994).

[20] F. Guerra and F.L. Toninelli, *The thermodynamic limit in mean field spin glass models*, Comm. Math. Phys. **230**, 71 (2002).

[21] Y. Gurevich, *Average case completeness*, JCSS **42**, 246 (1991).

[22] R. Kikuchi, *A theory of cooperative phenomena*, Phys. Rev. **81**, 988 (1951).

[23] S. Kirkpatrick and B.Selman, *Critical behavior in the satisfiability of random boolean expressions*, Science **264**, 1297 (1994).

[24] L.M. Kirousis, E. Kranakis, and D. Krizanc, *Approximating the unsatisfiability threshold of random formulae*, Random Structure and Algorithms **12**, 253 (1998).

[25] L.A. Levin, *Average case complete problems*, SIAM J. Comput., **14**, 285 (1986).

[26] S. Mertens, M. Mezard, and R. Zecchina, *Threshold values for random K-SAT from the cavity method*, preprint (2003), `http://lanl.arXiv.org/cs.CC/0309020`.

[27] M. Mézard, G. Parisi, and M.A. Virasoro, *Spin Glass Theory and Beyond*, World Scientific, Singapore (1987).

[28] M. Mézard, G. Parisi, and M.A. Virasoro, Europhys. Lett. **1**, 77 (1986).

[29] M. Mézard, G. Parisi and R. Zecchina, *Analytic and algorithmic solutions to random satisfiability problems*, Science **297**, 812 (2002) (Sciencexpress published on-line 27-June-2002; 10.1126/science.1073287).

[30] M. Mezard, F. Ricci-Tersenghi, and R. Zecchina, *Two solutions to diluted p-spin models and XORSAT problems*, J. Stat. Phys. **111**, 505 (2003).

[31] M. Mézard and R. Zecchina, *Random K-satisfiability: from an analytic solution to a new efficient algorithm*, Phys. Rev. E **66**, 056126 (2002).

[32] R. Monasson and R. Zecchina, *Entropy of the K-satisfiability problem*, Phys. Rev. Lett. **76**, 3881 (1996).

[33] R. Monasson and R. Zecchina, *Statistical mechanics of the random K-Sat problem*, Phys. Rev. **E 56** 1357 (1997).

[34] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman and L. Troyanksy, *Determining computational complexity from characteristic 'phase transitions'*, Nature **400**, 133 (1999).

[35] A. Montanari, G. Parisi and F. Ricci-Tersenghi, *Instability of one-step replica-symmetry-broken phase in satisfiability problems*, preprint (2003), URL: `http://arXiv.org/abs/cond-mat/0308147`.

[36] G. Parisi, *A backtracking survey propagation algorithm for K-satisfiability*, preprint (2003), URL: `http://arXiv.org/abs/cond-mat/0308510`.

[37] G. Parisi, in proceedings of SAT03 (2003); M. Mezard and R. Zecchina, unpublished (2003).

[38] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 2nd ed. (MorganKaufmann, San Francisco, 1988).

[39] B. Selman, H. Kautz, and B. Cohen, *Local search strategies for satisfiability testing*, in “Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge”, October 11-13, (1993).

[40] J.S. Yedidia, W.T. Freeman and Y. Weiss, *Understanding belief propagation and its generalizations*, talk given at International Joint Conference on Artificial Intelligence (IJ-CAI) 2001.

[41] SP code at www.ictp.trieste.it/$\sim$zecchina/SP

Part III: New Heuristics and Interdisciplinary Applications

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

10 Hysteretic Optimization

Károly Ferenc Pál

Curious children, keen to discover the world, enjoy playing with magnets. As most of them like watching television as well, some discover that magnets distort the picture in a strange way. Unfortunately, on color TV sets the magnet causes permanent discoloration. At first it may be amusing to see ones favorite TV stars with a black eye, but after a while it definitely becomes boring. Spots of the wrong color on the screen of an expensive set will compromise the viewing experience. The technician called in will use a large ring-shaped object with a long mains cord to sort out the problem. After connecting the appliance to the electricity supply, he begins a strange dance. First he holds the facility near to the screen and swings it around the center of the screen in small circles. Then he widens the circles while he starts backing away from the screen slowly, continuing to make circles. A couple of meters away from the screen the performance ends. By that time the set is repaired. Some technicians exaggerate the motion and even murmur magic words to impress the owners who will eventually pay the bill.

The cause of the discoloration is the following. The picture on the screen consists of dots of three basic colors. The phosphor on a dot lights up if it is hit by electrons coming from one of the three electron guns in the back of the tube. Each basic color corresponds to one of the guns. It is a metal mask with holes just behind the inside surface of the tube, which ensures that electrons from each gun hit only dots of appropriate color, simply by being in the path of those electrons that would hit the wrong spots. The best material for this mask happens to be a ferromagnetic alloy, which can be magnetized easily. If this happens, the trajectory of the electrons will be slightly distorted, and some of them will end up on a neighboring spot of different color. One has to demagnetize the alloy to solve the problem. This can be done with a degaussing coil. The coil fed by the alternating current (ac) from the mains will provide an oscillating magnetic field of alternating direction. Such a field, with a slowly decreasing amplitude, is the most appropriate way to demagnetize magnetic materials (ac-demagnetization). The reduction of the amplitude of the magnetic field is achieved simply by moving the coil away from the screen slowly. The role of the circular motion is mainly to cover the whole area of the screen. A less spectacular but everyday application of demagnetization with a diminishing alternating magnetic field is the erasure of magnetic tapes. The erase heads of tape recorders work by using exactly the same principle.

From our point of view it is not the demagnetization of the sample that is really important, but a side effect of the process, namely the fact that a sample ends up in a remarkably stable state after the demagnetization process. Like slow cooling, slow demagnetization drives the magnetic material into a low-energy state. Simulated annealing extended the notion of slow cooling to the approximate solution of virtually any optimization problem [11]. This method

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

has become one of the most widely used practical optimization techniques. In our recent work [16] we proposed the application of a procedure based on ac-demagnetization (ACD) to solve optimization problems. The most obvious class of problems on which to use this method is the minimization of the energy of disordered magnetic systems [6]. In this case the application of the method simply consists of simulating the behavior of the system under the effect of the appropriately varying magnetic field. The application of the procedure to Ising spin glasses will be shown in the next section. A detailed description of the algorithm will also be given. There is a variety of practical optimization problems whose objective function looks like the expression for the energy of a spin system. The application of the method to such problems is straightforward. To generalize the procedure to other types of optimization problems some concepts have to be generalized. Notably, something equivalent to the external magnetic field has to be introduced. As in the case of simulated annealing, the notion of temperature can be extended, so the generalization of the magnetic field can also be achieved. However, the choice is not unique, and the application of the present method turns out to require some more considerations than that of simulated annealing. We give the recipe of the generalization in Section 10.2. In Section 10.3 we show, using the example of the traveling salesman problem, how the general recipe can be implemented in practice. We also show and compare different possible implementations. In the last section we discuss what has to be done before we can find out whether the method may really become a useful practical tool.

10.1 Hysteretic Optimization for Ising Spin Glasses

The full Hamiltonian of a system of interacting Ising spins, including the effect of the external field is:

$$\mathcal{H} = -\frac{1}{2}\sum_{i,j}^{N} J_{ij}\sigma_i\sigma_j - H\sum_{i}^{N}\xi_i\sigma_i, \tag{10.1}$$

where $\sigma_i = \pm 1$ is the value of spin i, J_{ij} characterizes the interaction of spins i and j, H is the strength of the external field, and $\xi_i = \pm 1$ characterizes the direction of the external field at spin position i. For a homogeneous field, $\xi_i = 1$ for all i. It will be useful to consider fields with random orientations at each spin position. We note that such fields cannot be realized experimentally. The character of the spin system is determined by the interactions J_{ij}. In the case of spin glasses, J_{ij} are fixed random variables having both positive and negative signs. Details about spin glasses may be found in Chapter 3. We consider here two types of spin glass models. In the case of the Sherrington–Kirkpatrick (SK) model $J_{ij} = z_{ij}/\sqrt{N}$ for all $i \neq j$ pairs, where z_{ij} is a random Gaussian number with zero mean and unit variance. In the three-dimensional Edwards–Anderson model the spins are arranged on a cubic lattice. The interaction between spins on nearest neighbor sites is $J_{ij} = z_{ij}$, other pairs do not interact. We will call magnetization the quantity $m = 1/N\sum\xi_i\sigma_i$, which agrees with the usual definition in the case of a homogeneous external field. Because of a spin-gauge symmetry, any random choice of ξ_i is equivalent, only it should not be correlated with any special, for example locally optimal, state of the system. The transformation corresponding to this symmetry can be achieved with a series of elementary transformations T_k that changes the sign of J_{kj} for all j, and ξ_k at the same time. Then it is easy to see that there will be a one-to-one correspondence between the states of the two systems. A state of the original system under the influence of the original field and the state of the transformed system, differing from the above state in the sign of σ_k under the influence of the transformed field, will behave in exactly the same way, in particular, they will have the same energy at any H, including $H = 0$. With a series of such transformations any external field configuration ξ_i may be made homogeneous: we should apply T_k whenever $\xi_i = -1$. This way any concrete spin-glass instance given by J_{ij} may be transformed into another one that behaves in the same way under the influence of a homogeneous field as the original system behaves under the influence of the original field. With the ξ_i-dependent definition above, the magnetization of the corresponding states will be equal. We noted that ξ_i should not be correlated with special states of the system. The argument above is valid for such choices as well, but then the transformed system may not be considered a 'decent' spin-glass instance. For example, if the $\sigma_i = \xi_i$ aligned state is a low-lying state at zero field, then there will be a lot more positive J_{ij} values than negative ones in the transformed system, an extremely improbable realization of the distribution.

Demagnetization of the system is achieved by making the strength H of the magnetic field oscillate with a slowly decreasing amplitude. The dependence of the magnetization of a three-dimensional EA spin glass consisting of $N = 10^3$ spins, on the strength H of the magnetic field during a simulation of the system, is shown in Figure 10.1 (left). The figure was made with a homogeneous field, but it would look just the same with any random choice of ξ_i (it would look quite different with some special choices). The magnetization curve starts at the right-hand corner of the figure and spirals inwards, reaching approximately zero magnetization at the end. It can be seen that the magnetization, and consequently the state of the system, depends not only on the value of the magnetic field, but also on the history of the system. It matters, where the system started from previously and in which direction the strength of the field has changed. This history dependence is called hysteresis. We note that, in a strictly periodic magnetic field, the magnetization curve would be an almost closed curve. Not exactly the same sequence of microscopic states would be repeated over and over again, but at least they would have very similar macroscopic properties in every period. However, the almost closed curve would still have two markedly different branches depending on the direction of the change in the magnetic field. While the field changes, the resulting magnetization lags behind.

During the simulation we follow the evolution of the microscopic states of the system while we vary the magnetic field. We do not consider a finite temperature ($T = 0$ dynamics). The microscopic state is given by the set of spin values σ_i. From the Hamiltonian it is obvious that for a very strong field the energy will be dominated by the second term of Eq. (10.1), and in the equilibrium state each spin will be aligned with the external field, i.e., $\sigma_i = \xi_i$. This is the starting configuration. While the field is reduced from its very large value, nothing happens until one of the spins becomes unstable. As long as the flip of spin i would increase the energy of the system, it remains stable. From Eq. (10.1) the condition of stability is that $\sigma_i v_i > 0$, where the local effective field $v_i = w_i + H\xi_i$ at site i is the sum of the local internal field $w_i = \sum_j J_{ij}\sigma_j$ and the external field. In other words, spin i is stable if it is aligned with the local effective field.

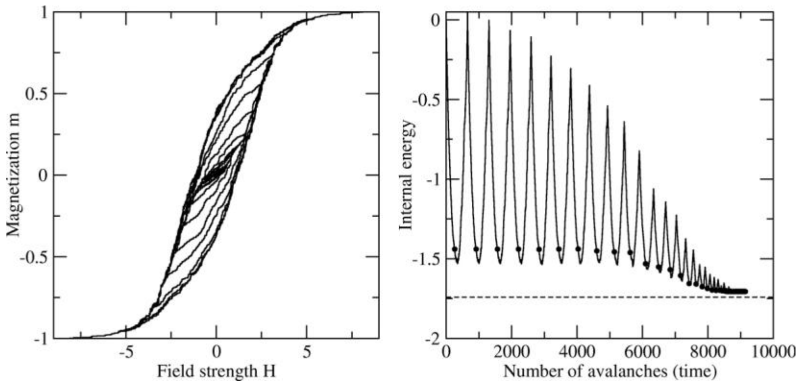


Figure 10.1: Magnetization curve (left) and the internal energy per spin during the demagnetization process (right) for an $N = 10^3$ three-dimensional Edwards–Anderson spin glass. Dots show values at zero field, dashed line marks the ground-state energy.

Therefore, the saturated configuration is stable until H is larger than $-\sigma_i w_i$ for all i. The spin first becoming unstable is the one for which this expression is maximal, and the strength of the field at that point will be just the value of the expression, $H_S = \max\{-\sigma_i w_i\}$ with $\sigma_i = \xi_i$. Therefore, we do not have to decrease the field gradually in the simulation, we can calculate directly when the first spin flip will happen, and start the procedure there. The flip of the spin will change the local effective fields for all spins it interacts with, possibly even reversing the sign of some of them. Therefore, the spin flip itself may destabilize other spins, whose flips may cause further spins to become unstable. A single spin flip may induce a whole avalanche. Eventually, the system gets into a stable state. As we assume a very slow change of the magnetic field, we implement the adiabatic limit, i.e., we keep the magnetic field fixed during the avalanche. After the system gets into a stable state at that field, we could reduce the field gradually further for a while without anything happening. Instead of doing that, we can easily calculate again directly when the next spin flip will happen, and just set H there. It can be done in the same way as for the first flip: we have to find the maximum value of $-\sigma_i w_i$ among spins aligned with the external field. Spins pointing against the field will become even more stable when the field is reduced, never starting an avalanche. For the initial state, before the very first spin flip, there are no such spins. This way we can follow the evolution of the system from avalanche to avalanche. Each avalanche will reduce the magnetization the system. The spins taking part in the avalanche drag each other into a stable configuration. If we started to increase the field back right after the avalanche, at first nothing would happen. The avalanche would not be reversed (except for ‘avalanches’ consisting of a single spin flip). The magnetization will not start increasing until another avalanche starts somewhere else, at some stronger field. This way the change of the magnetization will lag behind. This is the cause of hysteretic behavior. The simulation for an increasing field can be done analogously to the case of the decreasing field. We will show later exactly how an avalanche can be simulated on the computer. This will be the most critical part of the code, as the whole algorithm consists of following many such avalanches and little computer time is required for doing anything else.

For the demagnetization of the system we have to alternate the direction of the change of the magnetic field. This is easy to do. When the absolute value of the field at which the next avalanche would start is larger than the amplitude of the field at that stage, we do not start that avalanche. We start changing the field in the opposite direction. The full demagnetization cycle is as follows. We start with the fully aligned configuration and calculate H_S, where the first spin flip, and possibly avalanche, occurs. We go from avalanche to avalanche in the direction of decreasing field until the next avalanche would start at an H smaller than $H_1 = -\gamma H_0$, where the $\gamma < 1$ is the reduction factor of the amplitude, and H_0 is the maximum amplitude. Just before we would reach H_1, we start increasing the field back towards the positive strength values, i.e., find H in the increasing direction where the actual state becomes unstable, follow the avalanches until the next avalanche would start at a field strength larger than $H_2 = -\gamma H_1 = \gamma^2 H_0$. Then we start decreasing the field again, until the next turning point. We carry on until no spin flip occurs between H_k and H_{k+1}. For the maximum amplitude $H_0 = H_S$ is usually a reasonable choice. If the field configuration ξ_i is uncorrelated with any optimal state of the system, a field strength of $H'_S \approx H_S$ is enough to align the system starting from any state. Following the system until it becomes parallel with the external field again would obviously be a waste of computer time, so a choice of $H_0 > H'_S \approx H_S$ makes no sense. The argument fails if the field is parallel with an optimum state of the system. Then H_S is negative, because the aligned state is stable at any positive field strengths, even at zero field.

The magnetization curve of Figure 10.1 (left) was generated with the simulation outlined above, with the reduction factor $\gamma = 0.9$ and $H_0 = H_S$. For us, the result shown in Figure 10.1 (right) is more important. It shows the variation of the internal energy throughout the process. The internal energy is the energy without the contribution from the external field, the first term of Eq. (10.1). This is the objective function to be minimized in the spin glass ground state problem. For the spin models considered here with interactions J_{ij} of Gaussian distribution there are just two ground states, one of them is just like the other one with all spins reversed. Dots show the values at zero field strengths. We can see the, almost steady, decrease. This side effect of the demagnetization makes the process appropriate for optimization. We can see on the figure that the decrease of energy is marginal during the largest oscillations, which suggests that for the sake of the optimization a maximum amplitude H_0 smaller than the saturation field H_S is enough. Unfortunately, we have no general recipe how much smaller H_0 we may choose without compromising the results. We may make some experiments.

The full curves in Figure 10.2 show the distribution of the energy per spin after the ac-demagnetization process for an N=1000 spin SK (right), and for the previous $N = 10^3$ EA spin glass (left) examples, respectively. Energies are measured relative to the ground state energies, which have been determined with a combination of genetic algorithm and local optimization [13]. We applied a $\gamma = 0.95$ reduction factor for the amplitude of the field oscillations here. The slower we decrease the amplitude of the field the better the results we get. However, slower demagnetization obviously requires more computer time. Our experience is that increasing the factor γ closer to one than about $0.90 - 0.95$, makes only a marginal improvement, therefore it is not really worth the extra computing effort. It is unlikely that in the limit of infinitely slow reduction of the field ACD would guarantee a ground state even statistically, as SA does in the limit of extremely slow cooling [2]. It is possible that there is some correlation between the field configuration and the final demagnetized state, which makes some parts of the configuration space hard or even impossible to reach with a given field pattern ξ. Therefore, it is useful to repeat the process several times with different field patterns. The distributions on Figure 10.2 were extracted from such calculations. This is the reason it was useful to introduce the experimentally not realizable field whose direction changes from site to site.

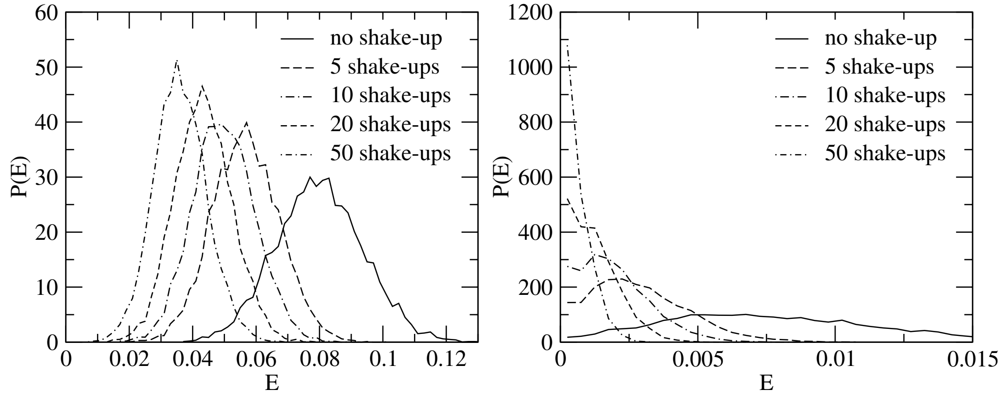


Figure 10.2: The distribution of the energy per spin after the ac-demagnetization process (no shake-ups) and after hysteretic optimization with 5, 10, 20 and 50 shake-ups for an $N = 10^3$ Edwards–Anderson spin glass (left) and for an $N = 1000$ Sherrington–Kirkpatrick spin glass (right). Energies are measured relative to the ground state energies. Different random choices of ξ_i have been used in different runs.

Instead of starting the process all over again, it is even more effective just to partially shake up the system repeatedly with fields of different configurations ξ_i. This way we may preserve the good correlations already achieved by the previous fields. A shake-up is another demagnetization process, but its maximum strength H_{shake} is far too weak to align the system completely. Consecutive shake-ups are always done on the best solution achieved by the previous shake-ups: if a shake-up makes the solution worse, we return to the previous one. As a shake-up preserves at least the best, that is the most stable, properties of the best solution already achieved, a certain number of shake-ups turns out to be significantly more effective than doing the same number of independent full ac-demagnetizations. Moreover, as shake-up is done with a smaller amplitude, it is also much faster than the process started with the full amplitude. As technically a shake-up is the same as the main demagnetization process itself, it requires very little extra programming. The only difference is that in this case the system initially is in a state stable at zero field, therefore we have to start the procedure by increasing the field from zero to H_{shake}. Then the process is the same as before: decrease the field from H_{shake} to $-\gamma H_{\text{shake}}$, increase it again to $+\gamma^2 H_{\text{shake}}$, and so on. In Ref. [16] we called hysteretic optimization (HO) the ac-demagnetization process followed by shake-ups.

In Figure 10.2 HO results are also shown with different numbers of shake-ups. For the SK case the results are fairly good. After 50 shake-ups, the exact ground state was found in 3.5% of the attempts, and states very near to the ground state were found most of the time. Shake-ups achieve a considerable improvement for the EA example as well (Figure 10.2), but the results are less convincing. The reason for the difference in performance lies most probably in the local properties of the EA model. In the low-energy states of an EA system there are remarkably stable local spin arrangements. At zero field only the relative orientation of the spins matter, therefore an arrangement and the one with all spins reversed are equivalent. Flipping a stable group of spins together gives an equally stable group. The lowest lying states differ from each other in the relative orientation of such groups. To go from one such state to another, preferably to the ground state, we should flip whole groups. However, no procedure based on single spin flips may flip a whole cluster without first destroying it. A field strong enough to make the necessary rearrangement will inevitably destroy important correlations. This actually makes such instances hard for any methods based on single-spin flip dynamics, including SA. A genetic algorithm [3, 7] may have different combinations of the overall orientations of the stable groups in the complete population of solutions which it works with simultaneously, and it may be able to recombine those groups with an appropriate crossover operation. Therefore, a carefully designed genetic algorithm has a better chance to find the ground state of such problems [4, 5, 12, 13]. Another very fruitful approach is to recognize those very stable local groups as common patterns in low-lying states and flip them together in a single step, that is to treat such groups as single spins [8, 9]. The reader may find a detailed account of this idea in Chapter 3 of the present book. We show in Ref. [16] that this idea may be combined with hysteretic optimization, and the resulting algorithm works very well for the EA case. Therefore, HO may be treated not only as a self-contained optimization technique, but also as a building block.

Another difference between the behavior of the SK and EA models – not unrelated to the above mentioned difference in their local properties – is that the size distributions of the avalanches are very dissimilar. In the SK case the distribution shows a power-law behavior, typical to critical systems [15], for finite systems the average avalanche size grows with the system size. Therefore, we often get very large avalanches during the simulation. For large enough EA systems the distribution is independent of the system size, and the average avalanche size is small; on the major hysteresis loop only about two for the three-dimensional model, and grows slowly with the number of dimensions. Whether this property has a direct relevance to the performance of the algorithm requires further investigation.

As we have already stated, the most critical part of the algorithm is the simulation of the avalanches. The input parameters are the stopping field H_x (one of H_1, H_2, ... introduced before), the number of spins N, the spin-spin interactions J_{ij}, and the field pattern ξ_i. The local effective field w_i and the spin configuration σ_i is updated by the algorithm. At input we should make sure that σ_i and ξ_i are such that the system is stable for a range of H values. This is true for the states parallel or antiparallel with the field configuration, or for a state given as an output from the same procedure. The first step is to find either the lower or the upper limit of this range of stability, depending on whether we are moving downwards ($H_x < 0$) or upwards ($H_x > 0$) with the field. That is where the next avalanche will happen. The spin that seeds the avalanche has also to be found. This is done by an algorithm next_field:

```
**algorithm** next_field($H_x, N, \xi_i, w_i, \sigma_i$);
**begin**
    **if** $H_x < 0$ **then**
    **begin**
        $H := -\infty$;
        **for** all spins $i$ parallel to external field $\xi_i$ **do**
            **if** $-\sigma_i w_i > H$ **then**
                $first\_spin\_to\_flip := i$; $H := -\sigma_i w_i$;
    **end**
    **else**
    **begin**
        $H := \infty$;
        **for** all spins $i$ not parallel to external field $\xi_i$ **do**
            **if** $\sigma_i w_i < H$ **then**
                $first\_spin\_to\_flip := i$; $H := \sigma_i w_i$;
    **end**
    **return** $H, first\_spin\_to\_flip$
**end**
```

Then algorithm avalanche can be organized in the following way:

```
**algorithm** avalanche($H_x, N, J_{ij}, \xi_i, w_i, \sigma_i$);
**begin**
    **next_field**($H_x, N, \xi_i, w_i, \sigma_i$)
    **if** $H$ is beyond limit **then return** $H$
    $number\_of\_spins\_to\_flip := 1$;
    add $first\_spin\_to\_flip$ to the list of spins to flip;
    **while** $number\_of\_spins\_to\_flip > 0$ **do**
    **begin**
        $i$ := randomly chosen spin from list of spins to flip;
        $\sigma_i := -\sigma_i$;
        **for** all spins $j$ interacting with spin $i$ **do**
        **begin**
            $w_j := w_j + 2\sigma_i J_{ji}$;
            **if** spin $j$ is unstable, i.e., $\sigma_j(w_j + H\xi_i) < 0$ **then**
                (**if** spin $j$ is not on the list of spins to flip **then**
                    include it; $number\_of\_spins\_to\_flip := number\_of\_spins\_to\_flip + 1$)
            **else**
                (**if** spin $j$ is on list of spins to flip **then**
                    delete it; $number\_of\_spins\_to\_flip := number\_of\_spins\_to\_flip - 1$)
        **end**
    **end**
    **return** $H$
**end**
```

The algorithm above will return the value of H where the avalanche occurred. If the avalanche would occur at a field strength H beyond the limit (i.e. below $H_x < 0$ or above $H_x > 0$), it will return that value and leaves the configuration unchanged. $H = \pm\infty$ is returned if the configuration cannot change further (i.e. the state becomes aligned before reaching H_x). We assumed that an avalanche starts always by the destabilization of a single spin. There are cases, where several spins may become unstable at exactly the same external field. This happens, for example, in the case of the so called $\pm J$ spin glass, where the interactions between spins differ only in sign, not in strength, therefore the influence of the neighborhood on many spins will be equally strong. For such systems the procedure has to be modified accordingly. Another important point is that when there are more than one unstable spins, we choose randomly which spin to flip next. One could think of flipping all such spins at once. However, it would not work, because it would often lead to an infinite loop. Think of just two antiparallel spins, which are unstable because their interaction would prefer them to be parallel. If we flipped both of them together, they would both be unstable again. Flipping them again we would then be back where we started from. Amongst possible serial update rules the random choice seems to be the most flexible and also the most fair one. Besides the random choice of ξ_i, this introduces another random element into the algorithm. Therefore, we may arrive at different states at the end of the procedure even if we use the same ξ_i all the time, for example uniform field. However, many low-lying states are never reached in this way. In Ref. [16] we introduced a third source of randomness: we only applied the rule for the amplitude decrease on average, and varied randomly the actual stopping points H_x. Although this helped a lot, it turned out that allowing randomness in the direction of the external field is a much better way of making the algorithm flexible, and with that, the variation of the stopping point is unnecessary.

After each spin flip we have to update the local internal fields at all sites that interact with the spin flipped. We also have to update the current list of unstable spins. As stability at a fixed H may only change due to a change in the local internal field, we do the two updates in the same loop. When updating the list of unstable spins, we must not forget that a spin flip not only may make a stable neighbor unstable, but it can also stabilize an unstable neighbor. We may have to include both some new spins in the list, and delete some. In the SK case, update has to be done at every site. Then it is not only simpler, but also faster not to modify the previous list as we do above, but to throw it away completely and make a new one from scratch.

It is interesting to note that a spin may flip more than once within one period (i.e., between H_k and H_{k+1}). Sometimes certain spins flip more than once even within the same avalanche. This may happen, because the actual stability of a spin depends not only on the external field, but also on the local internal field, which changes whenever a spin interacting with that spin flips.

The algorithm for the simulation of ac-demagnetization may be written in the following way:

```
**algorithm** ac-demagnetization($n, J_{ij}, H_0, \gamma$)
**begin**
    **for** $i := 1, 2, \ldots, n$ **do**
        $\xi_i := \pm 1$, sign chosen randomly; $\sigma_i := \xi_i$;
    **for** $i := 1, 2, \ldots, n$ **do**
        $w_i := 0$; **for** $j := 1, 2, \ldots, n$ **do** $w_i = w_i + J_{ij}\sigma_j$;
    $H_x := -H_0\gamma$; $avalanche\_calls := 0$;
    **while** $avalanche\_calls \neq 1$ **do**
    **begin**
        $H := 0$; $avalanche\_calls := 0$;
        **while** $H$ is within limit **do**
        **begin**
            **avalanche**($H_x, N, J_{ij}, \xi_i, w_i, \sigma_i$)
            $avalanche\_calls := avalanche\_calls + 1$;
        **end**
        $H_x := -H_x\gamma$;
    **end**
**end**
```

Here we assumed that H_0 had been chosen in advance, from some preliminary calculation. We may choose $H_0 = H_S$ instead, which becomes available in the above routine as the H value provided by the procedure avalanche when called the very first time. The internal loop terminates when the field would exceed the limit. The range of H is reduced by a factor of γ every time. The whole process terminates, when there is only a single avalanche call in the inside loop, which means that the state achieved remains stable within the last H range. We do not give the full algorithm for the hysteretic optimization including shake-ups. Each shake-up is done in the same way as the ac-demagnetization above with the smaller H_{shake} playing the role of H_0, only we should start not with the system aligned with the new external field, but with the state we get from the best state found so far by increasing the field in the usual way up to H_{shake}.

10.2 Generalization to Other Optimization Problems

Simulated annealing is one of the most widely used optimization methods. It was motivated by the observation that disordered systems may be brought into a low-energy state, sometimes into their ground state through annealing, that is cooling them down slowly. Simulated annealing applies this principle to virtually any optimization problem. It identifies the elements P of the configuration space by physical states, and the value of the objective function $W(P)$ to be minimized with the energy of the corresponding state. In case of a maximization problem the objective function is simply multiplied by minus one. Then a dynamics is defined, which involves the definition of an elementary step in the parameter space. This is the only problem-specific choice that has to be made to apply simulated annealing. The application of the algorithm is very simple, one has to change the state step by step, accepting or rejecting each attempted move according to a probability given by the Metropolis rule [11], which depends on a parameter corresponding to the temperature. The rule ensures that in the long term each state occurs with the same probability as if the system were immersed into a heat bath. Cooling is simply lowering the temperature parameter according to some schedule.

The generalization of ACD and HO is very similar. Here the objective function is identified with the internal energy of the physical system. We also have to define the dynamics by choosing an elementary step in the parameter space. Just as in the case of SA, a good dynamics is such that the expected change of the objective function due to a single elementary step should be small. Simply speaking, the reason for this requirement is that the larger the expected change, the rougher the landscape we are trying to navigate. In a rugged landscape there are many bad local optima, and good states are surrounded by very much inferior ones. Finding a good optimum on such a landscape is just like finding a needle in a haystack. Generally, the wider the basin around an optimum, the easier it is to find it. As the elementary step defines what we actually mean by the neighborhood of a state, the character of the landscape we encounter critically depends on this choice. We will demonstrate later the importance of good dynamics on the example of the traveling salesman problem. In the previous section, for the Ising spin systems, we applied the most obvious single spin flip dynamics. Single spin flip will change N of the $N^2/2$ terms and 6 of the $3N$ terms in the expression of the internal energy for the SK and the three-dimensional EA model, respectively, satisfying the requirement above. To ensure that we are able to reach any state we require, it is also important that the successive applications of elementary steps should connect the whole parameter space (ergodicity), which is obviously true for the single spin flip dynamics.

For the generalization of ACD and HO, we must add some extra term to the cost function that will correspond to the effect of the external field. The notion of external field may be generalized by realizing that its effect is to align spin σ_i with the direction of the external field ξ_i at each spin position. For very large field strengths H the optimum state will be the fully aligned state $\sigma_i^+ = \xi_i$, that we call the reference state. Analogously, we require the term in the Hamiltonian $\mathcal{H}(P)$ of the general problem corresponding to the external field, such that the optimum state for a very strong field should be the reference state. A term proportional to some distance $d(P, R)$ between the reference state and the actual state will achieve just that: the energy contribution of such a term will be positive for every state but the reference state itself. In the case of the spin systems, the external field with large negative field strengths will force the system into the state $\sigma_i^- = -\xi_i$, which we may call another reference state, which is the same as the first one with all spins reversed, that is the 'opposite' state. Unfortunately, the notion of 'opposite' state is not readily applicable to a general optimization problem. For large negative H values the system will usually not be pulled towards a well defined state, as there may be a huge collection of configurations equally far away from the reference state, so equally advantageous for $H \ll 0$. For most configurations of the system we probably find some of them nearby, much closer than the reference configuration, therefore not too much is expected to happen when $H < 0$. To solve this problem, in Ref. [16] we suggested simply choosing two independent random states R_+ and R_- as reference states for positive and negative values of H, respectively. The formula for the Hamiltonian given in Ref. [16] is:

$$\mathcal{H}(P) = W(P) + \sum_{\alpha=\pm} \alpha H \Theta(\alpha H) d(P, R_\alpha), \tag{10.2}$$

where $\Theta(x)$ is the step function.

In the case of HO, a new pair of reference states is chosen for each shake-up. We can see that the application of the present method to a general problem requires some more considerations than the application of simulated annealing, because besides the elementary step, we must also define a distance between the states. We have some freedom of choice for both ingredients, but we should not forget that they should be consistent with each other in the sense that a single step should not change the distance too much, that is the distance of neighboring configurations (one step away) should not be large. In the next section, examples will be shown for the case of the traveling salesman problem.

For the spin problems considered so far, each configuration is characterized by the spin values, i.e., $P = \{\sigma_i\}$. The distance between two configurations is $d(\{\sigma_i\}, \{\bar{\sigma}_i\}) = \sum_i |\sigma_i - \bar{\sigma}_i|$, that is twice the number of those spin positions where the two configurations differ. We note that if we represent the spin values by $1/2(\sigma_i + 1)$, each state will be characterized by a bit string, and the above distance will just be twice the well known Hamming distance of the corresponding strings. As $|\sigma_i| = |\bar{\sigma}_i| = 1$, then $|\sigma_i - \bar{\sigma}_i| = 1 - \sigma_i\bar{\sigma}_i$. Using this identity and substituting the actual form of W and the reference states $R_\pm = \pm\xi_i$ into Eq. (10.2), the formula we easily arrive at is the same as the energy of the spin system Eq. (10.1), except for an extra term $N|H|$, which is constant at each H, does not affect the relative energies of the states, and therefore can be dropped.

We managed to recover the formula for the spin systems from the more general Eq. (10.2), but only if we used spin reversed states as the two reference states. Not being able to find an analogy with such pairs of states in the general case, we suggested in Ref. [16] choosing two independent random states as reference states. Unfortunately, if we do this for the spin systems, we get significantly worse results than with the original recipe. In our Edwards–Anderson example we got at average about 75% farther away form the optimum state, both without and with shake-ups. In the Sherrington–Kirkpatrick case the average energy above the ground state was over a factor of two larger for ACD, and shake-ups made the difference even bigger. Moreover, the probability of actually finding the ground state decreased by almost two orders of magnitude. During the demagnetization process we are trying to drag the system back and forth between the reference states. The spin reversed states are very special, because, in a sense, every state of the system lies between them: it lies on a path (actually, several paths) between the two configurations that moves farther away from one of them and closer to the other one with each step. Such a path between two randomly chosen configurations exists only for some states, so not every state is between two arbitrary reference configurations. Those states may be harder to reach by demagnetization. Qualitatively, this may be the reason for the much better performance of the algorithm with the spin reversed reference states. Analogy from geometry may help to visualize the statements above. Each state of the N-spin system corresponds to a corner of an N-dimensional hypercube, and the spin reversed states are represented by two opposite corners. A step is moving into a neighboring corner through an edge. The distance of two corners is the shortest path between them along edges (not the same as the usual geometrical distance). For a square or an ordinary cube anyone can see that not all corners are between two other, arbitrarily chosen corners in the above sense.

To improve the situation, we suggest another approach [14], which actually works better in every case we tried: to use a completely new reference state in each half-period of the demagnetization process instead of just alternating between two. This way the system is pulled towards a different direction in every half-period. The Hamiltonian valid in the kth half-period is:

$$\mathcal{H}_k(P) = W(P) + Hd(P, R_k), \tag{10.3}$$

with H taken always to be positive for the sake of simplicity. There is no reason to distinguish between odd and even half-periods. Neither would it make sense to insist on giving a formula valid throughout the whole process, it would look even more awkward than Eq. (10.2) with the alternating reference states. Noting that in Eq. (10.2) it is the sign of H that distinguishes between odd and even half-periods, it is easy to see that with $R_{2k\pm1} = R_\pm$ the two formulae are equivalent, with αH – which is always positive in the nonvanishing term – is replaced by H. A new reference configuration for every half-period gives better results than alternating two random reference states and costs little extra computation time: a quick and dirty random number generator is appropriate. In the Edwards–Anderson case the quality of the results is similar to what we get with the original recipe with the spin reversed states. In the Sherrington–Kirkpatrick case the situation is not as good, but still much better than with the two random states. Therefore, whenever the optimization problem is such that it makes sense to talk about opposite states, we would recommend to try the algorithm first with such pairs of reference configurations.

Now we summarize how to apply the algorithm for a general optimization problem. First we have to choose a dynamics by defining an elementary step in the configuration space of the problem. Then we also have to define a distance between the elements of the configuration space. A good dynamics connects configurations with similar values of the objective function, and a good distance corresponding to the dynamics is such that distances do not change much during a single step.

To start ac-demagnetization we choose a reference state first. The initial state of the system will be just the reference state, which is the lowest state for $H \gg 0$ with the corresponding field configuration. At H_S, which can be directly calculated, one of the possible elementary steps becomes favorable. After the step is done more steps may become energetically favorable, that is unstable. Therefore, just as in the case of the spin systems, a step may seed an avalanche. Avalanches cause hysteretic behavior, so the term hysteretic optimization is justified in the general case as well. We decrease the field avalanche by avalanche until we get a state stable at zero field. Then we choose a new reference state, which corresponds to a new external field configuration, and start increasing the field again, up to $H_1 = \gamma H_0$. For the spin systems discussed previously it makes no sense to choose a value larger than H_S for the maximum field amplitude H_0, because if we did so, at $H'_S \approx H_S$ we would arrive at the well defined state aligned with the current reference state. However, in the general case this is not necessarily true. The function defined as the distance from the reference configuration may have local optima according to the chosen elementary step, and if so, we may get stuck in one of them as we increase the field, never arriving at the reference state itself. Moreover, the H'_S, when we arrive at the configuration that is stable at infinitely strong fields, may be considerably larger than H_S. In this case it makes sense to choose $H_0 > H_S$, and our experience shows, that we get the best performance of the ac-demagnetization algorithm if we do so. An $H_0 = H'_S$ choice is a safe one. We should not choose $H_0 > H'_S$, which would be a waste of computer time. The exact value of H'_S depends both on the state before we started to increase the field and the reference state. However, this dependence for a given system is not too strong, and the exact value of the maximum amplitude is not very critical. We may take the average H'_S from a few trials as the maximum amplitude. As before, we may save computer time by choosing a somewhat smaller value without getting worse results. The best choice can be found by some experimentation. We note, if we apply shake-ups, it becomes less important how optimally we choose the parameters of the initial demagnetization process.

In the case of discrete optimization problems the choice of the distance between two states most consistent with the elementary step would be just the minimum number of elementary steps to go from one state to the other. This distance function can never have local optima according to the elementary step. The distance we implicitly assumed for the Ising spin systems is like this. Unfortunately, in the general case this choice is often impractical, because it may be quite involved and time consuming to evaluate. In some configuration spaces its evaluation may even be exceedingly difficult. Try to find the shortest path between two configurations of a Rubik cube!

Going back to the description of the algorithm, after we increased H up to $H_1 = \gamma H_0$ with the new reference state, we decrease it to zero. Then we choose a new reference state again, and first increase H up to $H_2 = \gamma^2 H_0$, then decrease it to zero. Repeat this with amplitudes $\gamma^3 H_0$, $\gamma^4 H_0$, and so on, until the system stays stable up to the maximum field. This concludes ac-demagnetization. Doing shake-ups is straightforward. We first increase the field up to $H_{\text{shake}} \ll H_0$ again with a new reference configuration, then starting with this smaller amplitude do exactly the same as in ACD. At each stage we compare the states before and after the shake-up and keep the better one. The best H_{shake} may be found with experimenting.

Now we present the ac-demagnetization algorithm for the general optimization problem. We again give algorithm next_field first, then algorithm avalanche, and we present algorithm ac-demagnetization itself, last. As H changes only between zero and a positive value, the value of the input parameter H_x will be zero when we decrease the field, and the H_i actual amplitude when we increase it. The "system parameters" occurring in the parameter list include all parameters necessary to characterize the problem instance, the current state and the reference state. We will use Greek indices to distinguish between the elements of the set of all elementary steps allowed by the dynamics. We will denote by ΔW_μ and Δd_μ the change of internal energy and the change of distance from the actual reference state if step μ is taken from the current state of the system, respectively. Algorithm next_field is the following:

```
algorithm next_field($H_x$, system parameters);
begin
  if $field\_down$ then
  begin
    $H := -\infty$;
    for all possible elementary steps $\mu$ do
    begin
      calculate $\Delta d_\mu$;
      if $\Delta d_\mu > 0$ then
      begin
        calculate $\Delta W_\mu$;
```

```
                **if** $-\Delta W_\mu/\Delta d_\mu > H$ **then**
                    $first\_step\_to\_take := \mu$; $H := -\Delta W_\mu/\Delta d_\mu$;
            **end**
        **end**
    **end**
    **else**
    **begin**
        $H := \infty$;
        **for** all possible elementary steps $\mu$ **do**
        **begin**
            calculate $\Delta d_\mu$;
            **if** $\Delta d_\mu < 0$ **then**
            **begin**
                calculate $\Delta W_\mu$;
                **if** $-\Delta W_\mu/\Delta d_\mu < H$ **then**
                    $first\_step\_to\_take := \mu$; $H := -\Delta W_\mu/\Delta d_\mu$;
            **end**
        **end**
    **return** $H$, $first\_step\_to\_take$
**end**
```

Then follows algorithm avalanche:

```
**algorithm** avalanche($H_x$, system parameters);
**begin**
    **next_field**($H_x$, system parameters)
    **if** $H$ is beyond limit **then return** $H$
    $number\_of\_favourable\_steps := 1$;
    add $first\_step\_to\_take$ to the list of favorable steps;
    **while** $number\_of\_favourable\_steps > 0$ **do**
    **begin**
        $\mu$ := arbitrarily chosen step from list of favorable steps;
        take step $\mu$;
        empty list of favorable steps; $number\_of\_favourable\_steps := 0$;
        **for** all possible elementary steps $\nu$ except for $\mu$ **do**
        **begin**
            calculate $\Delta W_\nu$; calculate $\Delta d_\nu$;
            **if** $\Delta W_\nu + H\Delta d_\nu < 0$ **then**
            **begin**
                add $\nu$ to the list of favorable steps;
                $number\_of\_favourable\_steps := number\_of\_favourable\_steps + 1$;
            **end**
        **end**
    **end**
    **return** $H$
**end**
```

The algorithm above is quite general, but often inefficient. We need not prepare the full list of favorable steps, it is enough to find one of them. However, in this case we should go through the possibilities in a random order so that any of them has an equal chance to be found first. This costs time, so it will only be worth it if the problem is such that the average avalanche size is large. In this case we probably find a favorable step by checking only a small fraction of all possibilities. In some cases the change of ΔW_ν and Δd_ν due to step μ is much faster to calculate than ΔW_ν and Δd_ν themselves. Remember the spin systems, where the local internal field consists of several terms, but only one of them changes when a neighboring spin is flipped. If such is the case, it is better to calculate and store all ΔW_ν and Δd_ν in advance, and just correct them after each step. Similarly, it may also happen, that ΔW_ν and Δd_ν changes only for a fraction of possible steps ν when step μ is taken, similarly to the EA spin glasses, where local fields only changed for neighbors of the spin flipped. Then it is better updating the existing list of favorable steps instead of reconstructing it, analogously to the procedure given in the previous section. As we have stated before, H is always positive, so when we decrease the field, the limit is zero. For each stage, first we decrease the field to zero, and then we increase it to a reduced maximum value with the next reference state. We stop when the state is stable at that maximum value. Therefore, the algorithm is the following:

```
**algorithm** ac-demagnetization($H_0, \gamma$, parameters characterizing the instance)
**begin**
    choose a random reference state;
    current state := reference state;
    $H_x := H_0\gamma$; $avalanche\_calls := 0$;
    **while** $avalanche\_calls \neq 1$ **do**
    **begin**
        $H := 0$;
        **while** $H > 0$ **do**
            **avalanche**(0, system parameters);
        choose a new random reference state;
        $avalanche\_calls := 0$
        **while** $H < H_x$ **do**
        **begin**
            **avalanche**($H_x$, system parameters);
            $avalanche\_calls := avalanche\_calls + 1$;
        **end**
        $H_x := H_x\gamma$;
    **end**
**end**
```

We supposed again that H_0 has been chosen in advance. We might choose $H_0 = H_S$, which we may get again as the output H of algorithm avalanche when called first. However, we mentioned that a safer choice is $H_0 = H'_S$. This we may also get as the output of algorithm avalanche. If H_x is very large, the second internal loop will not terminate until algorithm avalanche reaches a state stable at very strong fields and returns $H = \infty$. Right before that call it returns just H'_S, the field when this stable configuration was created.

10.3 Application to the Traveling Salesman Problem

We have two aims when we discuss the example of the traveling salesman problem. One is to demonstrate that the algorithm does work for a problem different to the spin problems [14] discussed so far. The other is to give examples for possible choices of dynamics and distances and for their connection to make the more general considerations of the previous section clearer.

The traveling salesman problem is one of the classical optimization problems that is easy to state but difficult to solve. It involves N cities with given distances between them. The aim is to find the shortest round tour that goes through each of the cities once. Any conceivable tour can be characterized by giving the order of the cities that are visited, therefore the configuration space of the problem is given by all possible permutations P of the cities: $P(i)$ denotes the ith city to visit. The objective function is the length of the total path $W(P) = \sum_{i=1}^{N} \Delta(P(i), P(i+1))$, where $\Delta(i,j)$ denotes the distance of cities i and j, and to simplify notation we introduced $P(N+1) \equiv P(1)$, expressing the fact that the tour ends where it started.

In Ref. [16] we chose the interchange of two cities along the path as the elementary step, (swap dynamics) and we defined the distance of configurations P and P' (not to be confused with the distance of the cities) as

$$d(P,P') = \sum_{i=1}^{N} |\Delta(P(i), P(i+1)) - \Delta(P'(i), P'(i+1))|. \tag{10.4}$$

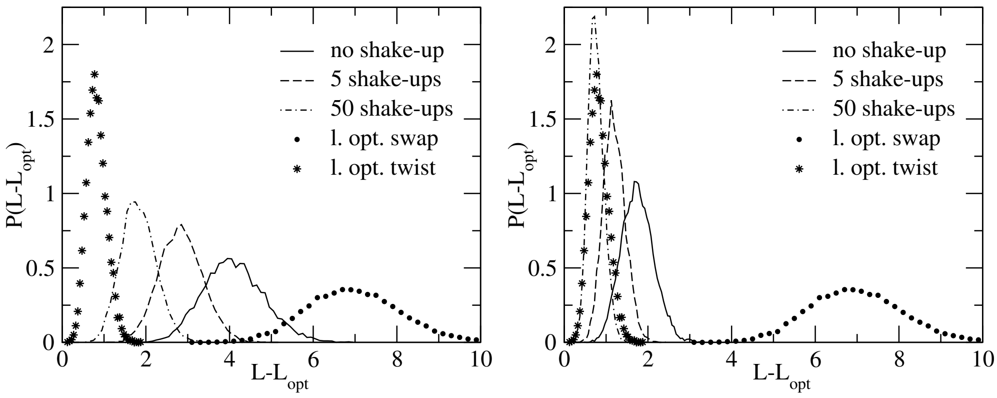


Figure 10.3: The distribution of the difference between the length of the tour and the optimum length after ac-demagnetization (no shake-ups) and after hysteretic optimization with 5 and 50 shake-ups with swap dynamics for an $N = 100$ city traveling salesman problem. Results from local optimization both with swap (interchanging two cities along the path) and twist dynamics (reversing a subtour) are also shown. Left: two alternating reference states, distance as Eq. (10.4). Right: new reference state for each half-period, distance is the number of non-common undirected links.

We used our original prescription with two reference states. Figure 10.3 (left) shows the distribution of the difference between the length of the tour from ACD and HO with a different

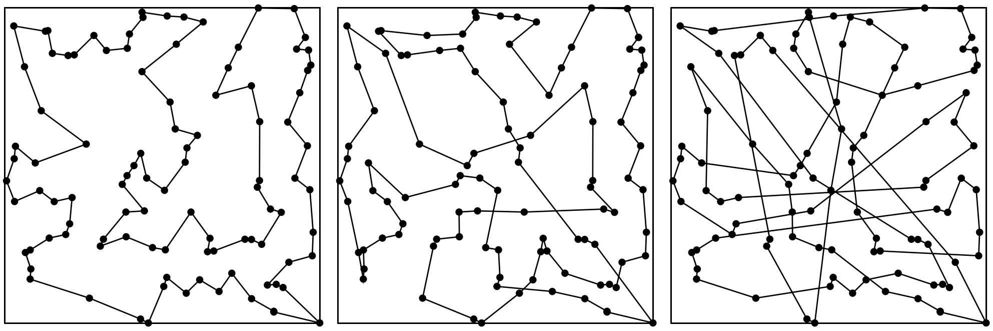


Figure 10.4: Optimum tour (left), an average tour from hysteretic optimization with 50 shake-ups (middle) as in the Figure 10.3 (swap dynamics, two reference states, distance as Eq. (10.4)) and an average tour from a local optimization with swap dynamics (right) for an $N = 100$ city traveling salesman problem. The cities are randomly distributed on a unit square.

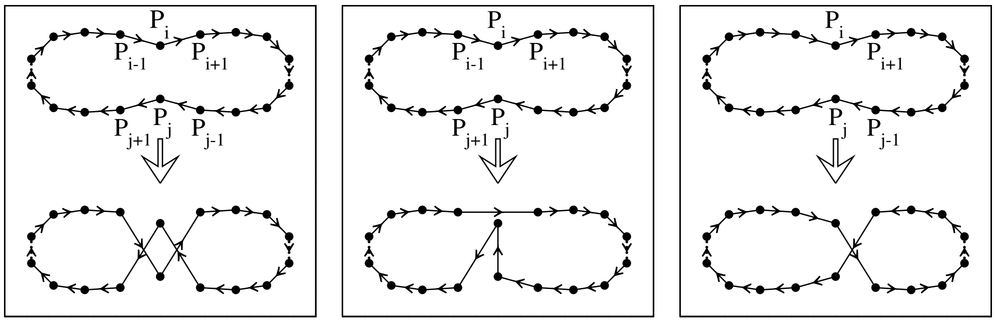


Figure 10.5: Elementary steps swap (left), move (middle) and twist (right) for the traveling salesman problem.

number of shake-ups and the optimum length for an $N = 100$ city traveling salesman problem, which was created by randomly distributing the cities on the unit square and taking their Euclidean distances. The optimum tour for that particular instance is shown in Figure 10.4 (left). In Figure 10.4 (middle) we also show an average solution given by HO with 50 shake-ups. The results are not really convincing. However, they are still not only much better than those of a local optimization with the same dynamics, but it turns out that it is more advantageous to do HO than to do a series of independent local optimizations for the same computation time. Figure 10.3 shows the length distribution, while Figure 10.4 (right) shows a typical result of local optimization with this swap dynamics. No matter how bad it looks, it is still a local optimum, that is we cannot improve it by swapping any two cities along the path. We have to conclude that HO still works. On the other hand, with a different dynamics (twist dynamics, based on reversing a subtour, see Figure 10.5) local optimization outperforms the much slower HO with the dynamics and distance we discussed so far. The distribution of tour lengths with such a local optimization is also shown in Figure 10.3. Using a new reference state in each half-period of the process instead of the two alternating ones, we could get about 20–25% closer to the global optimum, but it is still not really good.

In Figure 10.5 we show schematically three possible types of elementary steps. The first one (left) is the interchange of two cities, city $P(i)$ and $P(j)$, i.e. the ith and jth city along the tour. This is the operation we have considered so far. The permutation P' representing the new tour is very easy to express with P, namely $P'(i) = P(j)$; $P'(j) = P(i)$ and $P'(k) = P(k)$ for $k \neq i, j$. From the figure it is clear that this operation will replace four terms of the expression for the tour length by four other by cutting the $P(i-1) \to P(i)$, $P(i) \to P(i+1)$, $P(j-1) \to P(j)$, $P(j) \to P(j+1)$ links and establishing new links $P(i-1) \to P(j)$, $P(j) \to P(i+1)$, $P(j-1) \to P(i)$, $P(i) \to P(j+1)$. The next operation shown in Figure 10.5 (middle) is visiting one of the cities some other time, that is moving it somewhere else along the path. Here the ith city along the path is moved right after the jth one. The permutations representing the tours before and after the operation will look more different than in the case of the previous operation, as all cities after the ith one up to the jth one will come one place forward along the path if $i < j$. If $i > j$, i.e. the ith city gets scheduled earlier, a whole subtour will also be shifted, but this time backwards. Despite this, it is clear from the figure that here only three links are modified, so this operation is more basic than the previous one. The third operation shown in Figure 10.5 (right) is even more basic in this sense. It changes only two links. It involves traveling a part of the original tour in the opposite order. This sort of modification is called a twist or a 2-opt operation. We note that this operation disturbs few links and is therefore basic only for symmetric traveling salesman problems, that is when $\Delta(i, j) = \Delta(j, i)$. Asymmetric distances might look absurd at first sight, but there are real tasks leading to this kind of optimization problem. We might even think of the traveling salesman with one-way roads of different lengths between the destinations, or a cost-conscious one who tries to minimize fuel consumption that depends on which direction one travels on mountain roads.

As far as the expected change of the objective function is concerned, all three operations look reasonable, because they only change a few terms out of the N terms in the sum giving the total length. But the move operation (three terms) is expected to be better than the swap operation (four terms), and the twist operation (two terms) is expected to be the best. The local optimization that outperformed the previous HO results (Figure 10.3) were done with this elementary step. However, the distance between the configurations we defined in Ref. [16] and Eq. (10.4) is appropriate only for the swap operation. In the equation, links at the same absolute positions along the tours are compared, therefore, if the same link is shifted to another position, it will be compared to something else. Therefore, both the move operation that shifts a subtour by one position and the twist operation that reverses the order of cities in a subtour are expected to change the distance of the configurations a lot in most cases. Actually, two configurations differing only in where the tour starts are expected to be far away according to this definition. If we want to use a better elementary step, we must drop this notion of distance.

A very straightforward way to characterize the distance of two tours is to count how many cities would be directly followed by the same city according to both tours, which is simply the number of common directed links in the tours, and subtract this number from N. Directed link means that going from city i to j corresponds to a different link than going from j to i. With this definition the distance of tours differing only in their starting points is zero. This is actually reasonable, because the starting point is irrelevant. The proper configuration space is represented by not all permutations but by the cyclic permutations. The distance definition is fully compatible with this fact. It is easy to see that this definition is appropriate for both the swap and the move operation, but not for the twist operation. By counting the number of common undirected links between two tours and subtracting that from N, we get a third definition, which is good for each of the three elementary steps we have mentioned here. A common undirected link means that, if city i is followed by city j along one tour, than either city i is followed by city j or city j is followed by city i along the other tour. A common property of the latter two distance definitions with the distance definition for spin systems is that they depend only on the elements of the mathematical structure characterizing the configuration space, and – unlike the first definition – do not depend on the actual realization (instance) of the problem, that is on the interactions for the spin systems or on the distances between the cities for the traveling salesman problem.

If we throw away the original distance definition we get on average over 40% nearer to the optimum tour length, even if we keep the swap dynamics. The improvement due to the change in the distance definition and the prescription to choose a new reference state for each half-period is nearly 60%. This way, after 50 shake-ups, we get somewhat better tours than with local optimization with the twist dynamics (see Figure 10.3, right), which is still not something to be very proud of, taking into account the huge difference between the computation times required. The two latter distance definitions actually give results very similar to each other. We made 10 000 independent HO runs with 50 shake-ups with each prescription, but we still did not get the optimum solution with the swap dynamics.

As we expected, the main advantage of changing the distance definition is that it allows us to use better dynamics. Figure 10.6 shows results with both the move (left) and the twist (right) dynamics with 0, 5 and 50 shake-ups. All results are much better than the ones with the swap dynamics. Even an ac-demagnetization without any shake-ups turns out to be much better than the best we could achieve with the swap dynamics after 50 shake-ups. Twist dynamics – as expected – gives better average tour lengths, but shake-ups seem to be somewhat more effective with the move dynamics. What we find surprising is that, for our particular problem, move dynamics found the optimum configuration more often than twist dynamics: after 50 shake-ups we arrived at the optimum tour in 23% and 14% of the runs with the move and the twist dynamics, respectively.

10.4 Outlook

Hysteretic optimization is an optimization method inspired by a physical process: demagnetization of magnetic materials by an external field oscillating with a diminishing amplitude. As a general purpose method it is very new. Although there is still much work to be done, the first results are promising. Even in its present form the method performs as well as one may reasonably expect from a general purpose heuristic optimization algorithm. In most cases it cannot compete with the methods specifically developed for particular problems, but there are certainly cases – like the Sherrington–Kirkpatrick spin glass – when it works remarkably well.

It still has to be tried on many other types of problem to discover its strengths and weaknesses. It should also be found out how important the requirement of a slow variation of the external field is, and whether we could do something faster than simulating behavior avalanche by avalanche, without compromising too much of the quality of the results. In the case of sim-

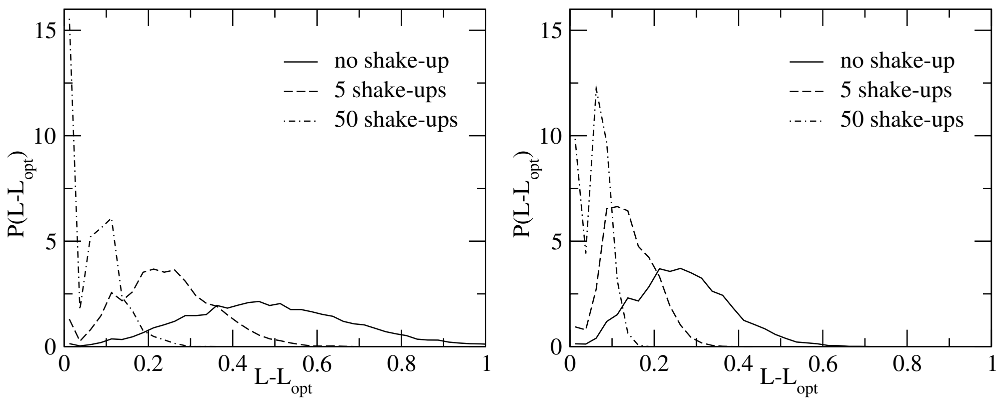


Figure 10.6: The distribution of the difference between the length of the tour and the optimum length after ac-demagnetization (no shake-ups) and after hysteretic optimization with 5 and 50 shake-ups with move (left) and with twist (right) dynamics for an $N = 100$ city traveling salesman problem. A new reference state was used for each half-period; distance is number of non-common undirected links.

ulated annealing it is known that even cooling is not the best strategy, one can work out more effective cooling schedules by measuring a quantity analogous to the specific heat of the system during the process [1, 10]. Similarly, we are sure that the simple strategy of decreasing the amplitude of the field according to a geometric sequence is not the optimal one. More theoretical work is required to find out exactly which properties of the system could guide the variation of the field amplitude. It would be important to understand better how and why the algorithm works. A better understanding could give ideas for further practical improvements, and give hints for what type of optimization problem we could expect the method to be competitive. It would also be good to have some better idea of the optimum choice of parameters like the maximum amplitude H_0 in the ac-demagnetization and H_{shake} for the shake-ups.

Another direction that is worth exploring is to consider the method as a building block, and combine it with other techniques. In Ref. [16] we achieved promising results for the Edwards–Anderson spin glass with the combination of the present method and the renormalization-group approach by treating groups of spins with stable relative orientations as single block spins, analogously to Ref. [8, 9] (see also details in Chapter 3).

Only the results of further investigations will decide whether hysteretic optimization becomes a competitive practical optimization tool used by a wider audience, or whether it simply remains an interesting curiosity.

Acknowledgments

The work has been supported by Hungarian grants OTKA T037212 and T037991.

References

[1] E. H. L. Aarts and P. J. M. van Laarhoven, *Simulated annealing: an introduction*, Statistica Neerlandica **43**, 31 (1985).

[2] E. Aarts and J. Korst, *Simulated annealing and Boltzmann machines*, (Whiley, Chichester, 1989).

[3] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, (Addison–Wesley, Reading, MA, 1989).

[4] A.K. Hartmann, *How to evaluate ground-state landscapes of spin glasses thermodynamical correctly*, Eur. Phys. J. B **13**, 539 (2000).

[5] A.K. Hartmann, *Scaling of stiffness energy for three-dimensional* $\pm J$ *Ising spin glasses*, Phys. Rev. E **59**, 84 (1999).

[6] A.K. Hartmann and H. Rieger, *Optimization algorithms in physics*, (Wiley-VCH, Berlin 2001).

[7] J.H. Holland, *Adaptation in natural and artificial systems*, (The University of Michigan Press, Ann Arbor, 1975).

[8] J. Houdayer and O.C. Martin, *Renormalization for discrete optimization*, Phys. Rev. Lett. **83**, 1030 (1999).

[9] J. Houdayer and O.C. Martin, *A hierarchical approach for computing spin glass ground states*, Phys. Rev. E **64**, 056704 (2001).

[10] M. Huang, F. Romeo and A. Sangiovanni-Vincentelli, *An efficient general cooling schedule for simulated annealing*, in *Proc. IEEE Int. Conf. on Computer Aided Design,* 381 (IEEE, Piscataway, NJ, 1986).

[11] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, *Optimization by simulated annealing*, Nature **220**, 671 (1983).

[12] K.F. Pál, *The ground state energy of the Edwards–Anderson Ising spin glass with a hybrid genetic algorithm*, *Physica A* **223**, 283 (1996).

[13] K.F. Pál, *The ground state of the cubic spin glass with short-range interactions of Gaussian distribution*, *Physica A* **233**, 60 (1996).

[14] K. F. Pál, *Hysteretic optimization for the traveling salesman problem*, Physica A **329**, 287 (2003).

[15] F. Pázmándi, G. Zaránd and G. T. Zimányi, *Self-organized criticality in the hysteresis of the Sherrington–Kirkpatrick model*, Phys. Rev. Lett. **83**, 1034 (1999).

[16] G. Zaránd, F. Pázmándi, K.F. Pál, and G. T. Zimányi, *Using hysteresis for optimization*, Phys. Rev. Lett. **89**, 150201 (2002).

11 Extremal Optimization

Stefan Boettcher

Physical processes have inspired many optimization heuristics. Most famously, variants of simulated annealing and genetic algorithms are widely used tools for the exploration of many intractable optimization problems. But the breadth and complexity of important real-life problems leaves plenty of room for alternatives to verify or improve results. One truly alternative approach is the *extremal optimization* method. Basically, *extremal optimization* focuses on eliminating only extremely bad features of a solution while replacing them at random. Good solutions emerge dynamically in an intermittent process that explores the configuration space widely. This method may share the evolutionary paradigm with genetic algorithms, but assigns fitnesses to individual variables within a single configuration. Hence, it conducts a local search of configuration space similar to simulated annealing, but it was intentionally conceived to leave behind the certainties (and limitations) of statistical equilibrium along a temperature schedule, handing control (almost) entirely to the update dynamics itself. In fact, as a few simple model problems reveal, the extremal update dynamics generically leads to a sharp transition between an ergodic and a non-ergodic ("jammed") search regime. Adjusting its only free parameter to the "ergodic edge," as predicted by theory, indeed leads to optimal performance in numerical experiments. Although our understanding of this heuristic is only at its beginning, some quite useful applications have already been devised.

11.1 Emerging Optimality

Many natural systems have, without any centralized organizing facility, developed into complex structures that optimize their use of resources in sophisticated ways [2]. Biological evolution has formed efficient and strongly interdependent networks in which resources rarely go to waste. Even the morphology of inanimate landscapes exhibits patterns that seem to serve a purpose, such as the efficient drainage of water [19, 57].

Natural systems that exhibit self-organizing qualities often possess a common feature: a large number of strongly coupled entities with similar properties. Hence, at some coarse level they permit a statistical description. An external resource (such as sunlight in the case of evolution) drives the system which then takes its direction purely by chance. Just as descending water that, driven by gravity, breaks through the weakest of all barriers in its wake, biological species are coupled in a global comparative process that persistently washes away the least fit. In this process, unlikely but highly adapted structures surface inadvertently. Optimal adaptation thus emerges naturally, from the dynamics, simply through a selection *against* the

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

extremely "bad". In fact, this process may prevent the inflexibility inevitable in a controlled breeding of the "good".

Various models relying on extremal processes have been proposed to explain the phenomenon of self-organization [48]. In particular, the Bak–Sneppen model of biological evolution is based on this principle [3, 15]. Species are located on the sites of a lattice, and have an associated "fitness" value between 0 and 1. At each time step, the one species with the smallest value (poorest degree of adaptation) is selected for a random update, having its fitness replaced by a new value drawn randomly from a flat distribution on the interval $[0, 1]$. But the change in fitness of one species impacts the fitness of interrelated species. Therefore, all of the species at neighboring lattice sites have their fitness replaced with new random numbers as well. After a sufficient number of steps, the system reaches a highly correlated state known as self-organized criticality (SOC) [4]. In that state, almost all species have reached a fitness above a certain threshold. These species, however, possess *punctuated equilibrium* [27]: only one's weakened neighbor can undermine one's own fitness. This co-evolutionary activity gives rise to chain reactions called "avalanches", large fluctuations that rearrange major parts of the system, potentially making any configuration accessible.

11.2 Extremal Optimization

Although co-evolution may not have optimization as its exclusive goal, it serves as a powerful paradigm. We have used it as motivation for a new approach [16] to approximate hard optimization problems [32]. The heuristic we have introduced, called *extremal optimization* (EO), follows the spirit of the Bak–Sneppen model, updating those variables which have among the "worst" values in a solution and replacing them at random without ever explicitly improving them.

11.2.1 Basic Notions

To introduce EO, let us consider a spin glass [44] as a specific example of a hard optimization problem. An instance may consist of a d-dimensional hypercubic lattice of size $n = L^d$, as in the case of the Edwards–Anderson model [22], or simply of a randomly connected, sparse graph of n vertices, as in the case of the Viana–Bray model [59]. An Ising spin variable $x_i = \pm 1$ is placed at each site $i \leq n$. Spins are connected to each of their nearest neighbors j via a random bond variable $J_{i,j}$ drawn from some distribution, and $J_{i,j} = 0$ for all unconnected pairs of spins. In this example, we simply focus on a discrete distribution $P(J) = \delta(J^2 - 1)$ of bond weights. The configuration space Ω consists of all feasible configuration vectors $\mathbf{x} \in \Omega$, where $|\Omega| = 2^n$ here.

The hard optimization problem then consists of an instance of a randomly assembled, fixed bond matrix $\mathbf{J}$ that connects a set of n spin variables x_i, for which we wish to minimize the cost function, or Hamiltonian

$$H(\mathbf{x}) = -\frac{1}{2} \sum_i \sum_j J_{i,j}\, x_i\, x_j. \tag{11.1}$$

Configurations $\mathbf{x}$ of lowest global energy are called ground states. At low-temperature, these are ostensibly the preferred states of the system (*if* they could be attained by the dynamics).

Arrangement conflicts between connected spins leads to frustration, making ground-state configurations hard to find, and it has been shown that for non-planar graphs the problem is among the hardest of optimization problems [7].

In general, to find near-optimal solutions for a particular optimization problem, EO performs a neighborhood search on a single configuration $\mathbf{x} \in \Omega$ of a given instance of a problem. Rearranging merely $O(1)$ variables in each update is a characteristic of a local search, in contrast to a genetic algorithm, say, where cross-over operations may effect the arrangement of $O(n)$ variables. In the spin problem, we typically consider single spin-flips to move from one configuration to the next. The cost function $H(\mathbf{x})$ is assumed to consist of the individual cost contributions, or "fitnesses", λ_i for each variable x_i (analogous to the fitness values in the Bak–Sneppen model from above). The fitness of each variable assesses its contribution to the total cost:

$$H(\mathbf{x}) = -\sum_i \lambda_i. \tag{11.2}$$

It is a crucial feature of hard optimization problems that the fitness λ_i depends on the state of x_i in relation to variables that x_i is connected to. If these connections are frustrated, some variable can only improve their fitness at the expense of neighbors. For the Hamiltonian in Eq. (11.1), we assign to each spin x_i the fitness

$$\lambda_i = \frac{1}{2} x_i \sum_j J_{i,j}\, x_j, \tag{11.3}$$

so that Eq. (11.2) is satisfied. Each spin's fitness thus corresponds to (the negative of) its local energy contribution to the overall energy of the system. In a similar way to the Bak–Sneppen model, EO then proceeds through a local search of Ω by sequentially changing variables with "bad" fitness on each update. To this end, EO establishes a rank-ordered list

$$\lambda_{\Pi(1)} \leq \lambda_{\Pi(2)} \leq \ldots \leq \lambda_{\Pi(n)}. \tag{11.4}$$

of fitnesses via a permutation Π that maps the fitness ranks onto the index of the associated variable, with the lowest fitness, $\lambda_{\Pi(1)}$, corresponding to the "worst" variable, x_j with $j = \Pi(1)$, at the bottom of the list. After each update, the fitnesses of the changed variable and of all its neighbors are re-evaluated according to Eq. (11.3).

11.2.2 EO Algorithm

In terms of an abstract algorithm, we can describe EO as follows. First, EO is a local search [32] on an instance **J**, and we can generically write:

```
algorithm LocalSearch($\mathbf{J}$,StopCondition)
begin
    $\mathbf{x}$:=InitConfiguration($\mathbf{J}$);
    $H_{\text{best}}$:=Cost($\mathbf{x}$);
    repeat
    begin
        $j$:=EvaluateState($\mathbf{x}, \mathbf{J}$);
        $x_j$ :=Update($j, \mathbf{x}, \mathbf{J}$);
        $H$:=Cost($\mathbf{x}$);
        if $H < H_{\text{best}}$ and IsFeasible($\mathbf{x}, \mathbf{J}$) then
        begin
            $\mathbf{x}_{\text{best}} := \mathbf{x}$;
            $H_{\text{best}} := H$;
        end
    end
    until StopCondition;
return $H_{\text{best}}, \mathbf{x}_{\text{best}}$;
end
```

In a local search by simulated annealing (SA) [36], for instance, `EvaluateState` would merely pick a new variable j at random for `Update` to apply the Metropolis procedure, which either accepts or rejects a change to x_j depending on the "state" of the system $(\mathbf{x}, \mathbf{J})$ and the temperature. In SA, `Update` would also have to maintain a list of tunable parameters, such as the temperature schedule and the acceptance rate, the latter determining `StopCondition` in a non-trivial way.

In contrast, in a local search with EO, `EvaluateState` would do most of the work:

```
algorithm EvaluateState.EO($\mathbf{x}, \mathbf{J}$)
begin
    $\lambda$:=EvaluateFitness($\mathbf{x}$,$\mathbf{J}$);
    $\Pi$:=RankOrder($\lambda$);
    $k$:=SelectRank;
return $j := \Pi(k)$;
end
```

In turn, `Update` becomes simply:

```
algorithm Update.EO($j, \mathbf{x}, \mathbf{J}$)
begin
return $x_j$:=Flip($x_j$);
end
```

where `Flip` assigns the selected variable a new value [here, `Flip`$(x_j) = -x_j$] *unconditionally*, i.e., independent of the state of the system $(\mathbf{x}, \mathbf{J})$ before or after the flip. For instance, in the spin glass from Section 11.2.1, a selected spin must be flipped, even if its local field is positive. Note, that there is no parameter in this process that would require any tuning. Clearly, without an acceptance criterion, the information that drives the process toward better solutions must be provided by the rank-ordered list of fitnesses, Eq. (11.4), that is manipulated in `EvaluateState.EO`. Furthermore, the absence of an acceptance criterion means that EO never "freezes" into a terminable configuration, which could provide a `StopCondition`. Instead, EO always keeps on searching to find even better solutions (and possibly to explore the degeneracy of states $\mathbf{x}_{\text{best}}$, see Section 11.3.1). It terminates simply after a desired number of updates: `StopCondition`$= (t > t_{\max})$, although more adaptive termination conditions are discussed in Ref. [11].

Within `EvaluateState.EO`, `EvaluateFitness` calculates the current fitness of each variable according to Eq. (11.3). Subsequently, `RankOrder` establishes the ranking of fitnesses in Eq. (11.4). The next step is crucial for the EO-algorithm, because `SelectRank` provides the only adjustable means to transfer information from the ranking to the local search. This ranking of the variables that are currently "better" than those below them provides the only measure of quality of a configuration. It is merely the memory encapsulated in the ranking that directs EO into the neighborhood of increasingly better solutions. If we impose an "extremal" selection condition in `SelectRank`, focusing on *atypically* bad variables, we may expect to facilitate a search dynamics similar to the Bak–Sneppen model, with large fluctuations but frequent returns to near-optimal states. In contrast to an equilibrium situation, those "better" variables possess *punctuated equilibrium* [27]: their memory gets altered only when a neighboring variable is forced to change. The unconditional acceptance affects large fluctuations in the cost function that can accumulate in a sequence of updates. Merely the bias *against* extremely "bad" fitnesses facilitates frequent returns to improved solutions.

These characteristics of the performance of EO highlight two additional advantages of EO. For one, EO can take full advantage of any pre-existing information about the solution in its initial configuration to expedite convergence [18]. This may not be the case with other heuristics, for instance, SA has to start at some sufficiently high temperature, which would in most cases instantly destroy whatever information may be contained in the initial configuration. Secondly, the fact that EO does not freeze but instead sustains large fluctuations throughout, allows it not only to converge to a single solution, but to explore Ω more extensively. For example, in one sweep EO could potentially sample the entire set of ground-state configurations to measure also more complex observables such as their entropy (logarithm of the degeneracy) or the overlap as discussed in Section 11.3.

11.2.3 Extremal Selection

The simplest selection procedure would simply follow the Bak–Sneppen model, and choose `SelectRank`$=\Pi(1)$, i.e., always pick one of the variables with the worst fitness. In this case, our EO-algorithm would be entirely free of adjustable parameters. This "basic" EO-algorithm often leads to quite satisfactory results, especially when updating the same variable as there is more than just one choice in outcome [16] or fitnesses are degenerate (see Figure 11.1). In the case of the spin glass, we have only one choice, $x_j := -x_j$, and a deterministic local search results (aside from degeneracies), if we only update the lowest-ranked variable. Such a search will most likely reach a "dead end" in the form of a poor local minimum. In contrast, adding a single, fixed parameter to `SelectRank` will allow us to tune EO for a large number of problems exactly to a phase boundary with a balanced dynamics of fluctuations and returns.

To this end, consider a scale-free probability distribution over the *ranks* k in Eq. (11.4),

$$P_k \propto k^{-\tau}, \qquad 1 \leq k \leq n, \tag{11.5}$$

for a fixed value of the parameter τ. Then the algorithm τ-EO is specified by the following choice for selecting variables for an update [14, 16]:

```
**algorithm** SelectRank($\tau$)
**begin**
    $k := \left(1 + \left(n^{1-\tau} - 1\right) \mathrm{rng}()\right)^{1/(1-\tau)}$;
**return** $j := \Pi(k)$;
**end**
```

Here `rng()` produces uniform random numbers on the unit interval.

For $\tau = 0$, the τ-EO algorithm is simply a random walk through Ω. Conversely, for $\tau \to \infty$, the process approaches the "basic" EO algorithm. However, for finite values of τ the choice of a *scale-free* distribution for P_k in Eq. (11.5) ensures that no rank gets excluded from further evolution, while still maintaining an "extremal" bias against variables with bad fitness. In fact, it appears the τ-EO works best when τ is tuned exactly to the boundary between a "too ergodic" ($\tau \to 0$) and an "ergodicity-broken" [6] ($\tau \to \infty$) phase of the search dynamics.

As suggested by Figure 11.1, careful numerical experiments show that a value of

$$\tau - 1 \sim (\ln n)^{-1} \qquad (n \to \infty) \tag{11.6}$$

seems to work best [18]. In Section 11.4 we will discuss simple model problems for which the asymptotic behavior of τ-EO can be solved exactly [14]. The model reproduces Eq. (11.6) exactly in cases where the model develops a "jam" among its variables, which is quite a generic feature of frustrated systems. After many update steps most variables freeze into a near-perfect local arrangement and resist further change, while a finite fraction remains frustrated in a poor local arrangement. More and more of the frozen variables have to be dislocated collectively to accommodate the frustrated variables before the system as a whole can improve its state. In this highly correlated state, slow variables block the progression of fast variables, and a "jam" emerges. The asymptotic analysis of the flow equations for a jammed system lead to Eq. (11.6).

11.2.4 Rank Ordering

As a price for its emergent capabilities, EO has to pay an overhead in computational cost for evaluating and sorting the fitnesses according to Eqs. (11.3)–(11.4) [16]. The overhead is not nearly as bad as the above algorithm `EvaluateState.EO` may suggest. Yet, despite the simplifications discussed below, rank ordering dominates the computational cost of any EO-algorithm; and more refined data structures could impact the performance of EO algorithms significantly. In the numerical experiments so far, we have emphasized conceptual aspects of the algorithm.

To establish a sufficient ordering of fitnesses, we often do not need to re-evaluate *all* fitnesses in `EvaluateFitness` or re-order *all* ranks in `RankOrder`. In particular, in a sparse

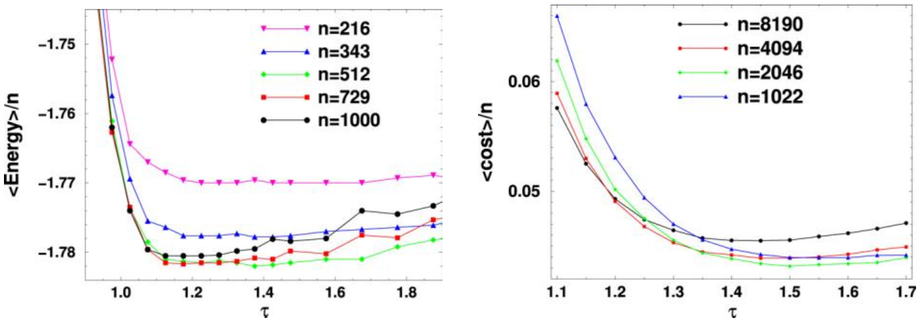


Figure 11.1: Plot of the average costs obtained by EO for a $\pm J$ spin glass (left) and for graph bipartitioning (right), all as a function of τ. A number of instances were generated at each n. For each instance, 10 different EO runs of the same duration were performed at each τ. The results were averaged over runs and over instances. Although these problems are quite distinct, in either case the best results are obtained at a value of τ that behaves according to Eq. (11.6), as predicted by the model in Section 11.4, see also Figure 11.14.

system, variables may have an average connectivity $\alpha = O(1)$, so each update effects only $\sim \alpha$ variables, i.e., only the updated spin and its neighbors change fitness, hence only a few entries in Π need to be changed.

While a perfectly ordered list would contribute a factor of $\alpha n \ln(n)$ to the computational cost of EO, one can actually get away with imperfectly ordered fitnesses using a "heap" [16] ($\sim \alpha \ln n$) or even a hash-table ($\sim \alpha$). In the case of the "basic" EO algorithm, such a heap would already be exact, since we only care about the variable at the top of the heap, $\Pi(1)$. Furthermore, in the case of a lattice spin glass with a $\pm J$ bond distribution, where each spin can take on only one of $2d + 1$ possible fitness values, a hash-table with $2d + 1$ buckets is actually exact. In general, in our experience heaps or hash-tables usually provide a sufficient ordering of fitnesses for τ-EO even when they are only approximate, due to the inherently stochastic selection process.

Clearly, for high-connected systems, such as the Sherrington–Kirkpatrick model (SK) [56], where the connectivity matrix $\mathbf{J}$ is dense ($\alpha = n$), EO would not appear to be very efficient. Therefore, we have restricted ourselves to systems with low connectivity (sparse graphs). But it should be pointed out that EO in principle may work for such problems as well, and possibly could regain its overhead expense for sorting through superior convergence. In SK, updating a single spin does effect *all* other spins, but while the fitness of the updated spin itself makes a relative change of $O(1)$, all other fitnesses only change by $O(1/n)$, as does the total energy. Hence, most spins's ranking would hardly change, except for the updated spin itself. On contrast, in problems where each update of even a single variable effects almost all other variables and the total energy by $O(1)$, the rejection-less approach of EO is hopeless, since there is no memory left in the ranking from one update to the next. An example for this worst case for EO is a polymer-folding problem [58], for which a high threshold against acceptance of a move seems essential.

11.2.5 Defining Fitness

A more fundamental restriction on the applicability of EO is imposed by the requirement of defining some form of comparable fitness for discernible entities within a problem. For example, in the traveling salesperson problem (TSP) it is *a priori* not clear whether the individual tour-segments or the "cities" should be the appropriate "variables" to update [16]. Also, to find the minimum of some polynomial in n variables on some finite domain, discernible variables seem apparent but their fitness is not. Furthermore, for heterogeneous problems combining variables of very different nature (say, workers, machines, and products in an industrial application), an approach based only on evaluating overall changes of a cost function, as used by SA, seems more convenient (albeit not necessarily more successful).

Even in the domain of $\pm J$ spin glasses with clearly distinguishable variables having local energy contributions to the Hamiltonian, the definition of "fitness" itself requires a second thought. On a regularly-connected graph, such as a lattice, each variable has exactly the same number $\alpha_i \equiv \langle\alpha\rangle = 2d$ of neighbors, and hence exactly the same range of fitness, $-\alpha_i \leq \lambda_i \leq \alpha_i$. But if connectivities α_i vary between variables, like in the Viana–Bray model, or even for a regular lattice but with varying bond weights (as in a Gaussian bond distribution), the range of fitnesses, as defined in Eq. (11.3), does not coincide between variables. For instance, in a sparse graph some spins will be entirely unconnected ($\lambda_i \equiv 0$). Although these spins are always in a "perfect" state, their fitness may be worse than some highly connected spin which happens to satisfy more than half of its bonds (by weight) but whose violations of its other bonds can contribute unreasonably to the overall energy. In fact, simply by aligning with its local field *every* spin x_i can be put into a state where $\lambda_i \geq 0$ in Eq. (11.3).

In most cases, this problem is easily solved with an all-too-obvious linear transformation of the cost function. After all, what matters for the optimization problem is not the energy *per se* but the total weight $C(\mathbf{x})$ of all frustrated bonds in the system. This positive-definite quantity is usually called the "cost" of a configuration, and is related to the energy in Eq. (11.2) by

$$C(\mathbf{x}) = \frac{1}{2}\,H(\mathbf{x}) + \frac{1}{2}\sum_{i<j} |J_{i,j}|\,. \tag{11.7}$$

Note that for discrete bonds, $J_{i,j} = \pm 1$, the sum simply yields the total number of connections in the system. Using C as the cost function, it is natural to redefine as fitness

$$\lambda_i = -\frac{1}{2}\sum_{j} |J_{i,j}|\,\theta\left(-J_{i,j}x_i x_j\right), \tag{11.8}$$

where θ is the Heaviside 0-to-1 step function. Each fitness is just the (negative) sum of the absolute weights of all *violated* bonds attached to each spin (times $1/2$, since the bond is shared with a neighbor). In this case, highly connected spins that can contribute a lot to the cost find themselves at the bottom of the ranking even though they also possess a majority of satisfied bonds, while totally unconnected bonds always have an optimal fitness of zero. Had we instead erroneously defined fitness as the (positive) sum of the absolute weights of all satisfied bonds, we would perpetually update those low-connected spins that even in a perfect state would have very little fitness compared to high-connected, imperfect spins.

The definition of fitness can be a subtle subject and is still poorly explored [16]. While for many problems there is often an obvious choice to define the fitness, it is worthwhile to consider and try even small variations. For instance, we have made the following observation. In situations for sparse graphs with (say, Poissonian) distributed connectivities α_i, the range of accessible fitnesses for each variable do not coincide, even as redefined in Eq. (11.8). Borrowing from the Bak–Sneppen model again, we have "normalized" all fitnesses into the unit interval, to wit,

$$\lambda_i = 1 - \frac{\sum_j |J_{i,j}|\,\theta\left(-J_{i,j}x_i x_j\right)}{\sum_j |J_{i,j}|}, \tag{11.9}$$

defining $\lambda_i = 1$ for unconnected variables. Note that this definition of fitness does *not* reproduce Eq. (11.2) or Eq. (11.7)! Yet, it is our experience that treating each variable *on equal footing* often results in faster convergence, if not better quality overall [18], although the fitnesses are not functionally related to the cost function. Thus, moving low-connected variables, which seemingly contribute little to the global energy, is important to dislodge "jammed" configurations and to progress toward better solutions. And although the ranking of fitnesses does not possess information about the cost function directly, it guides the local search into the vicinity of optima where fluctuations ensure that EO "trips" over good solutions inadvertently.

11.2.6 Distinguishing EO from other Heuristics

We hope to demonstrate that EO provides an *alternative philosophy* to the canon of heuristics [32]. Conceiving new, distinct methods improves the chances that at least one of the known heuristics will provide good results on some particular problem when all others fail; no general-purpose method will ever be a panacea! Comparisons of heuristics on standard or randomly generated testbeds is an instructive way to assess the capabilities of any newly conceived method, and to establish itself, the EO methods could used a lot more of those comparisons. In Section 11.3 we will discuss the results of some of our own comparison studies.

As discussed in Section 11.2.5, the most apparent distinction between EO and other methods is the need to define local cost contributions for each variable, instead of merely a global cost. EO's ability to access this local information directly constitutes its power as well as it limits its applicability. In the following, we will clarify the distinctions between EO and other search heuristics.

Simulated Annealing (SA) [36] emulates the behavior of frustrated systems in *thermal equilibrium:* if one couples such a system to a heat bath of adjustable temperature, by cooling the system slowly one may come close to attaining a state of minimal energy (i.e., cost). SA accepts or rejects local changes to a configuration according to the Metropolis algorithm at a given temperature, enforcing equilibrium dynamics ("detailed balance") and requiring a carefully tuned "temperature schedule."

In contrast, EO drives the system *far from equilibrium:* aside from ranking, it applies no decision criteria, and all new configurations are accepted indiscriminately. Instead of tuning a whole schedule of parameters, EO often requires few choices. It may appear that EO's results would resemble an ineffective random search, similar to SA at a fixed but finite temperature (see Ref. [21]). But in fact, by persistent selection against the worst fitnesses, EO quickly approaches near-optimal solutions. EO maintains significant fluctuations even at late runtimes (see Figure 11.2), crossing sizable barriers to access new regions in configuration space, whereas SA terminates by freezing into one particular solution for good.

In some versions of SA, low acceptance rates near freezing are circumvented using a scheme of picking trials from a rank-ordered list of possible moves [21,28] (see Chapter 2.3.4 in Ref. [54]). As in EO, every move gets accepted. But these moves are based on an outcome-oriented ranking, favoring downhill moves but permitting (Boltzmann-)limited uphill moves. On the other hand, in EO the ranking of variables is based on the current, not the future, state of each variable, allowing for unlimited uphill moves.

Genetic Algorithms (GA) [26] are similarly motivated by biological evolution (with deceptively similar terminology, such as "fitness"), yet GA and EO algorithms have hardly anything in common. GAs, mimicking evolution on the genotype's level, keep track of entire "gene pools" of configurations from which to select and "breed" an improved generation of solutions. By comparison, EO, based on evolutionary competition at the phenomenological level of "species" [17] operates only via local updates on a single configuration, with improvements achieved merely by elimination of bad variables. EO, SA, and most other meta-heuristics use updates typical of a local search. Instead, in GA, cross-over operators perform global exchanges on binary encodings of pairs of configurations.

Taboo-Search (TS) performs a memory-driven local search procedure that allows for limited uphill moves based on scoring recent moves [25,54]. Its memory permits escapes from local minima and avoids recently explored configurations. It is similar to EO in that it may not converge and in that moves are ranked. But the uphill moves in TS are limited by tuned parameters that evaluate the memory, and, as for SA above, rankings and scoring of moves in TS are done on the basis of anticipated outcome, not on current "fitness" of individual variables.

11.2.7 Implementing EO

To demonstrate the applicability of EO for a range of different combinatorial optimization problems, we describe here a selection of possible implementations. Below, in Section 11.3, we will discuss some of the results obtained by us and other researchers with these or similar implementations. Many of these problems could as well be mapped directly onto the spin-glass problem introduced in Section 11.2. Often it is more straightforward to define fitness directly for the contribution that variables make to the overall cost function. Examples of hybrid algorithms based on EO will be discussed in Section 11.3.3.

Graph Bipartitioning (GBP): One popular hard optimization problem, to which EO has been applied successfully [9,16,18,20] is the GBP [34,36,42]. In the GBP, we are given a graph of n vertices, where n is even, and "edges" connecting certain pairs of vertices. The problem is to partition the vertices into two equal subsets, each of size $n/2$, minimizing as cost function $C(\mathbf{x})$ (called "cutsize") that counts the total weight from edges cutting across the partition, see Figure 11.2. Hence, each vertex' fitness is $\lambda_i = -(1/2)\{\sum$ weights of cut edges on $x_i\}$ to give $C(\mathbf{x}) = -\sum_i \lambda_i$.

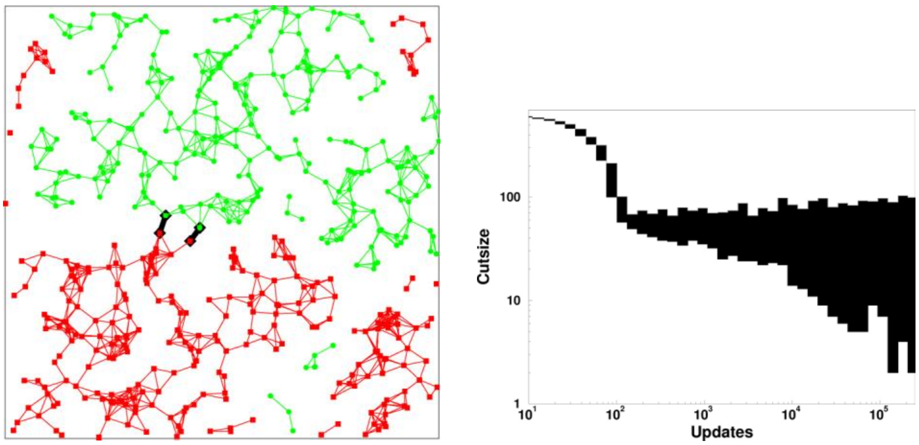


Figure 11.2: Example of a graph-bipartitioning problem. The left panel shows a random geometric graph of average connectivity $\alpha \approx 5$ with the optimal partition as found by the "basic" EO algorithm. There are 250 square and 250 round vertices, but there are only two edges (all of unit weight) connecting a round and a square vertex (thick lines). The right panel shows the evolution of the cutsize during an EO run on that graph. The shaded area marks the range of cutsizes explored in the respective time bins. The best cutsize ever found is 2, which is visited repeatedly in this run. In contrast to simulated annealing, which has large fluctuations in early stages of the run and then converges much later, extremal optimization quickly approaches a stage where broadly distributed fluctuations allow it to probe many local optima.

The size of the configuration space Ω grows exponentially with n, $|\Omega| = \binom{n}{n/2}$, since only unordered divisions of the n vertices into two equal-sized sets are feasible configurations $\mathbf{x}$. In this problem, frustration does not arise from bond disorder alone but also from the global constraint of *equal* partition. Limiting the feasibility of configurations $\mathbf{x}$, this constraint forces us to move *two* variables in each update (unless we introduce a "penalty function" without which SA seems to be unsuccessful for this problem [34]).

A typical move-class to update GBP is an "`exchange`" of one vertex from each subset. In the "basic" EO algorithm, we simply select the "worst" vertex $x_{\Pi(1)}$, and `exchange` it with a *randomly* selected vertex from the other subset. The evolution of $C(\mathbf{x})$ of this process is depicted on the right side of Figure 11.2, starting from a random marking of 250 red and 250 green vertices in the graph on the left. After an initial "mop-up" stage with a rapidly-dropping cutsize, large fluctuations in the cost persist for all times with frequent returns to increasingly better solutions. Even better results obtained with the τ-EO algorithm are discussed in Section 11.3.

Satisfiability (MAX-K-SAT) Instances of the satisfiability problem MAX-K-SAT consist of a formula composed of M clauses. Each clause contains K literals (i.e., x_i or $\neg x_i$), drawn randomly from a pool of n boolean variables x_i. A clause is verified, if at least one of its K literals is true (logical "or"), and the entire formula is verified only if every clause is true (logical "and"). Here, we try to maximize the number of true clauses by some configuration of the variables, accordingly the cost $C(\mathbf{x}) \leq 0$ is defined as the number of false clauses.

MAX-K-SAT has an obvious EO-implementation. For each variable we set $\lambda_i = -(1/K)\{$# of false clauses containing $x_i\}$, which satisfies $C(\mathbf{x}) = -\sum_i \lambda_i$. Typically, $K = O(1)$ and $M = O(n)$ so that each variable appears only in a few ($\alpha_i \approx M/n$) clauses, each connecting it to $\approx K$ other variables. In the light of Section 11.2.5, we define fitness here in terms of what is "bad" about each variable, since near-optimal configurations will have most variables in a nearly perfect state. Unfortunately, so far very little work has been done with EO for SAT. SAT is widely considered as the archetypal NP-hard problem because many other problems are conveniently mapped onto it.

Particularly promising would be an application of EO near the phase transition in satisfiability problems [33], which has been studied intensely in recent years [45,46]. The complexity of SAT transitions has been shown to be the ultimate challenge for any local search [37,55]. It would be interesting to explore the competitiveness of various EO implementations near such a phase transition.

Similarly, SAT problems are easily converted into the form of a spin Hamiltonian as in Eq. (11.1), and can be treated with the hybrid methods discussed in Section 11.3.3. This approach should be particularly successful on the high-complexity testbed ensembles defined in Refs. [37,55].

Graph Coloring (*K*-COL) Given K different colors to label the vertices of a graph, we need to find a coloring that minimizes as the cost $C(\mathbf{x}) \leq 0$ the absolute weight of "monochromatic" edges, i.e., those connecting vertices of identical color. Obviously, this problem corresponds to a K-state anti-ferromagnetic Potts model. EO for K-COL is implemented by defining $\lambda_i = -(1/K)\{\sum$ weights of monochromatic edges on $x_i\}$ as fitness. With this approach we have accurately determined the connectivity α_c for the phase transition of 3-COL on random graphs [17,47].

11.3 Numerical Results for EO

In the few years since we first proposed (EO) as a general purpose heuristic for some of the hardest combinatorial optimization problems [16], ample evidence has been provided for its practicality. Our own studies have focused on demonstrating elementary properties of EO in a number of implementations for classic NP-hard combinatorial problems [17,18], where EO provides a bound by explicit construction of at least one, and potentially *all,* configurations on the best-found cost. Comparative studies have shown that EO holds significant promise to provide a new, *alternative* approach to approximate many intractable problems [8,9,16,21,39]. Those investigations have also indicated certain universal performance features [18,20] that were reproduced with a simple theory [14] described in Section 11.4.

Initially, we have established EO on a set of classic combinatorial optimization problems, such as graph bipartitioning [9,16,18], three-coloring [17], and the traveling salesperson [16]. These implementations proved EO to be very successful on NP-hard partitioning and matching problems in comparison with other heuristics, such as Kernighan–Lin [34] and METIS [35], or with other stochastic optimization methods, such as simulated annealing (SA) [36] or genetic

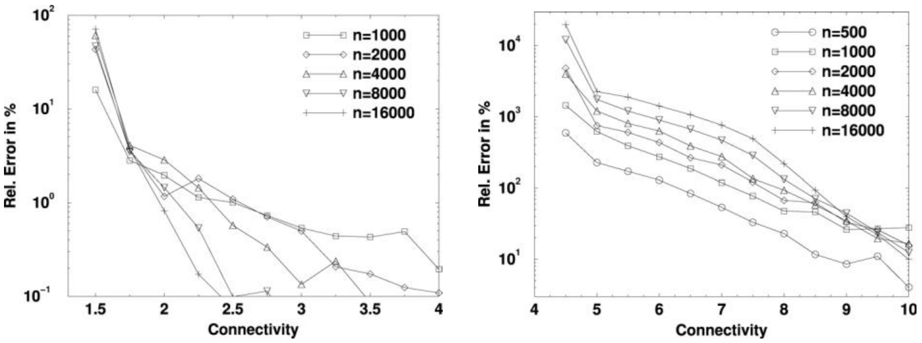


Figure 11.3: Plot of the error in the best result of SA relative to EO's on identical instances of random graphs (left) and geometric graphs (right) as a function of the mean connectivity α. The percolation points are at $\alpha_p = 1$ and $\alpha_p \approx 4.5$, respectively, and the critical points for the GBP (i.e., the first time a component of size $> n/2$ appears) are slightly above that (e.g. at $\alpha_c = 2\ln 2 = 1.386$ for random graphs [42]). SA's error relative to EO near the critical point in each case rises with n.

algorithms (GA) [26], either averaged over ensembles of instances [9] or on standard testbeds with up to 10^5 variables [16]. But our study of the traveling salesperson problem (TSP) also showed the limitations of EO, outperforming SA only for the non-Euclidean case and being, along with SA, far behind the iterated Lin–Kernighan heuristic [53] on this problem.

Several other researchers have picked up on our initial results, and have successfully applied EO to problems as diverse as pattern recognition [41], signal filtering of EEG noise [63], segregation of binary mixtures [8], artificial intelligence [39], and $3d$ spin-glass models [21, 60].

11.3.1 Early Results

11.3.1.1 Results for Graph Bipartitioning

In a numerical study of the GBP we have shown [9] that τ-EO can outperform SA [34, 36] near the partitioning transitions on random and geometric graphs, see Figure 11.3. At this transition, located just above the percolation point, cutsizes first become non-zero.

Studies of the GBP on the average rate of convergence toward better-cost configurations as a function of runtime t indicate power-law convergence, roughly like [18]

$$C_{\text{best}}(t) \sim C_{\text{opt}} + A\,t^{-0.4}, \tag{11.10}$$

where $C_{\text{best}}(t)$ refers to the (average) *best* results found by EO up to time t. For example, with regard to Figure 11.2, this quantity would be a (monotonically decreasing) lower envelope to the actual fluctuations marked by black bars in the runtime plot on the right. Note that this power-law convergence is already apparent there.

Of course, it is not easy to assert for graphs of large n that those runs in fact converge anywhere close to the true optimum C_{opt}, but the finite-size scaling analysis in Figure 11.4 indicates consistent convergence for any n [18]. The asymptotic value of C_{opt} extracted for the

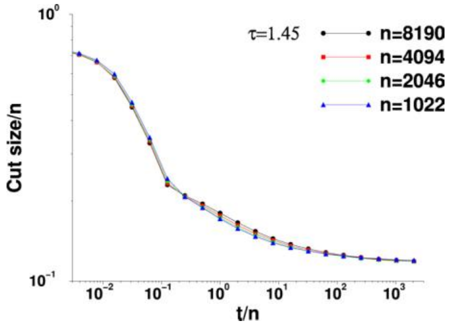


Figure 11.4: Scaling collapse of the τ-EO time evolution of $C_{\text{best}}(t)$ for the GBP on three-connected Bethe lattices, averaged over 8 separate runs on 32 instances for each system size n. For large $t(\gg n)$ the data collapses onto a single scaling curve given in Eq. (11.10) as a function of t/n, here at $\tau = 1.45$.

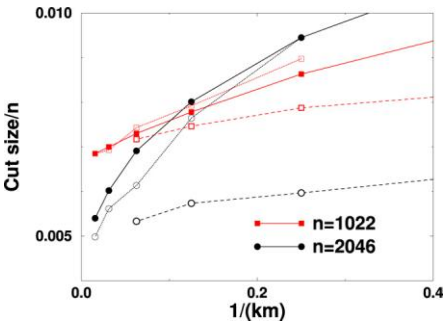


Figure 11.5: Equivalent run-time comparison of different strategies for τ-EO (here, $\tau = 1.3$) on a set of geometric graphs of sizes $n = 1022$ (squares) and 2046 (circles). The average best-of-k cutsizes found after t updates for k restarts are plotted as function of $1/(km)$, where $m = t/t_{\max}$ is the multiple of a fixed time-scale. Filled symbols refer to fixed $m = 1$ but varying k. Open symbols on the dotted lines refer to $k = 4$ and m varying, as do opaque symbols on dashed lines for which clustering initial conditions were used.

GBP on three-connected Bethe lattices (also called trivalent graphs) from Figure 11.4 can be compared with previous simulations [5]. There, the "energy" density $e = -1 + 4C_{\text{opt}}/(3n)$ was calculated using the best results obtained for a set of those graphs. Using simulated annealing Ref. [5] obtained $e = -0.840$. Our current extrapolation yields a cost per spin of $C_{\text{opt}}/n = 0.1158$ or $e = -0.845(1)$. If these results are accurate, the replica symmetric (RS) solution proposed in Refs. [42, 62] for this version of the GBP, which gives a value of $e = -2 \times 0.7378/\sqrt{3} = -0.852$, would be excluded.

In Figure 11.5 we have investigated the effect of providing EO with pre-optimized initial conditions (in `InitConfiguration` in Section 11.2.2) and of trading a given amount of update steps $t_{\max}$ between k restarts. While good start-up conditions (here, using an algorithm that clusters neighboring vertices together into one of the two partitions) can be utilized by EO to result in much faster convergence at short runtimes, the diffusive behavior of the local search seems to ensure that the effect of those improved start-ups eventually gets "forgotten." That data also suggests that it seems mildly advantageous to invest into few long runs instead of dividing it between many separate restarts, which we have converted into an adaptive strategy [11].

11.3.1.2 Results on Spin Glasses

Table 11.1 lists the results obtained with τ-EO at $\tau = 1.15$ for the $\pm J$ spin glass on a hypercubic lattice in $d = 3$ and 4 [17]. In comparison, the EO results provide an independent confirmation of previous genetic algorithm computations [30, 31, 50] (which were exclusively

Table 11.1: EO approximations to the average ground-state energy per spin $e_d(n) = H/n$ of the $\pm J$ spin glass of size $n = L^d$ in $d = 3$ and 4, compared with genetic algorithm results from Refs. [30,31,50]. Also shown is the average time t (in seconds) needed for EO to find the presumed ground state, on a 450 MHz Pentium.

L	$e_3(n)$	t	Ref. [50]	Ref. [30]	$e_4(n)$	t	Ref. [31]
3	−1.6712(6)	0.0006	−1.67171(9)	−1.6731(19)	−2.0214(6)	0.0164	−2.0222(16)
4	−1.7377(3)	0.0071	−1.73749(8)	−1.7370(9)	−2.0701(4)	0.452	−2.0685(4)
5	−1.7609(2)	0.0653	−1.76090(12)	−1.7603(8)	−2.0836(3)	8.09	−2.0850(3)
6	−1.7712(2)	0.524	−1.77130(12)	−1.7723(7)	−2.0886(6)	86.3	−2.0908(2)
7	−1.7764(3)	3.87	−1.77706(17)		−2.0909(12)	1090.	−2.0926(3)
8	−1.7796(5)	22.1	−1.77991(22)	−1.7802(5)			
10	−1.7832(5)	424.	−1.78339(27)	−1.7840(4)			
12	−1.7857(16)	9720.	−1.78407(121)	−1.7851(4)			

designed for spin-glass models on hypercubic lattices), with roughly the same computational overhead. But EO is conceptually simpler, has only one parameter (τ), and is more general in its applicability.

To gauge τ-EO's performance for larger $3d$-lattices, we have run our implementation also on two instances, $toruspm3$-8-50 and $toruspm3$-15-50, with $n = 512$ and $n = 3375$, considered in the $7th$ DIMACS challenge for semi-definite problems[1]. The best available bounds (thanks to F. Liers) established for the larger instance are $H_{\text{lower}} = -6138.02$ (from semi-definite programming) and $H_{\text{upper}} = -5831$ (from branch-and-cut). EO found $H_{\text{EO}} = -6049$ (or $H/n = -1.7923$), a significant improvement on the upper bound and already lower than $\lim_{n\to\infty} H/n \approx -1.786\ldots$ found in Table 11.1. Furthermore, we collected 10^5 such states, which roughly segregate into three clusters with a mutual Hamming distance of at least 100 distinct spins; though at best a small sample of the $\approx 10^{73}$ ground states expected [29]! For the smaller instance the bounds given are -922 and -912, while EO finds -916 (or $H/n = -1.7891$) and was terminated after finding 10^5 such states. While this run (including sampling degenerate states) took only a few minutes of CPU (at 800 MHz), the results for the larger instance required about 16 hours.

In Figures 11.6–11.9, τ-EO (with fixed $\tau = 1.3$) was used to explore the ground-state energies *and* entropies of spin glasses on Bethe lattices (BL) of various connectivities $k+1 \leq 25$ [10, 11]. As mentioned in Section 11.2.2, EO is well suited to obtain the ground-state entropies by enumerating all states of the lowest energy. Since EO never "freezes" this could be done in a single run. For larger instance sizes n, better accuracy is obtained with an adaptive multiple restart method as described in Ref. [11]. These simulations led to the most accurate numerical confirmation to date of theoretical predictions using replica symmetry breaking (RSB) [44] for any finite-connectivity model, an important step toward understanding more realistic finite-dimensional systems. In Figure 11.6 we show our extrapolation of the τ-EO data for the ground-state energy in the three-connected Bethe lattice, which coincides to within 0.1% with recent RSB calculations [43]. In Figure 11.7 we plot the extrapolation of the entropies obtained by EO for the same system. While there are no RSB predictions for the

[1] http://dimacs.rutgers.edu/Challenges/Seventh/

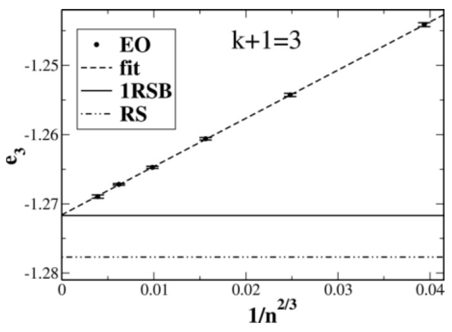


Figure 11.6: Extrapolation of ground-state energies obtained with EO for 3-connected Bethe lattices of sizes $n \leq 4096$, plotted vs. $1/n^{2/3}$. The energies extrapolate to $e_3 = -1.2716(1)$ [11], way above the RS result (-1.277) but consistent with the 1RSB result of -1.2717 (horizontal lines) [43].

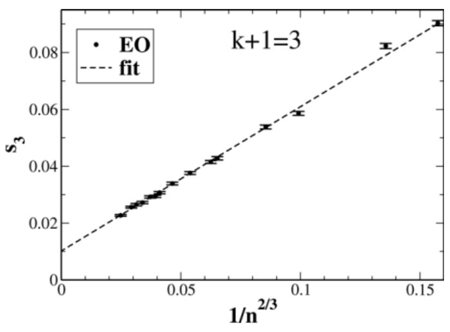


Figure 11.7: Same as Figure 11.6 but for ground-state entropies for lattices of sizes $n \leq 256$. The entropies extrapolate to $s_3 = 0.010(1)$. To obtain this data, EO has to explore almost all ground-state configurations. Only the computational cost of maintaining a list of such states without overcounts limits n.

ground-state entropy yet, the extrapolated value of $s_3 = 0.010(1)$ is at least close to its RS value [49].

Similarly, we have obtained energies e_{k+1} and entropies s_{k+1} for various connectivities up to $k+1 = 25$. In Figure 11.8 we have plotted $e_{k+1}/\sqrt{k+1}$ as a function of $1/(k+1)$. On this scale, we notice that the extrapolated energies split into a set of even and a set of odd values, each located apparently on a straight line. Even though $k+1 \leq 25$ is quite small, each line separately extrapolates for large $k+1$ very close to the exact value of $E_{SK} = -0.7633$ for the SK-model [44]: $E_{SK}^{\text{even}} \approx -0.763$ and $E_{SK}^{\text{odd}} \approx -0.765$. Even more amazing, the value of $e_2 = -1$ for the trivial $k+1 = 2$ Bethe lattice is very close to the linear fit for the even EO results. Clearly, a function that would interpolate continuously *all* the data would have to be very complicated (oscillatory). But could it be that its envelope on the even and the odd integers happens to be simple? Then, in the case of the even data[2], we could even write down the exact form of the function for $\mathcal{E}_{k+1}$ that would fit this data, since we know it also has to pass $e_2 = -1$ *and* E_{SK}:

$$\mathcal{E}_{k+1} = \sqrt{k+1}E_{SK} - \frac{2E_{SK} + \sqrt{2}}{\sqrt{k+1}}. \tag{11.11}$$

We find that all data points for even $k+1$ in fact fall within 0.2% of this line.

11.3.2 Applications of EO by Others

The generality of the EO method beyond the domain of spin-glass problems has recently been demonstrated by Meshoul and Batouche [41] who used the EO algorithm as described above successfully on a standard cost function for aligning natural images. Figure 11.10

[2] Although the odd data may equally well be fitted in this way, the line cannot be determined since only one point on it, E_{SK}, is exactly known.

demonstrates the results of their implementation of τ-EO for this pattern recognition problem. Here, τ-EO finds an optimal affine transformation between a target image and its reference image using a set of n *adjustable* reference points which try to attach to characteristic features of the target image.

The crucial role played by EO's *non-equilibrium fluctuations* in the local search is demonstrated in Figure 11.11. These fluctuations are amazingly similar to those we have found in Figure 11.2. As our discussion in Section 11.2.2 suggests, they are one of the key distinguishing features of EO, and are especially relevant for optimizing highly disordered systems. For instance, Dall and Sibani [21] have observed a significantly broader distribution of states visited – and thus, better solutions found – by τ-EO compared to simulated annealing [36] when applied to the Gaussian spin-glass problem. Blair and Gould [8] found that a hybrid heuristic, mixing annealing with EO, exploits EO's large fluctuations to provide a considerable speed-up over pure annealing in unjamming the segregation process of binary mixtures.

11.3.3 Large-scale Simulations of Spin Glasses

More recently, we have combined EO with reduction methods for sparse graphs [13]. These reductions strip graphs of all low-connected variables ($\alpha \leq 3$), thereby eliminating many entropic barriers that tend to bog down local searches [55]. Along the way, the rules allow for an accounting of the *exact* ground-state energy and entropy, and even of the approximate overlap distribution [12]. The "remainder" graph is subsequently handled efficiently with EO. With such a meta-heuristic approach, for example, we have been able to determine the defect energy distribution [23] for $d = 3$ and 4 dimensional spin glasses, bond-diluted to just above their percolation point, with great accuracy for lattices up to $L = 20$ [13]. As a result, we reduced the error on the stiffness exponent, $y_{d=3} = 0.240(5)$, from 20% [51] to about 2%.

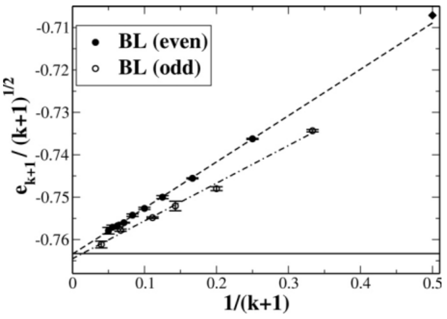


Figure 11.8: Asymptotic plot of the rescaled energies, obtained for *each* connectivity as in Figure 11.6, for Bethe lattices up to $k+1 = 25$. The data falls on two separate lines for even and odd $k + 1$, *including* the trivial result, $e_2 = -1$ (diamond). Fits (dashed) provide an reasonable estimate toward E_{SK} (horizontal) at $k \to \infty$.

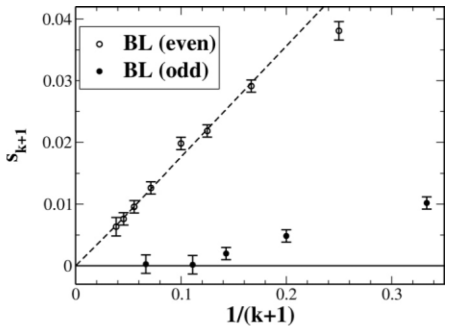


Figure 11.9: Asymptotic plot of all entropies, *each* obtained as in Figure 11.7, as a function of $1/(k + 1)$ for $k + 1$-connected Bethe lattices, $k+1 \leq 25$. The entropies vanish linearly with $1/(k+1)$ for $k+1$ even (dashed), but approach zero (horizontal) more precipitously for odd values.

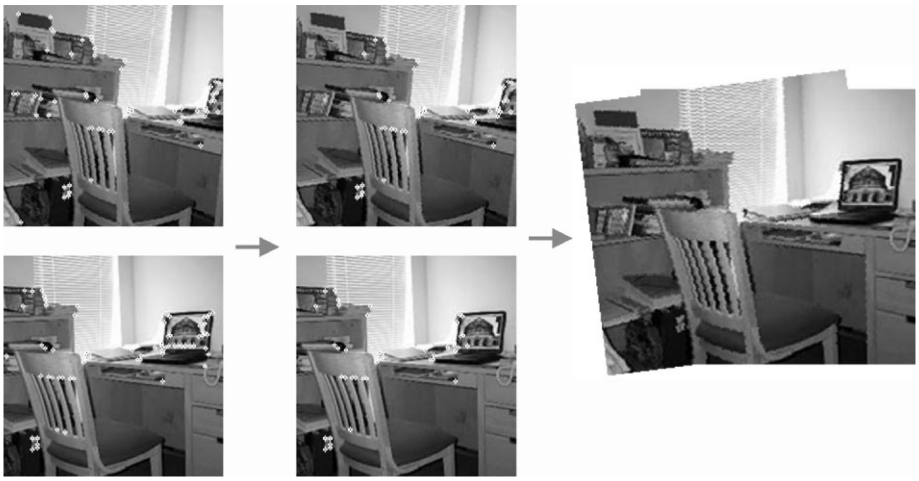


Figure 11.10: Application of EO to the image-matching problem, after [41]. Two different images of the same scene (top row and bottom row) are characterized by a set of n points assigned by a standard pattern recognition algorithm. Starting from an initial assignment (left, top and bottom), the points are updated according to EO, see also Figure 11.11, leading to an optimal assignment (center, top and bottom). This optimal assignment minimizes a cost function for the affine transformation, facilitating an automated alignment of the two images (right). Note that the points move to the part of the scene for which both images overlap. Special thanks to M. Batouche for providing those images.

Currently, we are using this meta-heuristic to explore the (possible) onset of replica symmetry breaking (RSB) for sparse mean-field and lattice models just above percolation. So far, we have only some preliminary data for the Viana–Bray model [59]. In the Viana–Bray model at connectivities near percolation $\alpha \approx \alpha_p = 1$, many spins may be entirely unconnected while a finite fraction is sufficiently connected to form a "giant component" in which interconnected spins may become overconstrained. There the reduction rules allow us to *reduce completely* a statistically significant number of graphs with up to $n = 2^{18}$ spins even well above α_p, since even higher-connected spins may become reducible eventually after totally reducible substructures (trees, loops, etc) emanating from them have been eliminated. At the highest connectivities reached, even graphs originally of $n = 2^{18}$ had collapsed to, at most, 100 irreducible spins, which our EO-algorithm easily optimized.

As a result, we have measured the cost of ground states, Eq. (11.7), as a function of connectivity α on 40 000 instances for each size $n = 2^8, 2^9, \ldots, 2^{14}$, and 400 instances for $n = 2^{15}, \ldots, 2^{18}$, at each of 20 different connectivities α as shown in Figure 11.12. We also account *exactly* for the degeneracy of each instance, which could number up to $\exp[0.3 \times 2^{18}]$; minuscule compared to all $2^{2^{18}}$ configurations! Not entirely reduced graphs had their entropy determined with EO in our meta-heuristic. Consistent with theory [37], Figure 11.12 shows that the entropy per spin follows $s \approx (1 - \alpha/2) \ln 2$ for $\alpha < \alpha_{\text{crit}} = 1$, then continues smoothly through the transition but deviates from that line for $\alpha > \alpha_{\text{crit}}$. Similar data for the overlap-moments [12] may help to determine the onset of RSB expected for this model.

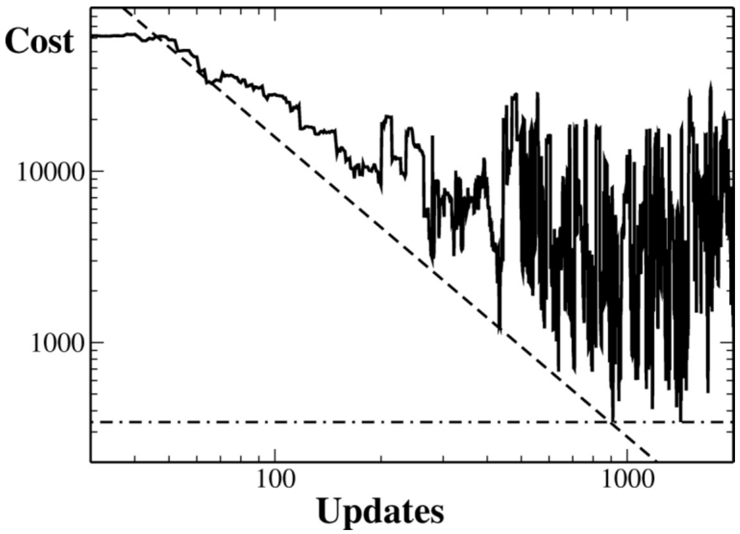
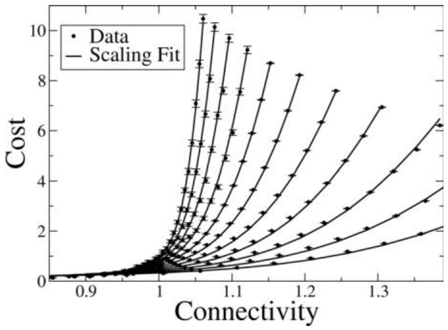


Figure 11.11: Evolution of the cost function in a single EO run for the image alignment problem, with permission from [41]. The inset shows the data on a log-log scale, emphasizing a power-law descent (dashed) toward increasingly better solutions, until saturation is reached (horizontal) after $\approx 10^3$ updates. This should be compared with Figure 11.2, 11.4, and Eq. (11.10). Despite the rather applied nature of this problem, the characteristics of EO prove to be robust.



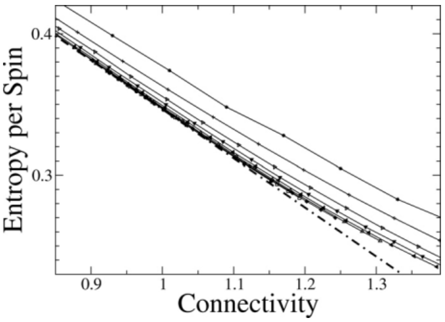


Figure 11.12: Plot (left) of the cost and (right) of the entropy per spin, as a function of connectivity α for random graphs of size $n = 2^8, 2^9, \ldots, 2^{18}$. For increasing n, the cost approaches a singularity at $\alpha_{\text{crit}} = 1.003(9)$, as determined from a finite-size scaling fit (lines on left) to $\langle C\rangle(\alpha, n) \sim n^\delta f\left[(\alpha - \alpha_{\text{crit}})\, n^{1/\nu}\right]$. The fit predicts also $\delta = 0.11(2)$ and $\nu = 3.0(1)$. The entropy density for increasing n drops toward $\approx (1 - \alpha/2)\ln 2$ (dashed line) for $\alpha < \alpha_{\text{crit}} = 1$ [37], continues unaffected through the transition, but deviates from that line for $\alpha > \alpha_{\text{crit}}$ even for large n.

11.4 Theoretical Investigations

Despite the general difficulty in predicting performance features for stochastic optimization methods [1, 38, 61], we are able to theoretically extract a few non-trivial properties of τ-EO [14]. To analyze the properties of the EO-update process, we have to access the fitness of individual variables. To this end, we have proposed an annealed model [14] consisting of n *a priori* independent variables x_i, taking on one of, say, $\alpha + 1 = 3$ fitness states, $\lambda_i = 0$, -1, or -2. At each point in time, respective fractions ρ_0, ρ_1, and ρ_2 of the variables occupy these states, where $\sum_a \rho_a = 1$. The optimal configuration is $\rho_0 = 1$, $\rho_{1,2} = 0$ with a cost per variable of $C = -\sum_i \lambda_i/n = \sum_{a=0}^{2} a\rho_a = 0$, according to Eqs. (11.7)–(11.8). With this system, we can model the dynamics of a local search for hard problems by "designing" an interesting set of flow equations for $\rho(t)$ that can mimic a complex search space through energetic or entropic barriers. In these flow equations, a transition matrix T_{ab} specifies what fraction of variables transitions in or out of a fitness state (a), *given* that a variable in a certain state (b) is updated. (This transition of a is conditioned by b, not necessarily between a and b!) The probabilities that a variable in ρ_b is updated, Q_b, can be derived *exactly* for local search, typically giving a highly non-linear dynamic system:

$$\dot{\rho}_a = \sum_b T_{ab} Q_b. \tag{11.12}$$

For example, for τ-EO the vector $\mathbf{Q}$ depends exclusively on ρ. Since for each update a variable is selected based on its rank according to Eq. (11.5). When a rank $k(\leq n)$ has been chosen, a spin is randomly picked from state $0 \leq a \leq \alpha (= 2$ here), if $k/n \leq \rho_\alpha$, from state $\alpha - 1$, if $\rho_\alpha < k/n \leq \rho_\alpha + \rho_{\alpha-1}$, and so on. We introduce a new, continuous variable $x = k/n$, approximate sums by integrals, and rewrite $P(k)$ in Eq. (11.5) as

$$p(x) = \frac{\tau - 1}{n^{\tau-1} - 1} x^{-\tau} \quad \left(\frac{1}{n} \leq x \leq 1\right), \tag{11.13}$$

where the maintenance of the low-x cut-off at $1/n$ will turn out to be crucial. Now, the average likelihood that a spin in a given state is updated is given by

$$\begin{aligned}
Q_\alpha &= \int_{1/n}^{\rho_\alpha} p(x)dx = \frac{1}{1 - n^{\tau-1}} \left(\rho_\alpha^{1-\tau} - n^{\tau-1}\right), \\
Q_{\alpha-1} &= \int_{\rho_\alpha}^{\rho_\alpha+\rho_{\alpha-1}} p(x)dx = \frac{1}{1 - n^{\tau-1}} \left[(\rho_{\alpha-1} + \rho_\alpha)^{1-\tau} - \rho_\alpha^{1-\tau}\right], \\
&\dots \\
Q_0 &= \int_{1-\rho_0}^{1} p(x)dx = \frac{1}{1 - n^{\tau-1}} \left[1 - (1 - \rho_0)^{1-\tau}\right],
\end{aligned} \tag{11.14}$$

where in the last line the norm $\sum_i \rho_i = 1$ was used in both integration limits. These values of the Q's completely describe the update preferences for τ-EO at arbitrary τ. In the case $\alpha = 2$, Eq. (11.15) gives

$$Q_0 = \frac{1 - (1 - \rho_0)^{1-\tau}}{1 - n^{\tau-1}}, \; Q_1 = \frac{(\rho_1 + \rho_2)^{1-\tau} - \rho_2^{1-\tau}}{1 - n^{\tau-1}}, \; Q_2 = 1 - Q_0 - Q_1. \tag{11.15}$$

For SA (with temperature schedule $\beta = 1/T = \beta(t)$) Metropolis-updates require [14]

$$Q_a \propto \rho_a \min\left\{1, \exp\left[-\beta \sum_{b=0}^{\alpha} b T_{ba}\right]\right\}, \quad (a = 0, 1, \ldots, \alpha). \tag{11.16}$$

Thus, with the *choice* of a specific model $\mathbf{T}$, we could study any (dynamic or stationary) property of τ-EO as a function of τ and compare it to SA.

To demonstrate the use of these equations, we consider a (trivial) model with a constant matrix describing the transition of fractions of variables, $T_{ab} = [-\delta_{ab} + \delta_{(2+a \bmod 3),b}]/n$, depicted on the left in Figure 11.13. Here, variables in ρ_1 can only reach the lowest-energy state in ρ_0 by first jumping *up* in energy to ρ_2. Eq. (11.12) gives

$$\dot{\rho}_0 = \frac{1}{n}\left(-Q_0 + Q_2\right), \quad \dot{\rho}_1 = \frac{1}{n}\left(Q_0 - Q_1\right), \quad \dot{\rho}_2 = \frac{1}{n}\left(Q_1 - Q_2\right), \tag{11.17}$$

with $\mathbf{Q}$ in Eq. (11.15) for EO and for SA with

$$Q_0 = \frac{\rho_0 e^{-\beta}}{(1-e^{-\beta})\rho_2 + e^{-\beta}}, \; Q_1 = \frac{\rho_1 e^{-\beta}}{(1-e^{-\beta})\rho_2 + e^{-\beta}}, \; Q_2 = 1 - Q_0 - Q_1, \tag{11.18}$$

The stationary solution, for $\dot{\rho} = 0$, yields $Q_0 = Q_1 = Q_2$, and

$$\rho_0 = 1 - \left(\frac{n^{\tau-1}+2}{3}\right)^{\frac{1}{1-\tau}}, \; \rho_2 = \left(\frac{2n^{\tau-1}+1}{3}\right)^{\frac{1}{1-\tau}}, \; \rho_1 = 1 - \rho_0 - \rho_2, \tag{11.19}$$

for EO, and for SA

$$\rho_0 = \frac{1}{2+e^{-\beta}}, \quad \rho_1 = \frac{1}{2+e^{-\beta}}, \quad \rho_2 = \frac{e^{-\beta}}{2+e^{-\beta}}. \tag{11.20}$$

Therefore, SA reaches its best, albeit suboptimal, cost $C = 1/2 > 0$ at $\beta \to \infty$, due to the energetic barrier faced by the variables in ρ_1. The result for EO is most remarkable [14, 24]: For $n \to \infty$ at $\tau < 1$ EO remains suboptimal, but reaches the optimal cost *for all* $\tau > 1$! This transition at $\tau = 1$ separates an (ergodic) random walk phase with too much fluctuation, and a greedy-descent phase with too little fluctuation, which in real NP-hard problems would probably produce broken ergodicity [6]. This "ergodicity breaking" derives from the scale-free power-law in Eq. (11.5), as argued in Section 11.2.3 and Ref. [16].

Naturally, the range of phenomena found in a local search of NP-hard problems is not limited to energetic barriers. After all, so far we have only considered constant entries for $T_{i,j}$. In our next model we let $\mathbf{T}$ merely depend linearly on the ρ_i. Most of these cases reduce to the phenomena already discussed in the previous example. An entirely new effect arises in the following case, depicted on the right in Figure 11.13:

$$\dot{\rho}_0 = \frac{1}{n}\left[-Q_0 + \frac{1}{2}Q_1\right], \qquad \dot{\rho}_1 = \frac{1}{n}\left[\frac{1}{2}Q_0 - Q_1 + (\theta - \rho_1)Q_2\right], \tag{11.21}$$

where $\dot{\rho}_2 = -\dot{\rho}_0 - \dot{\rho}_1$, since $\rho_0 + \rho_1 + \rho_2 = 1$. Aside from the dependence of $\mathbf{T}$ on ρ_1, we have also introduced the threshold parameter θ. The interesting regime is the case $0 < \theta < 1$, where further flow from state 2 into state 1 can be blocked for increasing ρ_1, providing a negative feedback to the system. In effect, the model is capable of exhibiting a "jam" as observed in many models of glassy dynamics, and which is certainly an aspect of local search processes.

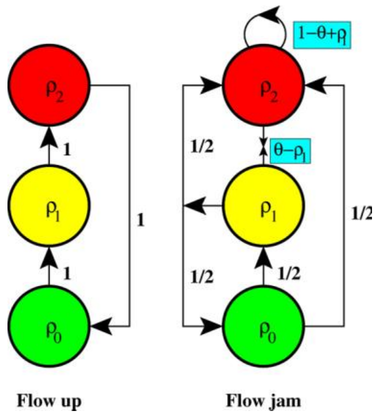


Figure 11.13: Plot of flow diagrams. In the diagram on the left, variables have to jump to higher energetic states first before they can attain the lowest state. The right diagram shows the model of a jam, where variables in the highest state can only traverse through the intermediate state to the lowest state, if the intermediate state moves its variables out of the way first to keep its density ρ_1 below the threshold θ.

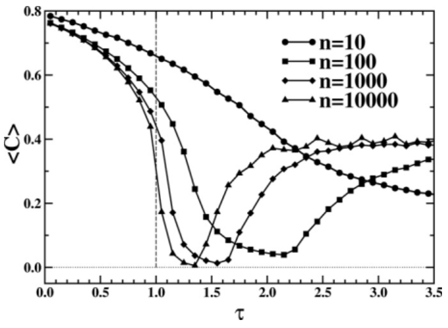


Figure 11.14: Plot of the energy $\langle C \rangle$ averaged over many τ-EO runs with different initial conditions as a function of τ for $n = 10$, 100, 1000, and 10 000 and $\theta = 1/2$. For small values of τ, $\langle C \rangle$ closely follows the steady-state solution to Eq. (11.21). It reaches a minimum at a value the moves inward slowly, as predicted for τ_{opt} by Eq. (11.6), and rises sharply beyond that. The phase transition at $\tau = 1$ is apparent.

Equations (11.21) again have a unique fixed-point solution with $\tau = \infty$ being the most favorable value at which the minimal energy $C = 0$ is definitely reached. But it can be shown that the system has an ever harder time to reach that point, requiring typically $t = O(n^\tau)$ update steps for a finite fraction of initial conditions. Thus, for a given finite computational time $t_{\max}$ the best results are obtained at some finite value of τ_{opt}. In that, this model provides a new feature – slow variables impeding the dynamics of faster ones [52] – resembling the observed behavior for EO on real problems, e.g. the effect shown in Figure 11.1. In particular, this model provides an analytically tractable picture for the relation between the value of τ_{opt} and the effective loss of ergodicity in the search conjectured in Refs. [16, 18].

For initial conditions that lead to a jam, $\rho_1(0) + \rho_2(0) > \theta$, we assume that

$$\rho_1(t) = \theta - \epsilon(t) \tag{11.22}$$

with $\epsilon \ll 1$ for $t \lesssim t_{\text{jam}}$, where t_{jam} is the time at which ρ_2 becomes small and Eq. (11.22) *fails*. To determine t_{jam}, we apply Eq. (11.22) to the evolution equations in (11.21) and obtain after some calculation [14]

$$t_{\text{jam}} \sim n^\tau, \tag{11.23}$$

Further analysis shows that the average cost $\langle C \rangle(\tau)$ develops a minimum as soon as $t_{\max} \geq n$ and when $t_{\max} \sim t_{\text{jam}}$, so choosing $t_{\max} = an$ leads directly to Eq. (11.6). This sequence of minima in $\langle C \rangle(\tau)$ for increasing n is confirmed by the numerical simulations (with $a = 100$) shown in Figure 11.14, with the correct n-dependence predicted by Eq. (11.6).

References

[1] E. H. L. Aarts and P. J. M. van Laarhoven, *Simulated Annealing: Theory and Applications* (Reidel, Dordrecht, 1987).

[2] P. Bak, *How Nature Works* (Springer, New York, 1996).

[3] P. Bak and K. Sneppen, *Punctuated Equilibrium and Criticality in a simple Model of Evolution,* Phys. Rev. Lett. **71**, 4083 (1993).

[4] P. Bak, C. Tang, and K. Wiesenfeld, *Self-organized Criticality,* Phys. Rev. Lett. **59**, 381 (1987).

[5] J. R. Banavar, D Sherrington, and N. Sourlas, *Graph Bipartitioning and Statistical Mechanics,* J. Phys. A: Math. Gen. **20**, L1 (1987).

[6] F. T. Bantilan and R. G. Palmer, *Magnetic Properties of a Model Spin Glass and the Failure of Linear Response Theory,* J. Phys. F: Metal Phys. **11**, 261 (1981).

[7] F. Barahona, *On the Computational Complexity of Ising Spin Glass Models,* J. Phys. A.: Math. Gen. **15**, 3241-3253 (1982).

[8] D. L. Blair and H. Gould (Clark University), private communication.

[9] S. Boettcher, *Extremal Optimization and Graph Partitioning at the Percolation Threshold,* J. Math. Phys. A. **32**, 5201 (1999).

[10] S. Boettcher, *Numerical Results for Ground States of Mean-Field Spin Glasses at low Connectivities,* Phys. Rev. B **67**, R060403 (2003).

[11] S. Boettcher, *Numerical Results for Ground States of Spin Glasses on Bethe Lattices,* Eur. Phys. J. B **31**, 29 (2003).

[12] S. Boettcher, *Reduction of Spin Glasses applied to the Migdal-Kadanoff Hierarchical Lattice,* Eur. Phys. J. B **33**, 439 (2003).

[13] S. Boettcher, *Determining Stiffness Exponents from the Reduction of Dilute Lattice Spin Glasses,* cond-mat/0303431.

[14] S. Boettcher and M. Grigni, *Jamming Model for the Extremal Optimization Heuristic,* J. Phys. A. **35**, 1109 (2002).

[15] S. Boettcher and M. Paczuski, *Exact Results for Spatiotemporal Correlations in a Self-organized Critical Model of Punctuated Equilibrium,* Phys. Rev. Lett. **76**, 348 (1996).

[16] S. Boettcher and A. G. Percus, *Nature's Way of Optimizing,* Artificial Intelligence **119**, 275 (2000).

[17] S. Boettcher and A. G. Percus, *Optimization with Extremal Dynamics,* Phys. Rev. Lett. **86**, 5211 (2001).

[18] S. Boettcher and A. G. Percus, *Extremal Optimization for Graph Partitioning,* Phys. Rev. E **64**, 026114 (2001).

[19] M. Cieplak, A. Giacometti, A. Maritan, A. Rinaldo, I. Rodriguez-Iturbe, and J. R. Banavar, *Models of Fractal River Basins,* J. Stat. Phys. **91**, 1 (1998).

[20] J. Dall, *Searching Complex State Spaces with Extremal Optimization and other Stochastic Techniques,* Master Thesis, Fysisk Institut, Syddansk Universitet Odense, 2000 (in Danish).

[21] J. Dall and P. Sibani, *Faster Monte Carlo Simulations at Low Temperatures: The Waiting Time Method,* Computer Physics Communication **141**, 260 (2001).

[22] S. F. Edwards and P. W. Anderson, *Theory of Spin Glasses,* J. Phys. F: Metal Phys. **5**, 965 (1975).

[23] K. H. Fischer and J. A. Hertz, *Spin Glasses* (Cambridge University Press, Cambridge, 1991).

[24] M. Frank, *Analysis of Extremal Optimization in Designed Search Spaces,* Honors Thesis, Dept. of Physics, Emory University, (in preparation).

[25] F. Glover, *Future Paths for Integer Programming and Links to Artificial Intelligence,* Computers & Ops. Res. **5**, 533 (1986).

[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* (Addison-Wesley, Reading, 1989).

[27] S. J. Gould and N. Eldridge, *Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered,* Paleobiology **3**, 115 (1977).

[28] J. W. Greene and K. J. Supowit, *Simulated Annealing without Rejecting Moves,* IEEE Trans. on Computer-Aided Design **CAD-5**, 221 (1986).

[29] A. K. Hartmann, *Ground-state Clusters of two-, three- and four-dimensional $\pm JJ$ Ising Spin Glasses,* Phys. Rev. E **63**, 016106 (2001).

[30] A. K. Hartmann, *Evidence for Existence of Many Pure Ground States in 3d $\pm J$ Spin Glasses,* Europhys. Lett. **40**, 429 (1997).

[31] A. K. Hartmann, *Calculation of Ground-state Behavior Of four-dimensional $\pm J$ Ising Spin Glasses,* Phys. Rev. E **60**, 5135 (1999).

[32] A. K. Hartmann and H. Rieger, *Optimization Algorithms in Physics* (Wiley-VCH, Berlin, 2001).

[33] *Frontiers in problem solving: Phase Transitions and Complexity,* eds. T. Hogg, B. A. Huberman, and C. Williams, special issue of Artificial Intelligence **81** (1996).

[34] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, *Optimization by Simulated Annealing - an Experimental Evaluation. 1. Graph Partitioning,* Operations Research **37**, 865 (1989).

[35] G. Karypis and V. Kumar, *METIS, a Software Package for Partitioning Unstructured Graphs,* see www-users.cs.umn.edu/˜karypis/metis/index.html.

[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing,* Science **220**, 671 (1983).

[37] M. Leone, F. Ricci-Tersenghi, and R. Zecchina, *Phase Coexistence and Finite-size Scaling in Random Combinatorial Problems,* J. Phys. A.: Math. Gen. **34**, 4615 (2001).

[38] M. Lundy and A. Mees, *Convergence of an Annealing Algorithm,* Math. Programming **34**, 111 (1986).

[39] M. B. Menai and M. Batouche, *Approximate Solution of Max-SAT Problem using Extremal Optimization Heuristic,* Journal of Automated Reasoning, (to appear).

[40] P. Merz and B. Freisleben, *Memetic Algorithms and the Fitness Landscape of the Graph Bi-partitioning Problem,* Lect. Notes Comput. Sc. **1498**, 765 (1998).

[41] S. Meshoul and M. Batouche, *Robust Point Correspondence for Image Registration using Optimization with Extremal Dynamics,* Lect. Notes Comput. Sc. **2449**, 330 (2002).

[42] M. Mézard and G. Parisi, *Mean-field Theory of Randomly Frustrated Systems with Finite Connectivity,* Europhys. Lett. **3**, 1067 (1987).

[43] M. Mézard and G. Parisi, *Cavity Method at Zero Temperature*, J. Stat. Phys **111**, 1 (2003).

[44] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).

[45] M. Mézard, G. Parisi, and R. Zecchina, *Analytic and Algorithmic Solution of Random Satisfiability Problems,* Science **297**, 812 (2002).

[46] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Determining Computational Complexity from Characteristic 'phase Transitions,'* Nature **400**, 133 (1999).

[47] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, *Coloring Random Graphs,* Phys. Rev. Lett. **89**, 268701 (2002).

[48] M. Paczuski, S. Maslov, and P. Bak, *Avalanche Dynamics in Evolution, Growth, and Depinning Models,* Phys. Rev. E **53**, 414 (1996).

[49] A. Pagnani, G. Parisi, and M. Ratieville, *Metastable Configurations on the Bethe Lattice,* Phys. Rev. E **67**, 026116 (2003).

[50] K. F. Pal, *The Ground-state Energy of the Edwards–Anderson Ising Spin Glass with a Hybrid Genetic Algorithm,* Physica A **223**, 283 (1996).

[51] M. Palassini and A. P. Young, *Triviality of the Ground State Structure in Ising Spin Glasses,* Phys. Rev. Lett. **83**, 5126 (1999).

[52] R. G. Palmer, D. L. Stein, E. Abrahams, and P. W. Anderson, *Models of Hierarchically Constrained Dynamics for Glassy Relaxation,* Phys. Rev. Lett. **53**, 958 (1984).

[53] A. G. Percus and O. C. Martin, *Finite Size and Dimensional Dependence of the Euclidean Traveling Salesman Problem,* Physical Review Letters **76**, 1188 (1996).

[54] *Modern Heuristic Techniques for Combinatorial Problems,* Ed. C. R. Reeves (Wiley, New York, 1993).

[55] F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, *Simplest Random K-Satisfiability Problem,* Phys. Rev. E **63**, 026702 (2001).

[56] D. Sherrington and S. Kirkpatrick, *Solvable Model of a Spin Glass,* Phys. Rev. Lett. **35**, 1792 (1975).

[57] E. Somfai, A. Czirok, and T. Vicsek, *Power-Law Distribution of Landslides in an Experiment on the Erosion of a Granular Pile,* J. Phys. A. **27**, L757 (1994).

[58] Thanks to A. Erzan for exploring EO for polymer folding, based on the model in E. Tuzel and A. Erzan, *Glassy Dynamics of Protein Folding,* Phys. Rev. E **61**, R1040 (2000).

[59] L. Viana and A. J. Bray, *Phase Diagrams for Dilute Spin Glasses,* J. Phys. C **18**, 3037 (1985).

[60] J.-S. Wang and Y. Okabe, *A Comparison of Extremal Optimization with Flat-histogram Dynamics for Finding Spin-glass Ground States,* J. Phys. Soc. Jpn. **72**, 1380 (2003).

[61] I. Wegener, *Theoretical Aspects of Evolutionary Algorithms,* Lect. Notes Comput. Sc. **2076**, 64 (2001).

[62] K. Y. M. Wong and D. Sherrington, *Graph Bipartitioning and Spin-Glasses on a Random Network of Fixed Finite Valence,* J. Phys. A: Math. Gen. **20**, L793 (1987).

[63] E. Yom-Tov, A. Grossman, and G. F. Inbar, *Movement-related Potentials During the Performance of a Motor Task I: The Effect of Learning and Force,* Bio. Cybernetics **85**, 395 (2001).

12 Sequence Alignments

Alexander K. Hartmann

Many computational methods used in molecular biology or bioinformatics are in fact optimization algorithms. An example is the determination of the structure of proteins, see, e.g., the contribution by U.H.E. Hansmann in Chapter 13. Here, we are dealing with *sequence alignments*, which is a method of comparing different biological sequences such as genes or proteins. We will start with a short introduction on the basic notions in molecular biology, which are necessary in order to understand this part. Next, sequence alignments are defined and different alignment algorithms are explained. In the final part, as an application, the tail of the distribution of alignment scores for random protein sequences is studied. Here, additional methods from statistical physics are employed in order to study the distribution in the part where sequences with probabilities as small as 10^{-40} occur.

12.1 Molecular Biology

Proteins play a dominant role in life [9]. Many of them act as enzymes, which means that they catalyze various chemical reactions in cells. Muscle contraction is due to protein–protein interactions. Proteins act as channels to control the transport of substances through cell membranes. As a final example, proteins play a crucial role in the immune system, see, e.g., Figure 12.1.

Chemically, proteins are polypeptide chains. They consist of a number of amino acids linked to each other. Single-chain molecules are very common, but there are also multiple-chain proteins. Typical lengths are several hundred amino acids, the longest contain about 5000 amino acids. 20 different amino acid molecules form the building blocks of proteins. Each amino acid consists of a carbon atom C to which a hydrogen atom H, an amino group (NH_2), a carboxyl group (COOH), and an amino-acid dependent group R are bound. Only one amino-acid, proline, has a slightly different structure, see below. In aqueous neutral solution, the amino acids exist in zwitterionic form, displayed in the Figures 12.2–12.5. Please note that not all amino acids are acids, despite their name. The smallest and simplest amino acid is Glycine, see Figure 12.2, denoted by the letter G. Here, the R group is just a single hydrogen atom. It is neither especially hydrophilic or hydrophobic.

There are eight different *hydrophobic*, i.e., non-polar, amino acids, see Figure 12.3. Amino acids where the group R can be ionized in solution are hydrophilic. The nine hydrophilic amino acids are displayed in Figure 12.4. Two special amino acids, which fall out of the previous classification, are Cysteine (C) and Proline (P), see Figure 12.5.

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

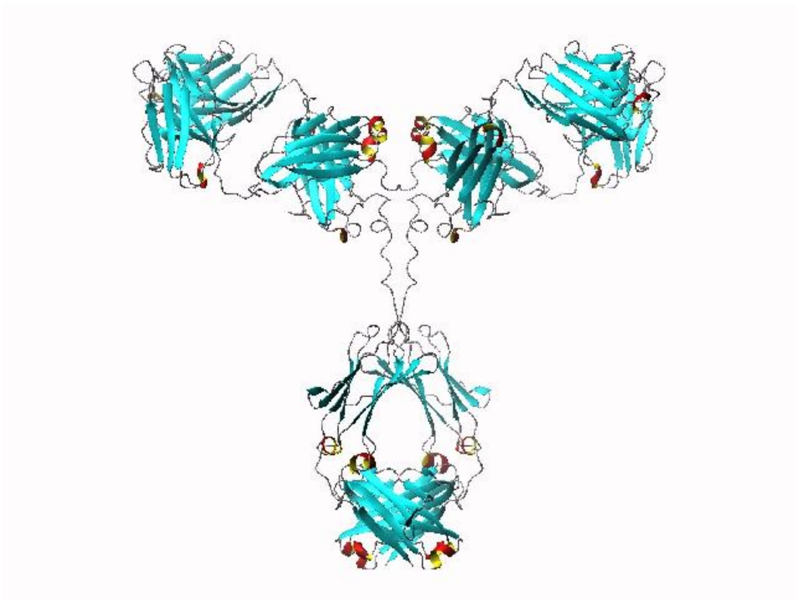


Figure 12.1: The antibody *immunoglobulin G* (IgG), which attaches to antigens, i.e., macromolecules foreign to the body, and then stimulates the immune system to generate a protective response.

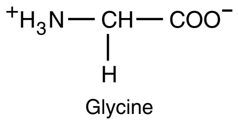


Figure 12.2: Glycine (G) is the smallest amino acid.

In a protein, the different amino acids are connected by *peptide bonds*, which connect, under formal removal of H_2O, the carbon atom of the carboxyl group with the nitrogen atom of the amino group. Since this process exhibits an energy barrier, in a cell usually enzymes, i.e., other proteins, perform the assembly. For the final protein, at one end an amino group (*N-terminal end*) and at the other end a carboxyl group (*C-terminal end*) remain unbounded. Usually one specifies proteins by stating the amino acid sequence starting at the N-terminal end. This sequence of amino acids is called the *primary* structure. It determines the structure of the protein in real-space[1] to a large extent. Important factors in the creation of the real-space structure are the formation of hydrogen bonds and the environment (water and enzymes) of the protein. The structure can be measured using X-ray diffraction or nuclear magnetic resonance

[1] The real-space structure is described in a hierarchical manner, i.e., by its so called *secondary, tertiary and quaternary structure*. Since we are here interested in the primary structure only, we do not go into details.

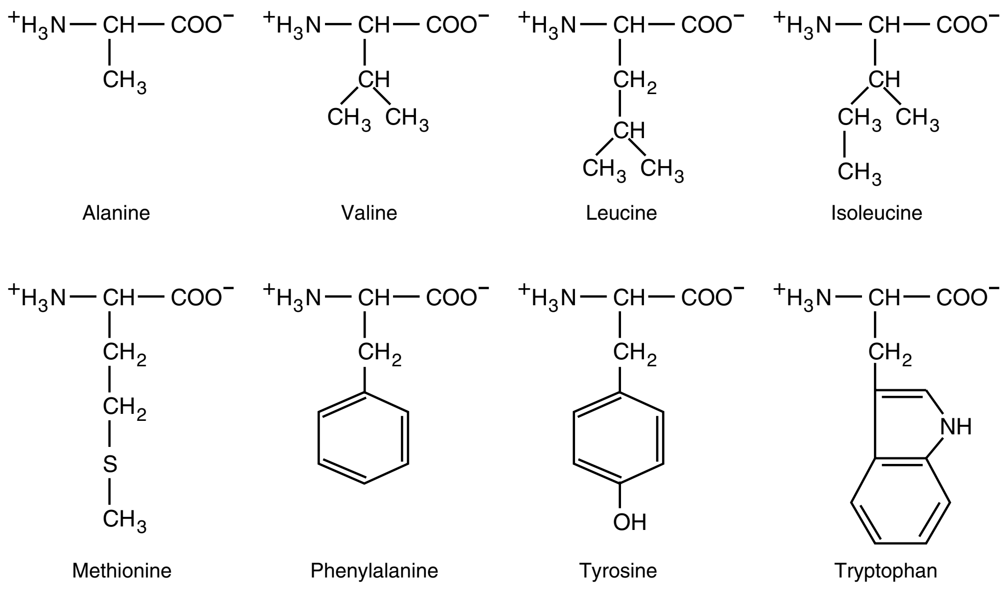


Figure 12.3: Hydrophobic amino acids: Alanine (A), Valine (V), Leucine (L), Isoleucine (I), Methionine (M), Phenylalanine (F), Tyrosine (Y) and Tryptophan (W).

(NMR). The real-space structure is very characteristic for each protein and of crucial importance for its function. For example, mad-cow disease is believed to be due to "missfolded" proteins, called *prions*. Hence, the prediction of the structures of proteins and the study of the folding process is a very important task in molecular biology. A usual approach to this problem is to look for minimum energy conformations, see e.g., Chapter 13.

In order to study proteins, one must know their sequences. Currently, two methods are used to investigate short sequences (up to, let us say, 30 amino acids), *sequenators* and *mass spectrometry*. The sequenator is an automatic device which is based on a chemical method: Using chemical reagents it detaches one amino acid after another, starting from the N-terminal end, and identifies the amino acid in a separating column. As separation column, very often *gel electrophoresis* is used. It works in principle in the following way. Charged molecules, which are marked, either radioactively or with a dye, are put in a gel and are subjected to an electric field. The molecules are driven by the field, the faster they move the higher is their charge, but the larger they are, the slower they travel. After calibration with known amino acids, the identity of an unknown amino acid can be inferred just from the final distance it has traveled in the field (if the amount measured is large enough).

When using mass spectroscopy, one needs a collection of many fragments from the protein. Since all fragments originate from the same protein, only certain combinations of amino acids can appear as fragments, i.e., all possible subsequences of the full sequence. Each subsequence has a certain mass, which is the sum the amino acid masses it contains. Hence, for a given protein, only a finite number of values for the masses of the fragments can appear. This means that the sequence of the given protein can be deduced from the set of feasible values of fragment masses, i.e., from the positions of the peaks in the spectrum.

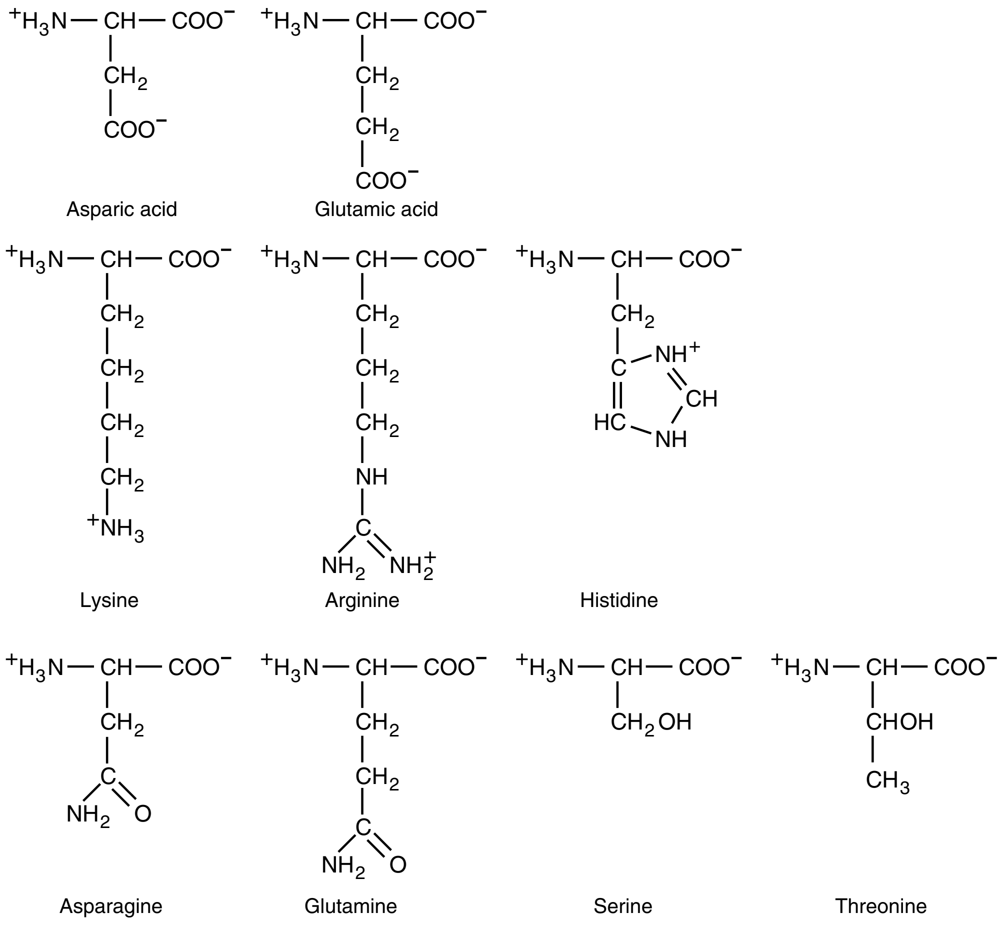


Figure 12.4: Hydrophilic amino acids: Aspartic acid (D), Glutamic acid (E), Lysine (K), Arginine (R), Histidine (H), Asparagine (N), Glutamine (Q), Serine (S) and Threonine (T).

For longer proteins, one has to break them into smaller fragments using enzymes breaking at specific positions. Then one analyzes the different fragments, and looks for overlaps of the different fragments to obtain the full sequence. The method of finding the overlapping regions is to utilize the sequence alignment algorithms, which are explained in the next section. As we will see later, sequence alignment can also be used for general protein comparison.

The primary structures of all proteins of a cell are stored in the *genome*, which is a collection of DNA (deoxyribonucleic acid) chains. They are used to generate all proteins in a cell, one can say, the proteins currently produced are *expressed* by the cell. DNA is a double-stranded helix, where each strand consists of a chain of nucleotides. Each nucleotide has the principle form shown in Figure 12.6. It consists of a sugar (2-deoxy-D-ribose) with an additional phosphate group and a base bound at position $1'$ of the sugar ring. The type of

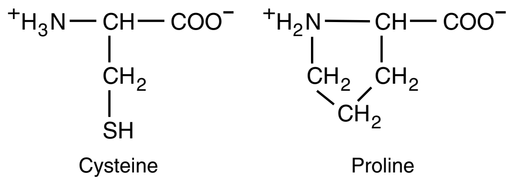


Figure 12.5: Cysteine (C) and Proline (P) are proteins for special purposes. Cystine is important for providing sulphur in its -SH group. Thus, in proteins containing Cystine, S–S bonds can be formed. They are very strong and lead to stable structures. Hence Cystine plays an important role for the protein folding process (see Chapter 13). Proline differs from the general structure of the 19 other amino acids. It has an imino (-NH) group instead of an amino (-NH_2) group, i.e., it should actually be called an imino acid.

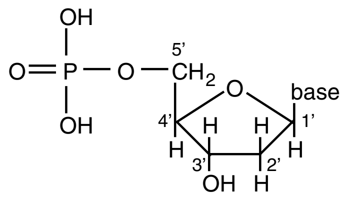


Figure 12.6: General structure for nucleotide.

nucleotide is specified by the base. For DNA, four different base types exist: Adenine (A), Guanine (G), Cytosine (C) and Thymine (T), see Figure 12.7. Adenine and Guanine are bound to the sugar at the 9 position, and Cytosine and Thymine at the 1 position.

The different nucleotides are connected to each other via formally removing one -OH group from the phosphate group (called $5'$, see Figure 12.6), removing a hydrogen atom at the $3'$ position and connecting the "open" bonds. Please note that this formation does not take place directly, because it involves a high free-energy barrier to overcome. In a cell, again, enzymes catalyze this process. The ends of the DNA chains are uniquely defined, because at one end, called $5'$, there is an phosphate group, while at the other, called $3'$ there is none. In a cell, DNA is processed always starting at a $5'$ end.

Since there are 20 different amino acids, 3 bases ($4^3 = 64$ combinations), are needed to code an amino acid in principle. A group of three bases is called *codon*. Most amino acids are coded by several codons. The codon ATG indicates the start of a coding sequence, (but it codes Methionine as well). The sequence of codons coding a protein is called a *gene*. The three codons TAA, TAG, TGA, (called *stop codons*) are used to indicate the end of a gene. This code is almost universal for all organisms, there are only a few minor variations. But there is not always a one-to-one correspondence between DNA and proteins, sometimes details of a protein are determined at assembly. Also, large parts of the genome do not code protein sequences, and not everything of this "non-coding" region is currently well understood.

The double-helix structure is held together by hydrogen bonds. Hydrogen bonds can be formed only between A–T pairs (2 bonds) and between G–C pairs (3 bonds). This means

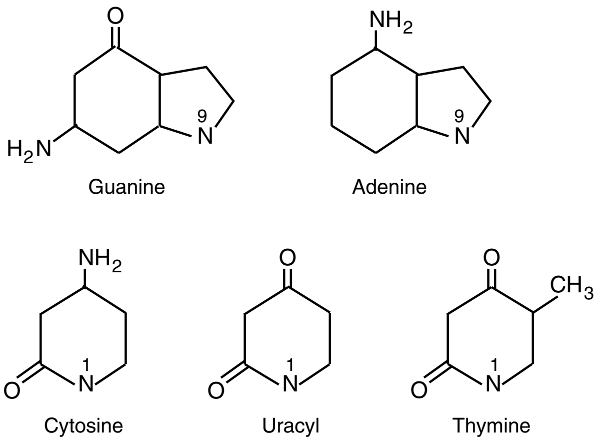


Figure 12.7: The five bases Guanine (G), Adenine (A), Cytosine (C), Uracil (U) and Thymine (T).

that one strand of the DNA already contains the full genomic information, the second one is somehow a "backup" to make the stored information insensitive against external perturbations. Furthermore, it makes duplication of the genes easier. Due to the weak nature of hydrogen bonds, the double-helix structure can be *denaturated in vitro* by heating, i.e., the double helix separates into two single strands. In living cells, the DNA has to be unfolded in order to be read. This is not done by heating but special enzymes are used by the cell. For the common case, where only a part of the DNA has to be read off to synthesize a protein, an enzyme called *RNA polymerase* splits the double strand locally and creates a copy from one strand of the DNA. This copy is called an mRNA (*messenger ribonucleic acid*). The next parts of protein synthesis use the mRNA, hence the valuable original data stored in the DNA has to be touched only rarely.

RNA is similar to DNA, i.e., a chain of nucleotides. There are two main structural differences. First, the sugar part of a nucleotide is a ribose instead of a deoxyribose, i.e., it has an -OH group at position $2'$ instead of a hydrogen (see Figure 12.6). Second, the four bases for RNA are Guanine, Adenine, Cytosine, as for DNA, and Uracil (U) (instead of Thymine for DNA), see Figure 12.7. RNA is single-stranded, but also folds into a three-dimensional structure which is crucial for its function.

In principle, DNA can be analyzed by similar methods to proteins. But one has the additional problem that usually a very small amount of the DNA is available. Thus, one has to amplify the probe. This is done using a *polymerase chain reaction* (PCR). This method iteratively heats the sample up to $95°$, causing the denaturation of the double helix. Then each occurrence of the desired gene section is duplicated using the enzyme RNA polymerase. The enzyme grows starting from *primers*, which attach to subsequences unique for each primer. Then the probe has to be cooled again, to allow further primers to attach to the new created sequences. This cycle can be repeated, each time doubling the amount of DNA.

The actual measurement of the DNA sequence can be performed by applying the *fluorescent method*. It works by adding modified bases called *terminators*, which do not allow the sequence to continue to grow during the duplication process beyond the terminator. These terminators are marked with four different dyes, one color for each base type, i.e., one obtains a collection of subsequences, all starting at the primer position and ending at all possible positions of the initial DNA sequence, with the color at the end of the subsequence indicating the base with which the corresponding subsequence ends. Then the sample is studied using gel electrophoresis (see above). The distance a segment has traveled within a certain time in the electric field decreases with the length of the segment. Therefore, from the different distances, the position where each given base was located can be immediately read off from the color visible at this position. This color detection can be performed automatically, allowing for fully automated DNA analysis. This technique works for relative short DNA, about 500 bases long. In e.g. human cells, the DNA is about 6 billion base pairs long. In this case, similar to the analysis of long proteins, one has to divide the genome into many fragments, analyze each fragment, and look for matching regions of the fragments (*shotgun method*).

For this method, but also for protein and gene comparison in general, sequence alignments are used, which are explained in the next section.

12.2 Alignments and Alignment Algorithms

Modern molecular biology, e.g., the *Human Genome Project* [16], relies heavily on the use of large databases [5, 25], where DNA or protein sequences are stored. The basic tool for accessing these databases and comparing different sequences is *sequence alignment*. The result of each comparison is a *maximum* alignment *score* S, i.e., an optimization algorithm is used to obtain the maximum score. One is interested either in *global* or *local* optimum alignments. For the first case, the score is maximized over all alignments of both complete sequences. The optimum local alignment is the optimum over all global alignments of all possible pairs of contiguous subsequences (for a precise definition see below). Next, a formal definition of alignment is given, then algorithms for several types of alignments are presented.

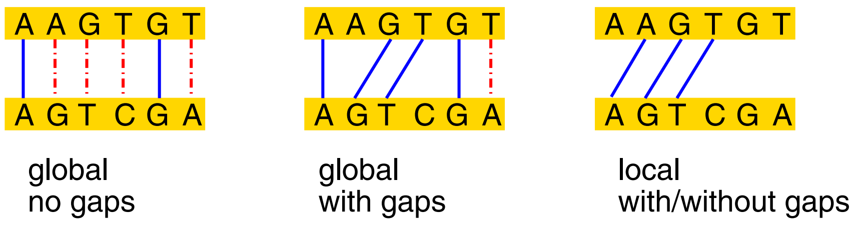


Figure 12.8: Different types of sequence alignments. Two DNA sequences are shown. Aligned symbols are joined by lines. Matching alignments are indicated by full lines, non-matches by broken lines. For global alignment without gaps (left), all letters are aligned. When allowing for gaps (middle), additional costs (i.e. negative scores) are taken into account. For local alignments (with and without gaps), the unpaired regions and the beginning and at the end of the sequences are not penalized, i.e. a zero score is contributed from there.

Let $\mathbf{x} = x_1 x_2 \ldots x_n$ and $\mathbf{y} = y_1 y_2 \ldots y_m$ be two sequences over a finite alphabet Σ with $r = |\Sigma|$ letters. For DNA the alphabet has four letters, representing the bases, for protein sequences it has 20 letters, representing the amino acids. The basic idea of using alignments to compare $\mathbf{x}, \mathbf{y}$ is to identify equal or similar subsequences in $\mathbf{x}$ and $\mathbf{y}$. An alignment is a pairing $\{(x_{i_k}, y_{j_k})\}$ ($k = 1, 2, \ldots, K$, $1 \leq i_k < i_{k+1} \leq n$ and $1 \leq j_k < j_{k+1} \leq m$) of letters from the two sequences. Some letters may not be aligned in principle, i.e., *gaps* occur. In Figure 12.8 examples for global alignment without gaps, global alignment with gaps, and local alignment are given. One can extend the alignment of two sequences to *multiple* alignments of more sequences in a straightforward way [8], but here we will concentrate on pairwise alignment.

To each alignment a score is assigned, via a scoring function $S(\mathbf{x}, \mathbf{y})$. The score is chosen in such a way, that a high similarity leads to a high score. This can be done by choosing the total score as the sum of scores of all pairs of aligned letters plus the costs of all gaps $S(\mathbf{x}, \mathbf{y}) = \sum_k s(x_{i_k}, y_{j_k}) + \sum_{\text{gaps } g} f(l_g)$. The gap costs are usually taken as a function of the length l_g of the gaps. Gap costs are discussed later. The part of the score for the aligned letters is a linear function of the alignment and neighboring pairings are independent, which is a simple approximation. The scores for equal or similar letters, "similar" meaning that the biological function of the corresponding units is similar, are positive, while the scores for very different letters are negative. For now one can assume that the score matrices $s(a, b)$ (also called *substitution matrices*) are just given, but below, the biological origin of the scores is outlined. Also the part of the score for the gaps is taken to be a sum over all gaps. The scores for the gaps are chosen to be negative, because the need to leave out some letters which are being aligned indicates that the compared sequences are dissimilar in some way.

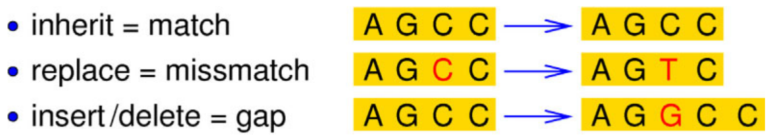


Figure 12.9: The evolution changes the DNA/amino-acid sequences. Inheritance and mutations correspond to matching, resp. missmatchings, and gaps of the alignment.

The score matrices cannot be "derived" in some defined way, but many choices are possible, depending on the way in which the data is to be analyzed. The fundamental concept behind the choice of the scores is that the sequences are generated by evolution. This is directly obvious for DNA. Proteins are produced in all cells, the blueprint is encoded in their DNA. During evolution, the DNA changes, hence the expressed proteins change. Life forms, which are closer in evolution, will have proteins which are closer as well. Thus, one can also speak of the evolution of the proteins itself. This evolution of a protein or the DNA can be drawn in the form of a (phylogenetic) tree. At each node of the tree, different forms originating from the same predecessor emerge. Parts of a sequence which are inherited to subsequent forms, correspond to matches in the alignments; mutations, where bases or amino acids are replaced, correspond to mismatches, while insertions or deletions of code correspond to gaps, see also Figure 12.9.

Here we consider score matrices for proteins. From the analysis of evolution a lot of data is already available. For the most simple approach in constructing a score matrix, one can take a probabilistic approach, where no gaps and only global alignments are considered for simplicity. A sample of already known sequences and the corresponding phylogenetic tree is assumed to be available. One can now just compare all sequences in the sample and obtain the frequencies/probabilities p_{ab} that the two amino acids a, b are derived from the same ancestor c (i.e., c might be a or b or different). If one assumes independence of the different mutations, the probability that the alignment of two sequences $\mathbf{x}, \mathbf{y}$ will occur is $P(\mathbf{x}, \mathbf{y}|\mathrm{A}) \equiv \prod_i p_{x_i y_i}$. Let f_a be the probability that a letter a appears at all in the sample. Then the probability that the alignment appears by pure chance is $P(\mathbf{x}, \mathbf{y}|\mathrm{R}) = \prod_i f_{x_i} f_{y_i}$. The so called *odds ratio* is the ratio of these probabilities:

$$\frac{P(\mathbf{x}, \mathbf{y}|\mathrm{A})}{P(\mathbf{x}, \mathbf{y}|\mathrm{R})} = \prod_i \frac{p_{x_i y_i}}{f_{x_i} f_{y_i}} \tag{12.1}$$

Now the score is just defined as the logarithm of the odds ratio. Due to the additivity of the logarithm for products, the score of the full alignment of the two sequences becomes a sum of scores $S = \sum_i s(x_i, y_i)$ of the paired letters with

$$s(a, b) = \log\left(\frac{p_{ab}}{f_a f_b}\right) . \tag{12.2}$$

For this approach, one must define which time window is analyzed, within which the sequences have evolved, i.e. how far in evolution the sequences are away, and which are being compared. The larger the time interval from which the data originates, the more different the sequences will be, hence the probabilities p_{ab} will change.

One frequently-used score matrix which uses this approach are the PAM (*point accepted mutation*) matrices by Dayhoff, Schwartz and Orcutt [7]. Their first step was to study only the direct ancestors in the phylogenetic tree which they used, leading to substitution matrices for one unit of time, called PAM 1. This matrix was extended to longer times, by raising it to the tth power, resulting in the probabilities $P(b|a, t)$ that a is substituted by b after t steps. Since $P(b|a) = p_{ab}/f_a$, the final score of the PAM t matrix according to Eq. (12.2) is then $s(a, b|t) = \log P(b|a, t)/f_b$. These values are scaled by some factor and rounded to the nearest integer. The most common form is the PAM 250 matrix with scaling factor $3/\log 2$.

Since the PAM matrices result basically from studying short-time mutations, long-time influences are not taken into account, e.g., changes of codons leading to mutations of proteins, where the corresponding codes differ in more than one base, are not considered. For this reason, the BLOSUM matrix set was developed [14]. Within this approach, related proteins from the BLOCK database [13] were clustered (again without allowing for gaps), where the proteins in each cluster agreed in at least a fraction L of the letters. Then the sequences from *different* clusters were compared and the frequencies of substitution measured. Then the single-letter alignment scores were obtained, after a suitable normalization, using the standard method Eq. (12.2). Finally the data is usually rescaled and rounded to the nearest integer, resulting in the BLOSUM L matrix. BLOSUM62 is the standard method used for alignments.

Whatever scoring matrix is used, in the case of local alignment, the expected score of two fully random sequences must be negative, i.e., $\sum_{a,b} s(a, b) f_a f_b < 0$. Otherwise, extending an alignment would on average always increase the score, hence no local alignment would occur when looking for large scores. Whenever the scores are derived from a likelihood ratio, as in Eq. (12.2), this is automatically fulfilled.

The way gap costs are set up is even more heuristic than for the alignment scores. The basic idea is as follows. When, due to a mutation, a subsequence of a protein is removed to form a new protein, the corresponding optimum alignment of both proteins will have a gap at this position. Since a longer mutation is less likely than a shorter one, it is reasonable to increase the gap costs with the gap length. But having two independent mutations of length l is less likely than having one mutation of length $2l$. Hence the corresponding gap costs should reflect this behavior as well. This is fulfilled for the so called *affine* gap costs (α, β) where a gap of length l has costs $g(l) = -\alpha - \beta(l-1)$. The most common cost function is (12,1) gap costs, but many other approaches exist as well.

For a *global* alignment without gaps ($n = m$), there is just one alignment, hence just one score, no algorithm is needed. When allowing for gaps, several alignments are possible, hence one can have for similar, even equal sequences $\mathbf{x}, \mathbf{y}$, a very low score when aligning the "wrong" letters. Therefore, one defines the similarity as the *optimum* global alignment $S_g(\mathbf{x}, \mathbf{y})$ with gaps, which is obtained by maximizing the score over all numbers K of aligned pairs and over all possible placements of the gaps. The optimum *local* alignment S is the maximum over all possible contiguous subsequences $\tilde{\mathbf{x}} = x_i x_{i+1} \dots x_{i+l-1}$, $\tilde{\mathbf{y}} = y_j y_{j+1} \dots y_{j+k-1}$ of the optima $S_g(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$. Here again cases both with and without allowing gaps are possible. Hence, since an alignment of zero length has a score of zero, the optimum local alignment is always non-negative by definition.

For both global and local alignment with gaps efficient algorithms [8, 10, 24, 30] exist, which calculate an optimum alignment in time $O(nm)$. They will be presented next. In principle all these algorithms use a dynamic programming approach. This is possible because the score function is linear, hence the optimum alignment can be decomposed into a series of optimum alignments of shorter sequences. We will start with an algorithm for global alignments with gaps. For the moment only *linear* gap costs, i.e., (0,d) affine gap costs will be considered. This is the *Needleman–Wunsch* algorithm [24].

The basic idea is that an optimum alignment of the partial sequences $\tilde{\mathbf{x}} = x_1 x_2 \dots x_i$, $\tilde{\mathbf{y}} = y_1 y_2 \dots y_j$ can be obtained by either extending the optimum alignment of $\tilde{\mathbf{x}} = x_1 x_2 \dots x_{i-1}$, $\tilde{\mathbf{y}} = y_1 y_2 \dots y_{j-1}$ by aligning x_i and y_j, or by extending either the optimum alignment of $\tilde{\mathbf{x}} = x_1 x_2 \dots x_{i-1}$, $\tilde{\mathbf{y}} = y_1 y_2 \dots y_j$ resp. $\tilde{\mathbf{x}} = x_1 x_2 \dots x_i$, $\tilde{\mathbf{y}} = y_1 y_2 \dots y_{j-1}$ by a gap. The real optimum alignment is the maximum over all three possible scores. The optimum score of the sequences up to letters i and j is denoted by $F(i,j)$, hence a recursive function is obtained

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{align} \\ F(i,j-1) - d & \text{gap in x} \\ F(i-1,j) - d & \text{gap in y}\,. \end{cases} \tag{12.3}$$

Since the alignment of one sequence with nothing is a gap, the recursion is started by $F(0,l) = F(l,0) = -d \times l$. The optimum alignment of the full sequences is then the rightmost entry in the bottom line $F(n,m)$.

As a simple example, we consider the (DNA) sequences $\mathbf{x}$ =AAGTGT and $\mathbf{y}$ =AGTCGA. For a match of two equal bases, the score $s(a,a) = 1$ is used, while for a mismatch an $s(a,b) = -3$ penalty is given. A gap has cost $-d = -2$. In Figure 12.10 the resulting

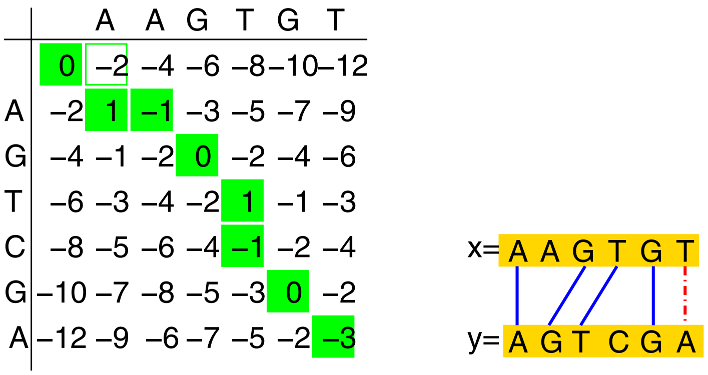


Figure 12.10: Example of how the Needleman–Wunsch algorithm works. The dynamic-programming matrix $F(i, j)$ and the resulting optimum alignment are shown.

dynamic-programming matrix $F(i, j)$ is shown together with the resulting maximum alignment. As an example, we consider how the score $F(1, 1)$ is calculated. Either x_1 aligned to a gap (score -2) or y_1 aligned to a gap (score -2) is extended by a y_1 resp. x_1 to a gap (score -2), leading to the total score of -4. Or the two letters x_1 =A and y_1 =A (score s(A,A) $= 1$) are aligned. Since $1 > -4$ the later possibility is taken. The remaining matrix is calculated in the same way.

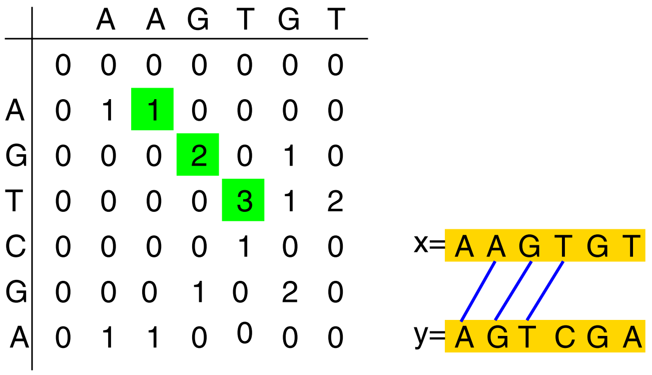
Since the dynamic-programming matrix $F(i, j)$ has $n \times m$ entries, and the calculation of each entry takes only constant time, the full algorithm runs in $O(nm)$ steps. Note that the actual alignment can be inferred from the calculation of the matrix as well, by always storing a pointer backwards to the previous optimum, from which the current optimum was obtained. Then the optimum alignment can be read off the matrix by starting at the element (m, n) and going backwards following the pointers. In the example in Figure 12.10 the "path" when going backwards in the matrix is indicated by shaded squares. The actual alignment may not be unique in the case when several possibilities in (12.3) lead to the same score.

Indeed the Needleman–Wunsch algorithm is a shortest–path algorithm for a special geometry. The nodes of the underlying graph are arranged in a rectangular array. Edges exist only between nearest neighbors and for two of the four possible next-nearest neighbors (pointing right downwards). Since the edges are directed and point into the direction of increasing sequence lengths, the corresponding graph contains no loops. Hence a very simple shortest–path algorithm can be applied. Note that other pattern-matching algorithms, like the UNIX command *grep*, work in a similar fashion.

The extension to find the optimum local alignment with linear gap costs is straightforward and was given by Smith and Waterman [30]. Since one can always align nothing, which gives score 0, it does not make sense to extend a subsequence with negative score, instead a new subsequence is started. This leads to the following recursion:

$$F(i,j) = \max \begin{cases} 0 & \text{restart} \\ F(i-1,j-1) + s(x_i,y_j) & \text{align} \\ F(i,j-1) - d & \text{gap in x} \\ F(i-1,j) - d & \text{gap in y} \end{cases} \tag{12.4}$$

Since gaps have always a negative cost, the recursion is started by $F(0,l) = F(l,0) = 0$. The final optimum alignment is the maximum over all matrix elements $S = \max_{i,j} F(i,j)$. When storing backpointers again, the optimum alignment can be obtained by starting from the element (i_0, j_0) where the maximum is found. Again, the optimum alignment might not be unique.



		A	A	G	T	G	T
	0	0	0	0	0	0	0
A	0	1	1	0	0	0	0
G	0	0	0	2	0	1	0
T	0	0	0	0	3	1	2
C	0	0	0	0	1	0	0
G	0	0	0	1	0	2	0
A	0	1	1	0	0	0	0

Figure 12.11: Example of how the Smith–Waterman algorithm works. The dynamic-programming matrix $F(i,j)$ of partial scores and the resulting optimum local alignment are shown.

In Figure 12.11 the resulting dynamic-programming matrix $F(i,j)$ along with the optimum local alignment for the same two sequences as above are shown.

The most common form used to compare protein sequences is local alignment with (α, β) affine gap cost. The algorithm [10] is also a dynamic programming algorithm. But now one recursion relation is not sufficient here. The reason is that the costs for opening a gap are different from the costs of extending a gap. Therefore, one uses three dynamic-programming matrices, i.e.,

- $A(i,j)$, the optimal score up to (i,j), given that x_i and y_j are aligned.
- $G_x(i,j)$, the optimal score up to (i,j), given that x_i is aligned to a gap.
- $G_y(i,j)$, the optimal score up to (i,j), given that y_i is aligned to a gap.

The recursive equations, for the case of local alignment, read as follows:

$$vA(i,j) = \max\begin{cases} 0 & \text{restart} \\ A(i-1,j-1)+s(x_i,y_j) & \text{align} \\ G_x(i-1,j-1)+s(x_i,y_j) & \text{align} \\ G_y(i-1,j-1)+s(x_i,y_j) & \text{align} \end{cases} \tag{12.5}$$

$$G_x(i,j) = \max\begin{cases} 0 & \text{restart} \\ A(i-1,j)-\alpha & \text{new gap} \\ G_y(i-1,j)-\alpha & \text{new gap} \\ G_x(i-1,j)-\beta & \text{extend gap} \end{cases} \tag{12.6}$$

$$G_y(i,j) = \max\begin{cases} 0 & \text{restart} \\ A(i,j-1)-\alpha & \text{new gap} \\ G_x(i,j-1)-\alpha & \text{new gap} \\ G_y(i,j-1)-\beta & \text{extend gap}\ . \end{cases} \tag{12.7}$$

The actual total optimal score up to (i,j) is $F(i,j) = \max\{A(i,j), G_x(i,j), G_y(i,j)\}$ and the local optimum is obtained again as the optimum of $F(i,j)$ over all elements of the dynamic-programming matrix. Retrieving the actual alignment is performed again by pointers, but now one has to store one pointer for each matrix, which points to the element in the actual matrix which was the previous one in the optimal subsequence. For example, if opening a new gap after a pair of aligned letters led to the optimal score for $G_y(i,j)$, then there will be a pointer from $G_y(i,j)$ back to $A(i,j-1)$.

There are several algorithms for other types of alignment [8]. When looking for *repeated matches*, one allows that a subsequence of $\mathbf{x}$ is aligned several times to different subsequences of $\mathbf{y}$. For *overlap alignments*, one allows that one sequence contains the other, or that they overlap. This is a special form of local alignment, with the restriction that at least one prefix and one postfix of the two sequences have to be aligned. One can obtain even more complex algorithms, by describing them by finite-state automates, e.g., for local alignment with affine gap cost, one would have three states *aligned*, *x-gap-ed* and *y-gap-ed*. With more states, more evolved algorithms can be realized, e.g. if one wants to score different regions of a protein in different ways. All these algorithms have in common that they run in $O(nm)$. In the case of multiple alignment, i.e., when one is interested in patterns that appear in more than two sequences, no polynomial-time algorithms are known. Due to this slow running time, multiple alignment is not used for large databases.

For the largest public databases, which have millions of sequences stored, even the $O(nm)$ algorithms are too slow, since an answer within a few seconds is required. For this reason, often fast heuristics are applied, which do not guarantee to give the exact optimum alignment. For example, the BLAST package [1] works by first searching for short subsequences with match exactly, and then trying to extend these subregions to form larger high-scoring subregions. BLAST is used in the protein database Swiss–Prot [33].

In the next chapter, the statistics of alignment scores for random sequences are studied. Using a special algorithm originating in statistical physics, one can greatly reduce the number of sequences one has to compare in order to get good results. Thus, no heuristic algorithm is necessary, an exact $O(nm)$ algorithm can be applied instead.

12.3 Low-probability Tail of Alignment Scores

The result of a sequence alignment is a maximum score, e.g., $S = 200$. Now one wants to estimate the meaning of this score: does it mean a high similarity or a low one? Thus, one wants to know the *significance* of the result. One way to estimate the significance, is to compare it against complete randomness, i.e., to calculate the probability $p(S)$ that such a (large) score can be obtained by pure chance.

For biologically relevant models, e.g., for protein sequences with BLOSUM62 substitution scores [14] and affine gap costs [10], $p(S)$ is not known in the interesting region, where $p(S)$ is small. A number of empirical studies [2, 6, 23, 31, 35, 36] for local alignment, in the region where $p(S)$ is large, suggest that $p(S)$ is an *extreme value* (or Gumbel) distribution [12]

$$p_G(S) = \lambda e^{-\lambda(S-u)} \exp\left(-e^{-\lambda(S-u)}\right), \tag{12.8}$$

where u denotes the maximum of the distribution and λ characterizes the behavior for large values of S, i.e., the tail of the distribution.

As an attempt to check whether (12.8) holds, one can easily generate, e.g., $N \approx 10^5$ samples of pairs of sequences according to the frequencies f_a, obtain for each sample the optimum alignment, and calculate a histogram of the optimum scores S, as in the above mentioned previous studies [2, 6, 23, 31, 35, 36]. This *simple sampling* allows one to calculate $p(S)$ in the region where the probabilities are large (e.g., $p(S) \geq 10^{-4}$). Hence it is actually the region of the distribution, where the relevant scores are *not* found, since the proteins generated by evolution are indeed *not* random. Recently, the *island method* [3] was introduced, which allows a speed up several orders of magnitude for very long sequences like $n = 10^4$, but still the far end of the distribution is out of reach. Also, please note that biologically relevant protein sequences have lengths of only a few hundred amino acids.

Here, to determine the tail of $p(S)$, a *rare event simulation* is applied. For dynamical problems, such as investigating queuing systems or studying the reliability of technical components, several techniques [27] have been developed. Related methods have been introduced in physics [4, 11].

By simply changing perspective, one can apply these standard techniques to many other problems. Here, the method is applied to sequence alignment. The basic idea is that one uses a physical system, which has a state given by a pair of sequences and is held at temperature T, instead of directly drawing the random sequences. This idea is similar to the simulated annealing approach [19], used to find approximate solutions of hard optimization problems. But the method presented here goes far beyond simulated annealing, because it is not only the minimum of one system, but the whole distribution over all random instances that is sampled during one run. The state of the system changes in time, governed by the rules of statistical mechanics. The energy E of the system is defined as $E = -S$. Therefore, at low temperatures, the system prefers pairs of sequences with high optimum score value S. Since the thermodynamic properties of such a system are known in principle, it is possible to extract from the measured distribution $p^*_T(S)$ of optimum scores the target distribution $p(S)$.

To determine the behavior of $p(S)$ at the rare event tail (e.g., $p(S) \approx 10^{-40}$), one views each pair $c = (\mathbf{x}, \mathbf{y})$ of sequences as the state of the physical system, which behaves according to the rules of statistical mechanics, with $-S$ being the energy of the system. More precisely, instead of considering many independent pairs of fixed sequences, a Markov chain [20]

$c(0) \to c(1) \to c(2) \to \ldots$ of pairs is used to generate the instances. For each instance $c(i)$, the optimum local alignment score S is calculated. Let $p(c \to c')$ denote the transition probability from state c to state c'. Changing the sequences dynamically is similar to annealed disorder simulations [29, 32, 34]. There, the quenched disorder (in this case the sequences) *and* the dynamic variables (in this case the alignment) are degrees of freedom, i.e., allowed to change during the annealed simulation. While the physics of an annealed system is different from the physics of the related quenched system, here an annealed-disorder-like simulation is used via the application of a simple transformation (see below) to obtain the *true* behavior of the quenched system.

The simplest rule for the transition is, to choose randomly a position in one of the sequences with all positions being equiprobable and to choose randomly a new letter from the alphabet, the letters having probabilities f_a, i.e. $p(c \to c') = f_a/(n+m)$ if c, c' differ by at most one letter, and $p(c \to c') = 0$ otherwise. So far no temperature T and no statistical mechanics is involved. With this choice of the transition probabilities, iterating this step t times, for $t \to \infty$ all possible pairs of sequences have the probability $P(c) = \prod_i f_{x_i} \prod_j f_{y_j}$ of occurrence. Hence, simple sampling is reproduced.

To increase the efficiency, one can change the sampling distribution [22, 27], a standard method for simulating rare events [18], which allows one to concentrate the sampling in small regions in configuration space. A good choice for the transition rule of the sequence-alignment problem is first to change one position of one sequence randomly as above and recalculate the optimum alignment $S(c')$ with a standard algorithm as described in the previous section. This move $c \to c'$ is accepted with the Metropolis probability [22]

$$P_{\mathrm{Metr}} = \max(1, \exp(\Delta S/T)), \quad \text{where } \Delta S = S(c') - S(c)\,. \tag{12.9}$$

This leads to the equilibrium state of a physical system at temperature T with energy $E = -S$, with the distribution weighted by the sequence probabilities $P(c)$. The advantage of this approach is that the equilibrium distribution $Q(c)$ is known from statistical physics [26]: $Q(c) = P(c)\exp(S(c)/T)/Z$ with $Z(T) = \sum_c P(c)\exp(S(c)/T)$ being the partition function. Thus, the estimator for the probability to have score S in the ensemble at temperature T is

$$p_T^*(S) = \sum_c{}' Q(c) = \frac{\exp(S/T)}{Z(T)} \sum_c{}' P(c)\,, \tag{12.10}$$

where the sum $\sum'$ runs over all sequences with score S.

Thus, from the measured histogram of scores $p_T^*(S)$ one obtains the estimator for the unbiased distribution through

$$p(S) = \sum_c{}' P(c) = p_T^*(S) Z(T) \exp(-S/T)\,. \tag{12.11}$$

$Z(T)$ is unknown *a priori*, but can be determined very easily, as shown below.

Please note a striking difference to conventional Monte-Carlo (MC) simulations of random systems. For the conventional approach, when studying quenched-disorder systems, different random samples of the quenched disorder are studied by MC, each sample having the predefined probability $P(c)$. Within the method presented here, a biased simulation is done *on the* *disorder*, while the behavior of each random sample is determined exactly, resulting finally in the unbiased distribution over the disorder. Note that to describe the behavior of $p(S)$ over a wide range, the model must be simulated at several temperatures. Hence, to enhance the efficiency of the simulation, the *parallel tempering* method [15,21] may be applied, where several realizations are kept in parallel at different temperatures and exchanges between neighboring temperatures occur.

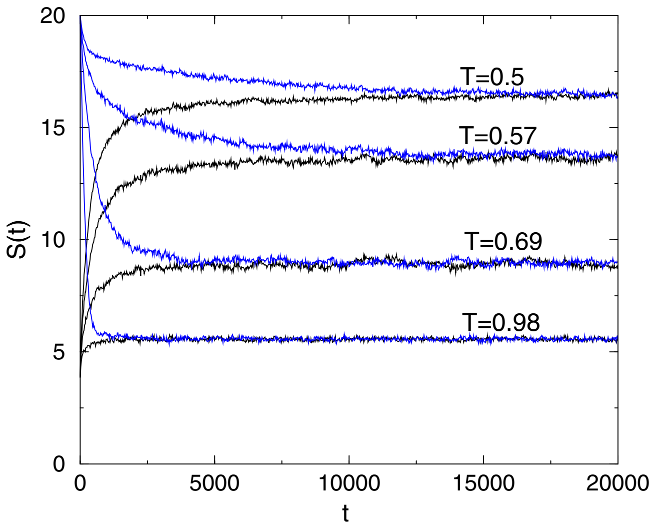


Figure 12.12: Average alignment score S as a function of step t for a toy model ($n, m = 20$, 4 letters, local alignment without gaps) for different temperatures T. For each temperature, 1000 independent simulations were started with two random sequences (low scores) and 1000 simulations with two equal sequences (high scores).

For the Monte Carlo simulation, one has to ensure *equilibration*. To check equilibration in our case, two types of runs are performed:

(a) Initially, all pairs of sequences are random, i.e., the score is low at the beginning.

(b) Initially, each pair consists of two equal sequences. Thus, the score is maximal at the beginning, i.e., the system is in the ground state.

In this way equilibration can be checked easily: if after time t_0 the score averaged over several independent runs for both initial configurations agree within error bars, the simulation is long enough.

Next, a simple example is given, illustrating how the method works. Optimum local alignments without gaps for sequences of equal length $m = n = 20$ and $r = 4$ letters, all having the same probability $1/4$, are calculated. For the test, the following score is applied: $s(x, y) = 1$ if $x = y$ and $s(x, y) = -3$ otherwise. This is the same model as was considered in the previous chapter to illustrate the alignment algorithms.

In Figure 12.12 the average optimum score S for the beginning 10% of the running time of 1000 independent runs and four different temperatures T is shown. For not too small temperatures, the system equilibrates quickly. For very low temperatures (not shown in the figure), equilibration may not be possible, hence the data for these temperatures cannot be used. This means basically, that the artificial dynamics of the disorder exhibits a glassy phase, which might be worthy of its own study. However, as we will see, the data obtained at higher temperatures is sufficient to obtain the true distribution $p(S)$ over many orders of magnitude.

To obtain weakly correlated samples when measuring $p^*_T(S)$, only values at t_0, $t_0 + \tau$, $t_0 + 2\tau$ etc. are taken, where τ is the characteristic time in which the score–score correlation $c_S(t_0, t) = (\langle S(t_0)S(t)\rangle - \langle S\rangle^2)/(\langle S^2\rangle - \langle S\rangle^2)$ decreases to $1/e$.

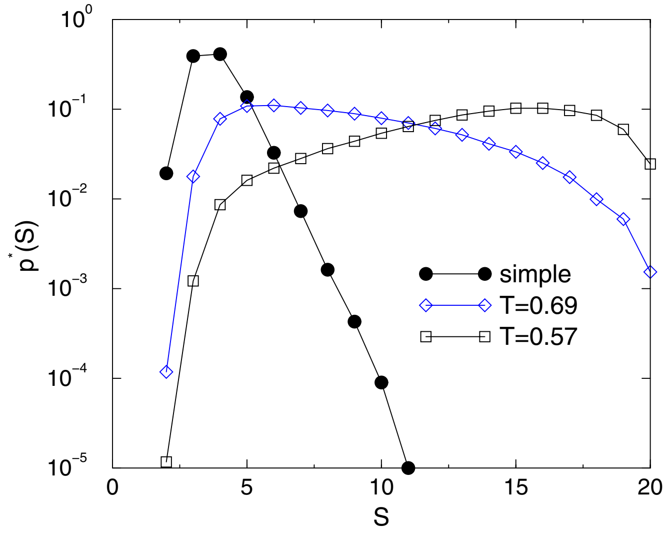


Figure 12.13: The four-letter toy model. Raw distribution of alignment scores S, for the direct simulation, and the distributions $p^*_T(S)$ obtained at finite temperatures $T = 0.57$ and $T = 0.69$.

In Figure 12.13 the raw distributions $p^*_T(S)$ for two temperatures are shown together with a distribution from a simple sampling of $N = 10^4$ realizations. Clearly, with the statistical mechanics approach, the region of high scores is sampled much more frequently.

The first step, in obtaining the final distribution $p(S)$ according to Eq. (12.11), is to rescale the measured distributions $p^*_T(S)$ with the Boltzmann factor $\exp(-S/T)$. The result is displayed in Figure 12.14.

Hence, only the partition functions $Z(T)$ remain to be determined to obtain the final result. The starting point is the unbiased distribution which is known to be correct in the region of high probabilities. Now, in an interval $[S_1, S_2]$, the data of the simple sampling and the highest temperatures overlap. Hence, the finite-temperature data must be shifted (on a logarithmic scale) such that it becomes identical with the simple-sampling data, resulting in $Z(T)$ for the highest temperature. In the same way $Z(T)$ at lower temperatures can be obtained by matching to distributions obtained before at higher temperatures. The final distribution is

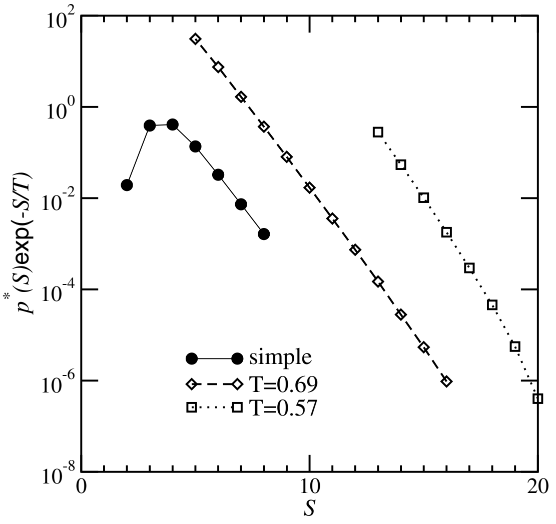


Figure 12.14: Distribution of alignment scores for the direct simulation and and rescaled distributions $p^*_T(S)\exp(-S/T)$ for $T = 0.57$ and $T = 0.69$. Only the regions with the best statistics are displayed.

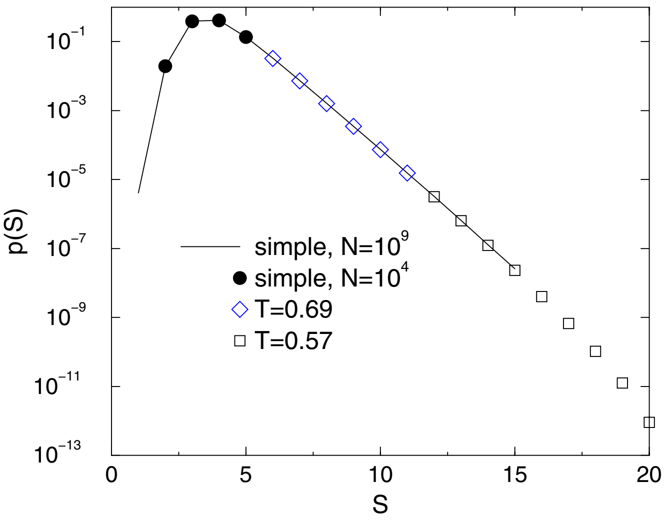


Figure 12.15: Rescaled distribution $p(S)$ for the direct simulation and for $T = 0.57, T = 0.69$. The solid line is the result of a large simple-sampling simulation with $N = 10^9$ samples.

shown in Figure 12.15. For each data point, the distribution with the highest accuracy was taken. For comparison, a simple-sampling distribution obtained by using a huge number of samples ($N = 10^9$) is shown. Both results agree very well. Please note that the distribution from the finite-T approach spans almost the entire interval $[0, 20]$. In principle, the region for very small score S can also be investigated using the new method by simulating at *negative* temperatures. How powerful the new method is can be seen by looking at the right border of the interval, where a value $p(20) = 9.13(20) \times 10^{-13}$ was obtained. This agrees within error bars with the exact result $0.25^{20} \approx 9.09 \times 10^{-13}$ (only when $\mathbf{x} = \mathbf{y}$ the highest possible score is achieved). Also the same model with $(3, 1)$ gap costs was tried and again a perfect agreement with a huge simple-sampling simulation was found. This example illustrates that the method presented here is indeed able to calculate accurately the distribution $p(S)$ of optimum alignment scores in regions where $p(S)$ is very small.

In Ref. [17] the method was applied for a biologically relevant case. Sequences of amino acids distributed according to the background frequencies by Robinson and Robinson [28] were used, together with the BLOSUM62 scoring matrix [14] for $(12, 1)$ affine gap costs. This type of system was previously studied in Ref. [2] in the region where $p(S)$ is large. Sequences of length $n = m$ in the range $[40, 400]$ were considered. The main result was that, for high scores, significant deviations from the pure Gumbel behavior were visible, in contrast to the earlier predictions. Since the deviations occur at high score values, they could not be detected before by using conventional methods. From a theoretical point of view it is interesting that with increasing lengths n, m, on a scale of scores proportional to $u \sim \log n$, the measured $p(S)$ approaches the Gumbel distribution more and more closely. For practical applications, where short and medium length sequences are relevant, the measured distribution rather than a fitted Gumbel distribution should be used in databases like Swiss–Prot [33] to achieve the highest possible accuracy.

Acknowledgments

The author is grateful to Bernd Burghardt and Joachim Schmidt-Brauns for critically reading the manuscript. Figure 12.1 was provided generously by H. Grubmüller. The author obtained financial support from the VolkswagenStiftung (Germany) within the program “Nachwuchsgruppen an Universitäten”.

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, *Basic local alignment search tool*, J. Mol. Biol. **215**, 403 (1990).

[2] S.F. Altschul and W. Gish, *Local alignment statistics*, Methods in Enzymology **266**, 460 (1996).

[3] S.F. Altschul, R. Bundschuh, R. Olsen, and T. Hwa, *The estimation of statistical parameters for local alignment score distributions*, Nucl. Acid Res. **29**, 351 (2001).

[4] B.A. Berg and T. Neuhaus, *Multicanonical ensemble: a new approach to simulate first-order phase transitions*, Phys. Rev. Lett. **68**, 9 (1992).

[5] S.M. Brown, *Bioinformatics*, (Eaton Publishing, Natick (MA) 2000).

[6] J.F. Collins, A.F.W. Coulson, and A. Lyall, *The significance of protein-sequence similarities*, CABIOS **4**, 67 (1988).

[7] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt, *A model of evolutionary change in proteins*; in: M.O. Dayhoff (ed), *Atlas of Protein Sequence and Structure* **5**, suppl. 3, (National Biomedical Research Foundation, Washington D.C., 1978).

[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*, (Cambridge University Press, 1998).

[9] W.H. Elliott and D.C. Elliott, *Biochemistry and Molecular Biology*, (Oxford University Press, Oxford 2001).

[10] O. Gotoh, *An improved algorithm for matching biological sequences*, J. Mol. Biol. **162**, 705 (1982).

[11] P. Grassberger, *Prune-enriched Rosenbluth method: Simulations of Θ polymers of chain length up to 1000000*, Phys. Rev. E **56**, 3682 (1997).

[12] E.J. Gumbel, *Statistics of Extremes*, (Columbia Univ. Press, New York 1958).

[13] S. Henikoff and J.G. Henikoff, *Automated assembly of protein blocks for database searching*, Nucl. Acids. Res. **19**, 6565 (1991).

[14] S. Henikoff and J.G. Henikoff, *Amino-acid substitution matrices from protein blocks*, Proc. Natl. Acad. Sci. USA **89**, 10915 (1992).

[15] K. Hukushima and K. Nemoto, *Exchange Monte Carlo method and application to spin glass simulation*, J. Phys. Soc. Jap. **65**, 1604 (1996).

[16] International Humane Genome Sequencing Consortium, Nature **409**, 860 (2001).

[17] A.K. Hartmann, *Sampling rare events: statistics of local sequence alignments*, Phys. Rev. E 65, 056102 (2002).

[18] V. Kalashnikov, *Geometric Sums: Bounds for Rare Events with Applications*, (Kluwer Academic Publishers, Dordrecht 1997).

[19] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, *Optimization by simulated annealing*, Science **220**, 671 (1983).

[20] S. Lipschutz and M. Lipson, *Probability*, (McGraw-Hill, New York 2000).

[21] E. Marinari and G. Parisi, *Simulated tempering – a new Monte-Carlo scheme*, Europhys. Lett. **19**, 451 (1992).

[22] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. **21**, 1087 (1953).

[23] R. Mott, *Maximum-likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores*, Bull. Math. Biol. **54**, 59 (1992).

[24] S.B. Needleman and C.D. Wunsch, *A general method applicable to search for similarities in amino acid sequence of 2 proteins* J. Mol. Biol. **48**, 443 (1970).

[25] H.H. Rashidi and L.K. Buehler, *Bioinformatics Basics*, (CRC Press, Boca Raton (FL) 2000).

[26] L.E. Reichl, *A Modern Course in Statistical Physics*, (Wiley, New York 1998).

[27] B.D. Ripley, *Stochastic Simulation*, (Wiley, New York 1987).

[28] A.B. Robinson and L.R. Robinson, *Distribution of Glutamine and Asparagine residues and their near neighbors in peptides and proteins*, Proc. Natl. Acad. Sci. USA, **88**, 8880 (1991).

[29] P.J. Shah and O.G. Mouritsen, *Dynamics of ordering process in annealed diluted systems – island formation, vacancies at domain boundaries, and compactification* Phys. Rev. B **41**, 7003 (1990).

[30] T.F. Smith and M.S. Waterman, *Identification of common molecular subsequences*, J. Mol. Biol. **147**, 195 (1981).

[31] T.F. Smith, M.S. Waterman, and C. Burks, *The statistical distribution of nucleic-acid similarities*, Nucleic Acids Res. **13**, 645 (1985).

[32] R.B. Stinchcombe, in C. Domb and M.S. Green (eds.) *Phase Transitions and Critical Phenomena* **7**, (Academic, New York 1983).

[33] Swiss–Prot, e.g. at the European Molecular Biology Laboratory (EMBL) in Heidelberg, see `http://dove.embl-heidelberg.de/Blast2/`.

[34] R. Voča, *Transport on an annealed disordered lattice*, Phys. Rev. E **60**, 3516 (1999).

[35] M.S. Waterman and V. Vingron, *Rapid and accurate estimates of statistical significance for sequence data-base searches*, Proc. Natl. Acad. Sci. USA **91**, 4625 (1994).

[36] M.S. Waterman and V. Vingron, *Sequence comparison significance and Poisson approximation*, Stat. Sci. **9**, 367 (1994).

13 Protein Folding *in Silico* – the Quest for Better Algorithms

Ulrich H.E. Hansmann

13.1 Introduction

Proteins are one of the most common and important class of molecules in living systems. In the human body, they form muscles and connective tissues, and as enzymes, proteins catalyze and regulate biochemical reactions in the cell. While differing greatly in size and structure, all proteins are chemically long linear chain molecules with the twenty naturally occurring amino acids as monomers. Regular elements like helices, sheets and turns are formed locally, but for the biological function of a protein the most important characteristic is its unique overall three-dimensional shape that is determined solely by the sequence of amino acids.

The sequence of amino acids that make up a protein is specified in the human genome. Hence, after the successful completion of the human genome project one knows in principal the chemical composition of all proteins in the human body. However, this achievement has only aggravated an old challenge in protein science: for most of the resolved protein sequences one does not know the corresponding structures. Since proteins are only functional if they fold into their specific shape, it is important to understand how the structure and function of proteins emerge from their sequence of amino acids. A detailed knowledge of the sequence-structure (function) relation would allow us to understand better the (mal)function of enzymes, and could yield more efficient methods of drug design.

The problem is amplified by the difficulties in solving experimentally the structure of proteins. While it takes only hours to days to determine the sequence of amino acids of a protein, months to years are needed to find out the corresponding 3D shape by X-ray crystallography or nuclear magnetic resonance (NMR) experiments. Equally challenging are experiments to explore the kinetics and dynamics of the folding process. For these reasons, there is considerable interest in finding alternative ways to tackle the protein-folding problem.

One possibility is the use of computer experiments. Most proteins are at room temperature thermodynamically stable, i.e., the biologically active configuration is the global minimum in *free* energy at $T = 300$ K. Since this state is *unique*[1], one can identify it with the lowest *potential* energy conformation [3]. Hence, structure prediction of proteins is a global optimization problem.

As with all optimization problems, the choice of an appropriate energy function is of crucial importance. Often used in protein simulations are minimal models that capture only a few,

[1] "Unique" means here that there is only one *structure*. Oscillations around this state exist, but the entropy of these microstates is negligible when compared with the entropy of the denatured states.

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6

but presumably dominant, interactions in proteins [11,39]. These simplified models proved to be successful in exploring the general characteristics of possible folding mechanisms. However, calorimetric measurements show that a protein in its native state is only marginally more stable (by a free-energy difference of $\approx 10 - 20$ kcal/mol) than the ensemble of the denatured conformations. Simplified models lack the necessary precision to describe such small differences and therefore do not allow investigation of the details of structural transitions and folding in *specific* proteins. For the latter purpose, one has to utilize more realistic models where the interactions among all atoms are taken into account. The resulting potential energy $E_{\text{tot}} = E_{\text{protein}} + E_{\text{solv}}$ (given in kcal/mol) is often written as a sum of two terms. The first term, E_{protein}, describes the interactions between all atoms within a protein, and the second term, E_{solv}, the interaction of a protein with the surrounding water.

The later term especially is a serious hurdle since explicit inclusion of water molecules is computationally demanding. Hence, one often has to rely on implicit solvent models. Very often a solvent-accessible surface term is introduced that accounts in an empirical way for the hydrophobic forces on the protein [40]

$$E_{\text{solv}} = \sum_i \sigma_i A_i \ . \tag{13.1}$$

Here A_i is the solvent-accessible surface area of the ith atom in a given configuration, and σ_i is the empirically determined solvation parameter of the atom i.

On the other hand, the intramolecular interactions of E_{protein} are modeled in various atomic force fields such as ECEPP [43], CHARMM [8] or AMBER [47]. I show here as an example the ECEPP energy function that is given by the sum of an electrostatic term E_{es}, a van der Waals energy E_{vdW}, and a hydrogen-bond term E_{hb} for all pairs of atoms in the peptide, together with a torsion term E_{tors} for all torsion angles:

$$E_{\text{ECEPP}} = E_{\text{es}} + E_{\text{vdW}} + E_{\text{hb}} + E_{\text{tors}} \ , \tag{13.2}$$

$$E_{\text{es}} = \sum_{(i,j)} \frac{332 q_i q_j}{\epsilon r_{ij}} \ , \tag{13.3}$$

$$E_{\text{vdW}} = \sum_{(i,j)} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) , \tag{13.4}$$

$$E_{\text{hb}} = \sum_{(i,j)} \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) , \tag{13.5}$$

$$E_{\text{tors}} = \sum_l U_l \left(1 \pm \cos(n_l \alpha_l)\right) . \tag{13.6}$$

Here, r_{ij} is the distance between the atoms i and j, and α_l is the torsion angle for the chemical bond l. The parameters ($q_i, A_{ij}, B_{ij}, C_{ij}, D_{ij}, U_l$ and n_l) are calculated from crystal structures of amino acids. Since the bond lengths and bond angles are set constant, the true degrees of freedom are rotations around these bonds. Note that protein energies are measured in kcal/mol, leading to the factor "332" in the electrostatic energy E_{es} term.

Unfortunately, computer simulations are notoriously difficult for such detailed protein models. The complex form of the intramolecular forces and the protein–water interactions makes it extremely difficult to sample low-energy protein conformations. Containing both repulsive and attractive terms, all-atom models of proteins lead to a very rough energy landscape with a huge number of local minima separated by high-energy barriers. For this reason, sampling of low-energy conformations becomes a hard computational task, and physical quantities cannot be calculated accurately from simple low-temperature molecular dynamics or Monte Carlo simulations.

Only recently there has been any progress in alleviating the above stated multiple-minima problem. For a review, see, for instance, Ref. [24]. In the following, I will describe some of these methods that proved to be successful in numerical simulations and in whose development I was involved. While these techniques are of general applicability and not restricted to biological molecules, I will concentrate on their use in protein studies. Especially, I will present some recent applications that illustrate the success and limitations of current protein simulations, and the ongoing need for novel techniques to enhance sampling of low-energy protein structures.

13.2 Energy Landscape Paving

A general characteristic of successful optimization techniques is that they avoid entrapment in local minima and continue to search for further solutions. For instance, in taboo search [9] the search is guided away from areas that have already been explored. However, since such an approach does not distinguish between important and less important regions of the landscape, it can result in slow convergence. Smoothening the energy landscape is another way to avoid entrapment in local minima [7, 22, 48]. In the optimal case, the original energy landscape is transformed in a funnel-landscape, and convergence toward the global minimum is fast. However, most of these landscape-deformation methods require a considerable amount of fine tuning or *a priori* information. Moreover, problems may exist when connecting back to the original landscape since minima on the deformed surface may have been displaced or merged.

Luc Wille (Florida Atlantic University) and I have recently developed a new optimization method, energy landscape paving (ELP) [26], that combines ideas from taboo search and energy landscape deformation and avoids some of the pitfalls of either technique. ELP is of general applicability and proved very promising in protein studies.

In ELP, one performs low-temperature Monte Carlo simulations with an effective energy designed to steer the search away from regions that have already been explored:

$$w(\tilde{E}) = e^{-\tilde{E}/k_B T} \quad \text{with} \quad \tilde{E} = E + f(H(q,t)) \; . \tag{13.7}$$

Here, T is a (low) temperature, $\tilde{E}$ serves as a replacement of the energy E and $f(H(q,t))$ is a function of the histogram $H(q,t)$ in a pre-chosen "order parameter" q. Since the histogram is updated at each MC step, the search process keeps track of the number of prior explorations of a particular region and biases against revisiting the same types of states. Rather than using the system states themselves in the histograms an appropriate order parameter is employed. This may be a "natural" quantity for the system under study or the energy itself.

It follows that the weight of a local minimum state decreases with the time the system stays in that minimum. ELP deforms the energy landscape locally till the local minimum is no longer favored, and the system will explore higher energies. It will then either fall in a new local minimum or walk through this high-energy region till the corresponding histogram entries all have similar frequencies, and the system again has a bias toward low energies. ELP bears some similarities to taboo search [9] in that recently visited regions are not likely to be revisited immediately. ELP is also similar to an energy deformation approach [7] in that the additional histogram parameter leads to a deformation of the energy landscape depending on the frequency with which a particular area has been explored. Since the weight factor is time dependent it follows that ELP violates detailed balance. Hence, the method cannot be used to calculate thermodynamic averages. Note, however, that for $f(H(q,t)) = f(H(q))$ detailed balance is fulfilled, and ELP reduces to the *generalized-ensemble* methods [22] discussed in Section 13.3.2.

I show in the following an algorithmic presentation (in "pidgin Algol") of ELP. For simplicity, I consider only the case when the energy E itself is the order parameter q. However, in many applications of ELP, other parameters q, than the energy E may be a better choice (see, for instance, the simulations of HP-36 described in Section 13.4.2). The ELP simulation goes over $sweeps$ Monte Carlo sweeps, each a sequence of sequential updates of the $variables$ degree of freedom with a (low)temperature T. The algorithm returns the lowest-energy estimate $emin$.

```
**algorithm** ELP($sweeps, variables, T, emin$)
**begin**
  $E :=$ Energy of start configuration
  $H(E) := 0$
  $emin := E$
  **for** $i := 1, 2, \ldots, sweeps$ **do**
    **for** $j := 1, 2, \ldots, variables$ **do**
      $\tilde{x}(j)$ :=a randomly chosen values for variable $x(j)$
      $Enew := E(x(1), x(2), \ldots, \tilde{x}(j), \ldots, x(variables))$
      $w := \min\{1, \exp(-[(Enew + H(Enew)) - (E + H(E))]/k_BT)\}$
      $r :=$ random number from interval $[0, 1]$
      **if** $w \geq r$ **then**
        $x(j) := \tilde{x}(j)$
        $E := Enew$
      **end**
      $H(E) := H(E) + 1$
      $emin := \min(emin, E)$
    **end**
  **end**
**end**
```

The small peptide Met-enkephalin is used to illustrate the search process in ELP [26]. This pentapeptide has the sequence Tyr-Gly-Gly-Phe-Met and is a frequently used benchmark model to examine new algorithms. Its ground state is known for the ECEPP/2 field (see Eq. (13.6)), as implemented in the computer code SMMP [14], and has an energy $E_0 = -10.7$ kcal/mol. Since the next higher local minimum has an energy of $E_1 = -9.8$ kcal/mol [13], one can easily identify any configuration with energy below $E = -9.8$ kcal/mol as a representative of the ground state. As in our algorithmic presentation of ELP we use the potential energy itself as an order parameter. Thus the deformed energy landscape of Met-enkephalin is generated by $\tilde{E} = E + H(E,t)$, where $H(E,t)$ is the histogram in energy at MC sweep t. We chose a bin size $E_{\text{bin}} = 0.25$ kcal/mol in the histogram and set the temperature to $T = 50$ K.

Figure 13.1 illustrates the search process in energy landscape paving. The starting configuration has an energy of $E_{\text{start}} = -5.1$ kcal/mol and was obtained from a random configuration through quenching in an initial 100 sweeps. The simulation soon gets trapped in a local minimum of $E \approx -7.8$ kcal/mol (after only 250 MC sweeps). Though the following MC sweeps entries in the corresponding histogram bin are accumulated and the energy landscape locally deformed, until after about 750 MC sweeps the simulation escapes this local minimum to find a lower local minimum after 2000 MC sweeps. This process is repeated until the simulation finds the global minimum conformation for the first time after 7260 sweeps. Within the 50 000 sweeps of our simulation the ground-state region ($E < -9.8$ kcal/mol) was reached 5 times each time separated by explorations in the high-energy region. Note that the range of energies covered increases with MC time: ELP starts by filling up the small "potholes" in the energy landscape, but fills up also large valleys as the simulation continues.

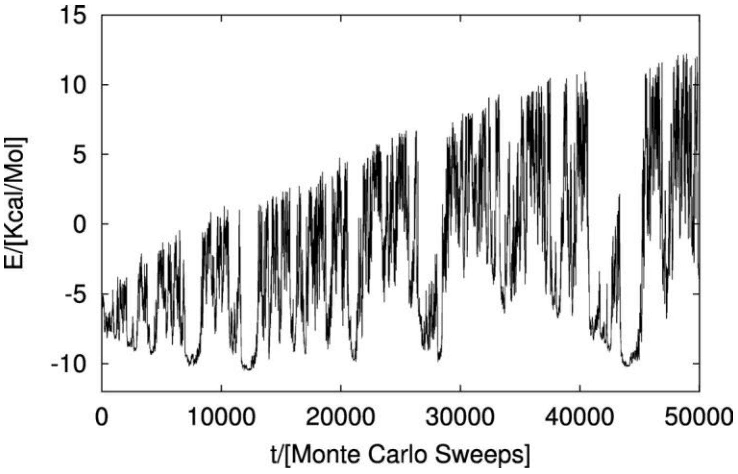


Figure 13.1: "Time series" of energy for ELP simulation of the peptide Met-enkephalin. The figure is taken from Ref. [26].

We have tested the efficiency of ELP by performing 20 independent ELP runs of each 50 000 MC sweeps. The results of the ELP runs are compared with 20 simulated annealing [33] runs of equal statistics using the annealing schedule that proved to be optimal for Met-enkephalin in Ref. [17]. However, even with this optimized annealing schedule, the ground state is found only in $8/20 = 40\%$ of the simulations and the average value of the lowest energy conformation ($\langle E_{\text{min}} \rangle = -8.5$ kcal/mol) is above our threshold for ground state configurations (-9.8 kcal/mol). On the other hand, with ELP we find the ground state in each of the 20 runs. As a consequence, the average of the lowest energy states $\langle E_{\text{min}} \rangle = -10.3$ kcal/mol is well below our threshold for ground-state configuration.

Similar results were found in an application of ELP to the problem of crystal structure prediction from powder X-ray diffraction data [29] where the method was again compared with simulated annealing. For the purpose of ground-state predictions, ELP proved also more efficient than multicanonical sampling (described in Section 13.3.2.1) [4]. Further applications of ELP include the prediction of ground states in Ising spin glasses [27] and of a 3D-structure of the villin headpiece subdomain HP-36, that has a root-mean-square deviation (rmsd) of less than 6 Å to the experimentally determined one [26]. The later application will be discussed in more detail in Section 13.4.2.

13.3 Beyond Global Optimization

Structure prediction by means of global optimization requires the use of an energy function that describes the interactions within a protein and between the protein and the surrounding water. However, neither the available force fields nor the inclusion of solvation effects are perfect. While one expects that the folded structure (as determined by X-ray or NMR experiments) will appear with sufficient probability in the ensemble of low-energy structures, it is not certain that this structure corresponds to the global minimum conformation. Hence, any global optimization approach to structure prediction of proteins is limited by the accuracy of the force fields. Global optimization techniques are also unsuitable for investigations of the structural transitions in proteins that are a key issue for understanding the folding and biological function of a number of proteins. These transitions include, for instance, the change in shape of many proteins when interacting with other molecules, or the appearance of misfolded structures in some proteins, for example, in the Prion protein [44]. As with structure prediction, it is necessary to go beyond global optimization techniques and to measure thermodynamic quantities, i.e., to sample a set of configurations from a canonical ensemble and take an average of the chosen quantity over this ensemble.

13.3.1 Parallel Tempering

As discussed in the introduction, such sampling is hampered for detailed protein representations by the roughness of the energy landscape. One popular method of overcoming the resulting extremely slow thermalization at low temperatures is parallel tempering [30] (also known as replica exchange method or Multiple Markov chains), a techniques that was first applied to protein studies in Ref. [21].

In its most common form, in parallel tempering one considers an artificial system built up of *N non–interacting* replicas of the molecule, each at a different temperature T_i. In addition to standard Monte Carlo or molecular dynamics moves that effect only one copy, parallel tempering introduces a new *global* update [30]: the exchange of conformations between two copies i and $j = i+1$ ($i \geq 1$ and $j \leq N$). This replica exchange move is accepted or rejected according to the Metropolis criterion with probability

$$w(\mathbf{C}^{\text{old}} \to \mathbf{C}^{\text{new}}) = \min\bigl(1, \exp\bigl(-\beta_i E(C_j) - \beta_j E(C_i) + \beta_i E(C_i) + \beta_j E(C_j)\bigr)\bigr). \quad (13.8)$$

Hence, while before the exchange move configuration C_i (C_j) is at the (inverse) temperature β_i, both configurations are exchanged after a successful move and configuration C_i (C_j) is now at temperature β_j (β_i). This exchange of conformations leads to a faster convergence of the Markov chain than in regular canonical simulations, since the resulting random walk in temperatures allows the configurations to move out of local minima and cross energy barriers.

The above description can be easily translated into an algorithmic presentation of parallel tempering: $Sweeps$ is the number of parallel tempering sweeps and $nodes$ is the number of copies of the system with initial sets of temperatures $E(j)$ and $T(j)$. Instead of protein configurations, temperatures $T(j)$ are exchanged between the copies.

```
algorithm Parallel Tempering($sweeps, nodes, E(j), T(j)$)
begin
  for $i := 1, 2, \ldots, sweeps$ do
    for $j := 1, 2, \ldots, nodes$ do
      $E(j)$ := energy of conformation $j$ after canonical Monte Carlo or molecular
          dynamics update at temperature $T(j)$
    end
    for $j := 1, 2, \ldots, nodes - 1$ do
      $k := j + 1$
      $w := \min\left\{1, \exp\left(-\left[\frac{E(j)}{k_B T(k)} + \frac{E(k)}{k_B T(j)}\right] + \left[\frac{E(j)}{k_B T(j)} + \frac{E(k)}{k_b T(k)}\right]\right)\right\}$
      $r$ := random number from interval $[0, 1]$
      if $w \geq r$ then
        $save := T(j)$
        $T(j) := T(k)$
        $T(k) := save$
      end
    end
  end
end
```

Expectation values of a physical quantity A are calculated as usual according to:

$$\langle A \rangle_{T_i} = \frac{1}{MES} \sum_{k}^{MES} A(C_i(k)) \, , \tag{13.9}$$

where MES is the number of measurements taken for the ith temperature. Values for intermediate temperatures are calculated using multihistogram reweighting techniques [15]. The temperature distribution should be chosen such that the exchange move has a 30% acceptance rate and the highest temperature is such that any relevant energy barrier can be crossed. Note that parallel tempering does not require Boltzmann weights. The method can be combined easily with *generalized-ensemble* techniques [21] (see Section 13.3.2).

Met-enkephalin is used again to illustrate the parallel tempering algorithm. Simulations with seven copies were performed [21]. The corresponding temperatures are $T_1 = 1000$ K, $T_2 = 500$ K, $T_3 = 330$ K, $T_4 = 250$ K, $T_5 = 170$ K, $T_6 = 100$ K and $T_7 = 50$ K. The simulation consists of 144 000 sweeps for each copy. After each sweep, an exchange of conformations between pairs of copies at neighboring temperatures was tried. The "time series" of temperatures for one of the seven copies is shown in Figure 13.2. Due to the exchange

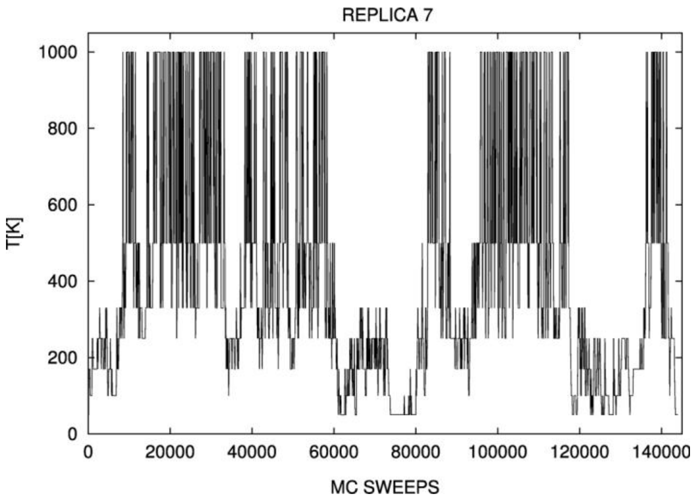


Figure 13.2: "Time series" of temperature for one copy of Met-enkephalin over 144 000 Monte Carlo sweeps as obtained from a parallel tempering simulation. The figure is taken from Ref. [21].

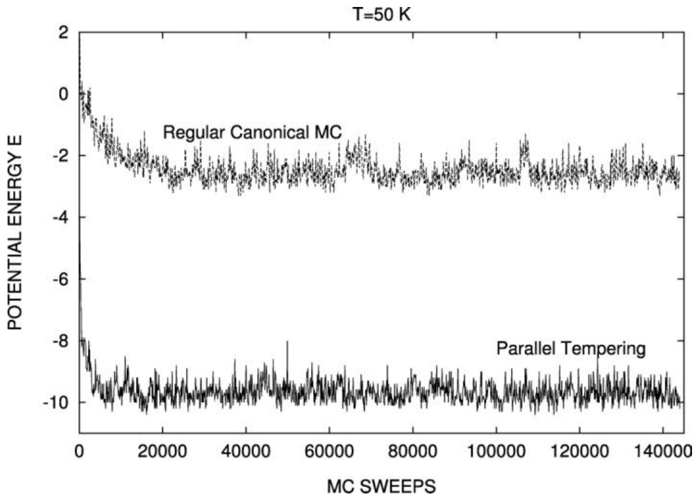


Figure 13.3: "Times series" of energy at temperature $T = 50$ K as obtained from the parallel tempering algorithm and a regular canonical simulation. The figure is taken from Ref. [21].

move, the configuration walks randomly between low temperatures and high temperatures. The resulting random walk in energy ensures – as in the case of ELP – that any energy barrier can be overcome, and the molecule will thermalize at all seven temperatures. The faster convergence can be seen in Figure 13.3 where the "time series" in energy is displayed for both a regular canonical simulation at $T = 50$ K and for the copy with $T = 50$ K of a parallel tempering simulation. Obviously the regular canonical Monte Carlo was trapped in a local minimum and was not able to thermalize. From previous simulations (see Ref. [19]) it is known that even $1000\,000$ sweeps are not enough to thermalize Met-enkephalin at $T = 50$ K. On the other hand, with the exchange of configurations by parallel tempering, the simulation thermalizes at $T = 50$ K in less than $10\,000$ sweeps.

13.3.2 Multicanonical Sampling and Other Generalized-ensemble Techniques

Generalized-ensemble simulations [22] offer another possibility of overcoming the multiple minima problem and of calculating reliable low-temperature quantities. The idea is again to ensure that a simulation does not get trapped in local minima but samples both low and high-energy states with sufficient probability. Such movement in and out of local minima is obtained by requiring that a Monte Carlo or molecular dynamics simulation shall lead to a uniform distribution of a pre-chosen physical quantity. For simplicity we will restrict ourselves to ensembles that lead to flat distributions in only *one* variable. Extensions to higher dimensional generalized ensembles are straightforward [28, 34] as are combinations with annealing techniques [17, 38].

Probably the earliest realization of the generalized-ensemble idea is *umbrella sampling* [45], but it has been lately revived in various forms such as multicanonical sampling [5], simulated tempering [36], etc. The first application of these new techniques to protein simulations can be found in Ref. [16] where a Monte Carlo technique was used. Later, a formulation for the molecular dynamics method was also developed [18].

13.3.2.1 Multicanonical Sampling

In the *multicanonical algorithm* [5] configurations with energy E are assigned a weight $w(E)$ such that the distribution of energies

$$P_{mu}(E) \propto n(E) w_{mu}(E) = \text{const}, \tag{13.10}$$

where $n(E)$ is the spectral density. Since all energies appear with the equal probability, a free random walk in the energy space is enforced: the simulation can overcome *any* energy barrier and will not get trapped in one of the many local minima. In order to demonstrate the latter point the "time series" of energy is shown in Figure 13.4 as a function of Monte Carlo sweeps for both a regular canonical Monte Carlo simulation at temperature $T = 50$ K (dotted curve) and a multicanonical simulation. The displayed data are again from a simulation of the pentapeptide Met-enkephalin using a slightly modified version [17] of the ECEPP/2 force field. Starting from a random configuration the two simulations continued for $1000\,000$ Monte Carlo sweeps. For the canonical run the curve stays around the value $E = -7$ kcal/mol with small thermal fluctuations, reflecting the low-temperature nature. The run has apparently been trapped in a local minimum, since the mean energy at this temperature is $\langle E \rangle = -11.1$ kcal/mol as found in Ref. [17]. On the other hand, the multicanonical simulation covers a much wider energy range than the canonical run. It is a random walk in energy space, which keeps the simulation from getting trapped in a local minimum.

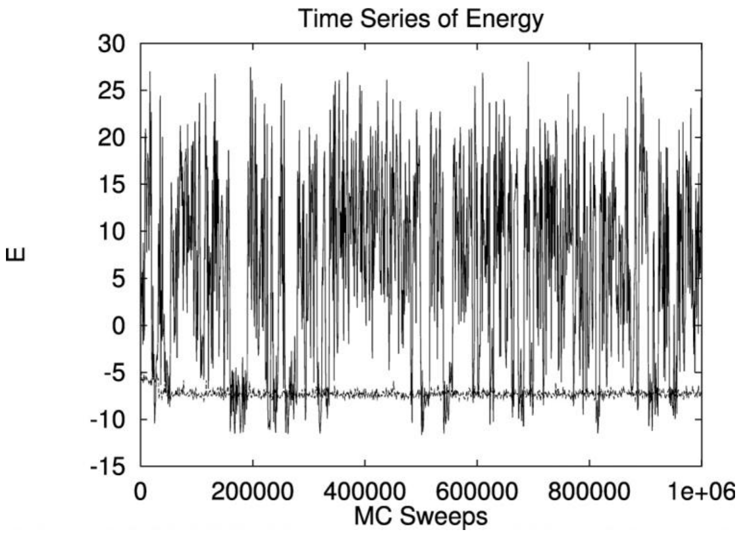


Figure 13.4: "Time series" of energy for the pentapeptide Met-enkephalin. Both the results from a canonical simulation at $T = 50$ K (dotted line) and a multicanonical simulation are shown.

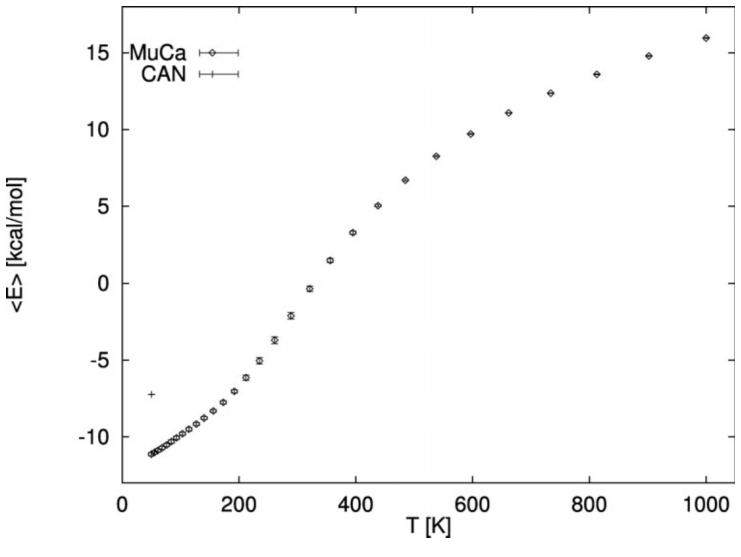


Figure 13.5: The average energy $\langle E \rangle$ of the pentapeptide Met-enkephalin as a function of temperature T.

From such a multicanonical simulation one cannot only locate the energy global minimum, but can also calculate the expectation value of any physical quantity $\mathcal{O}$ at temperature T by re-weighting techniques [15]

$$\langle \mathcal{O} \rangle_T = \frac{\int dE\ \mathcal{O}(E) P_{mu}(E)\ w_{mu}^{-1}(E)\ e^{-E/k_B T}}{\int dE\ P_{mu}(E)\ w_{mu}^{-1}(E)\ e^{-E/k_B T}} \tag{13.11}$$

$$= \frac{\int dx\ \mathcal{O}(x)\ w_{mu}^{-1}(E(x))\ e^{-\beta E(x)}}{\int dx\ w_{mu}^{-1}(E(x))\ e^{-\beta E(x)}} \tag{13.12}$$

where x stands for configurations.

The average (ECEPP/2) energy of Met-enkephalin as a function of temperature is shown as an example in Figure 13.5. Thermodynamic averages are calculated by Eq. (13.12) from the same data as displayed in Figure 13.4 over a temperature range of 1000 K to below 100 K where canonical simulations failed (see the value for the average energy at $T = 50$ K as obtained by a canonical simulation of 1000 000 MC sweeps).

It has to be noted that, unlike in the canonical ensemble, the weights $w_{mu}(E) \propto n^{-1}(E)$ are not *a priori* known (in fact, knowledge of the exact weights is equivalent to obtaining the density of states $n(E)$, i.e., solving the system) and one needs their estimates for a numerical simulation. Hence, multicanonical sampling consists of three steps:

```
algorithm Multicanonical Sampling
begin
    Calculate Estimators for multicanonical weights
    Do Simulation with these weights
    Calculate physical quantities at desired temperatures by reweighting
end
```

Calculation of the multicanonical (and other generalized-ensemble weights) is usually done by an iterative procedure [16, 17]. The following algorithmic presentation describes a simple version of this procedure. In it, one uses the fact that the histogram of a multicanonical simulation can be written as $H(E) = n(E)w_{mu}^{i}(E)$ where $w_{mu}^{i}(E)$ is the ith estimate of the canonical weight. Setting $w_{mu} = 1/n(E)$, one obtains the iterative relation $w_{mu}^{i+1} = w_{mu}^{i}(E)/H(E)$. $Iter$ is the number of iterative improvements of the weights $w_{mu}(i)$, $sweeps$ is the number of Monte Carlo sweeps in each cycle, and $nbin$ is the number of energy bins. We remark that calculation of the weights can be slow (about 40% of the total CPU time was spent in Ref. [16] on this point) and several attempts were made to obtain generalized-ensemble weights in a faster way; see, for instance, Refs. [20, 46].

```
algorithm Multicanonical Weights($iter, sweeps, nbin, w_{mu}(i)$)
begin
    for $i := 1, 2, \ldots, nbin$ do
        $w(i) = 1$
    end
    for $i := 1, 2, \ldots, iter$ do
        for $k := 1, 2, \ldots, nbin$ do
            $H(k) = 0$
        end
        for $j := 1, 2, \ldots, sweeps$ do
            $E :=$ energy after Monte Carlo update with $w(i)$
            $k :=$ index of energy bin that contains $E$
            $H(k) := H(k) + 1$
        end
        for $k := 1, 2, \ldots, nbin$ do
            $H(k) := \max(1, H(k))$
            $w_{mu}(k) = w_{mu}(k)/H(k)$
        end
    end
end
```

13.3.2.2 Other Generalized-ensemble Techniques

In multicanonical simulations, the computational effort increases with the number of residues like $\approx N^4$ (when measured in Metropolis updates) [23]. In general, the computational effort in simulations increases with $\approx X^2$ where X is the variable in which one wants a flat distribution. This is because generalized-ensemble simulations realize, by construction of the ensemble, a 1D random walk in the chosen quantity X. In the multicanonical algorithm the reaction coordinate X is the potential energy $X = E$. Since $E \propto N^2$ the above scaling relation for the computational effort $\approx N^4$ is recovered. Hence, multicanonical sampling is not always the optimal generalized-ensemble algorithm in protein simulations. A better scaling of the computer time with size of the molecule may be obtained by choosing a more appropriate reaction coordinate for our ensemble than the energy.

One often-used choice is *simulated tempering* [36] where the temperature itself becomes a dynamic variable and is sampled uniformly. Temperature and configuration are both updated with a weight:

$$w_{ST}(T, E) = e^{-E/k_B T - g(T)} \; . \tag{13.13}$$

Here, the function $g(T)$ is chosen so that the probability distribution of temperature is given by

$$P_{ST}(T) = \int dE \; n(E) \; e^{-E/k_B T - g(T)} = \text{const} \; . \tag{13.14}$$

Physical quantities have to be sampled for each temperature point separately and expectation values at intermediate temperatures are calculated by re-weighting techniques [15].

As is common in generalized-ensemble simulations, the weight $w_{ST}(T, E)$ is not *a priori* known (since it requires knowledge of the parameters $g(T)$) and their estimator has to be calculated. They can again be obtained by an iterative procedure as described in Section 13.3.2.1. In the simplest version, the improved estimator for $g^{(i)}(T)$ for the ith iteration is calculated from the histogram of *temperature* distribution $H_{ST}^{(i-1)}(T)$ of the preceding simulation as follows:

$$g^{(i)}(T) = g^{(i-1)}(T) + \log H_{ST}^{(i-1)}(T) \; . \tag{13.15}$$

In this procedure one uses the fact that the histogram of the ith iteration is given by

$$H_{ST}(T) = e^{-g_{i-1}(T)} Z_i(T) \,, \tag{13.16}$$

where $Z_i(T) = \int dE n(E) \exp(-E/k_B T)$ is an estimate for the canonical partition function at temperature T. Setting $\exp(g_i(T)) = Z_i(T)$ leads to the iterative relationship of Eq. (13.15).

Various other realizations of the generalized-ensemble approach exist. All aim at sampling a broad range of energies. This is because in protein simulations we are interested not only in sampling the low-energy region but also in visiting high-energy states with finite probability. In this way the simulation will overcome energy barriers and allow escape from local minima. Yuko Okamoto (Institute for Molecular Science, Japan) and I have proposed in Ref. [19] for this purpose yet another generalized ensemble where configurations are updated according to the following probability weight:

$$w(E) = \left(1 + \frac{\beta(E - E_0)}{n_F}\right)^{-n_F} \,. \tag{13.17}$$

Here E_0 is an estimator for the ground-state energy, n_F is the number of degrees of freedom of the system, and $\beta = 1/k_B T$ is the inverse temperature with a low-temperature T. Note that this weight can be understood as a special case of the weights used in Tsallis generalized mechanics formalism [10] (the Tsallis parameter q is chosen as $q = 1 + 1/n_F$). The weight reduces in the low-energy region to the canonical Boltzmann weight $\exp(-\beta E)$. This is because $E - E_0 \to 0$ for $\beta \to 0$ leading to $\beta(E - E_0)/n_F \ll 1$. On the other hand, high-energy regions are no longer exponentially suppressed but only according to a power law, which enhances excursions to high-energy regions.

Recently, another ensemble (stochastic tunneling [48]) with similar properties was proposed where conformations enter with a weight $w(E) = \exp(f(E)/k_B T)$. Here, $f(E)$ is a non-linear transformation of the potential energy onto the interval $[0, 1]$ and T is a low temperature. The physical idea behind such an approach is to allow the system to "*tunnel*" through energy barriers in the potential energy surface [48]. Such a transformation can be realized by

$$f(E) = e^{-(E-E_0)/n_F} \,, \tag{13.18}$$

where E_0 is again an estimate for the ground state and n_F the number of degrees of freedom of the system. Note that the location of all minima is preserved. Hence, at a given low-temperature T, the simulation can pass through energy barriers of arbitrary height, while the low-energy region is still well resolved. An exploratory study on the efficiency of this algorithm for protein-folding simulations can be found in Ref. [25].

In both ensembles a broad range of energies is sampled. Hence, one can again use reweighting techniques [15] to calculate thermodynamic quantities over a large range of temperatures. In contrast to other generalized-ensemble techniques, the weights are explicitly given for both new ensembles. One needs only to find an estimator for the ground-state energy E_0 which is easier than the determination of weights for other generalized ensembles.

13.4 Results

In the following, I will present two examples that demonstrate how these novel simulation techniques can be used to explore the physics of folding and to predict the biologically active state of proteins.

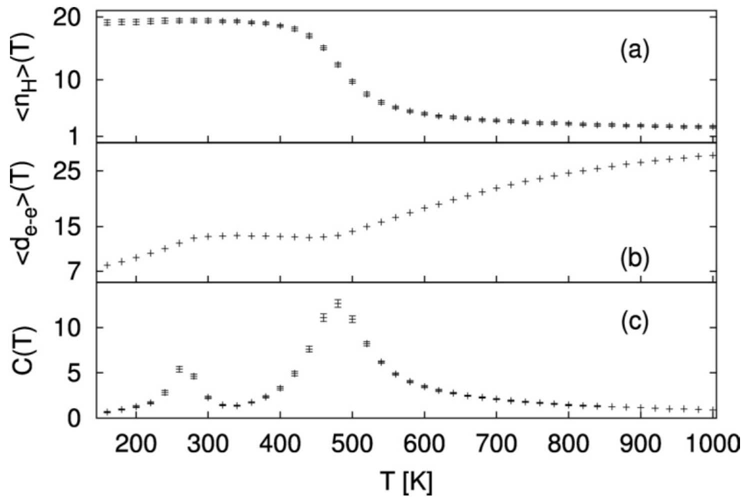


Figure 13.6: (a) Average number of helical residues $\langle n_H \rangle(T)$, (b) average end-to-end distance $\langle d_{e-e} \rangle(T)$ and (c) specific heat $C(T)$ as a function of temperature as calculated from simulations of Ala$_{10}$-Gly$_5$-Ala$_{10}$.

13.4.1 Helix Formation and Folding

I start with a recent investigation into the relation between helix formation and folding. Nelson Alves (FFCLRP, University of Sao Paulo, Brazil) and I have studied the artificial peptide Ala$_{10}$-Gly$_5$-Ala$_{10}$ in the gas phase by means of multicanonical simulations [1]. Estimators for the weights were determined by an iterative procedure [6, 41] in 500 000 sweeps. All thermodynamic quantities were then estimated from one production run of 8000 000 Monte Carlo sweeps which followed 10 000 sweeps for thermalization.

Our peptide, Ala$_{10}$-Gly$_5$-Ala$_{10}$, is build up out of two chains of each 10 alanine residues connected by five glycine residues. We have shown in previous work [2, 23, 38] that polyalanine has a pronounced helix-coil transition. For this reason, we have measured the average number of helical residues $\langle n_H \rangle$ and displayed in Figure 13.6(a). Two temperature regions are observed. At high temperature, few residues are found that are part of an α-helix. On the other hand, at low temperatures we observe helix formation, and almost all of the alanine residues are part of an α-helix. The transition between the two temperature regions is sharp indicating the existence of a helix-coil transition. The transition temperature T_{hc} can be determined from the corresponding peak in the specific heat (Figure 13.6(c)) at a temperature $T = 483 \pm 8$ K. However, we find in Figure 13.6(c) in addition, a second, smaller peak at a lower temperature

$T_f = 265 \pm 7$ K indicating yet another transition. The meaning of this second peak becomes clear from Figure 13.6(b) where we plot the the average end-to-end distance $\langle d_{e-e} \rangle_T$ as a function of temperature. This quantity is a measure for the compactness of a protein conformation and defined here by the distance between N of Ala_1 and O of Ala_{25}. We observe that this quantity decreases with decreasing temperature. Below the helix-coil transition T_{hc} the decrease slows down and the curve becomes almost flat at a value of $\langle d_{e-e} \rangle \approx 10$ Å indicating that there is little further change in the compactness of the molecule. However, at temperature T_f the end-to-end distance decreases again sharply toward a new value $\langle d_{e-e} \rangle = 6.1$ Å. Hence, T_f marks the folding of the molecule into a defined compact structure with the two terminal ends of the peptide close together. This scenario is supported by Figure 13.7 in which we display the configuration with lowest energy ever found in our multicanonical simulation of 8000 000 sweeps. It consists of two helices (made up out of the alanine residues) connected by a turn (built out of the flexible glycine residues) toward a U-turn-like structure that is consistent with the small value of the end-to-end distance d_{e-e} observed in Figure 13.6(b) below T_f.

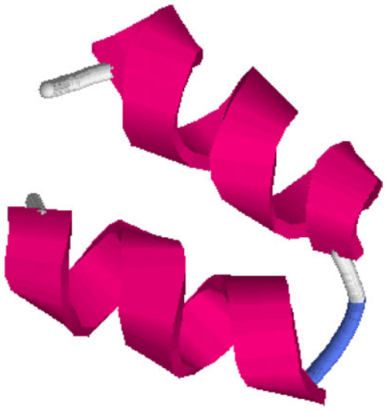


Figure 13.7: Lowest energy configuration of Ala_{10}-Gly_5-Ala_{10} as obtained from multicanonical simulations in the gas phase.

Our above analysis of the thermodynamics of our peptide suggests that Ala_{10}-Gly_5-Ala_{10} folds in a two-step process. The first step is the formation of α-helices and can be characterized by a helix-coil transition temperature $T_{hc} = 483 \pm 8$ K. The formation of α-helices then restricts the possible configuration space. Energetically most favorable is the folding of two α-helices (made out of the alanine residues) into a hairpin. This second step can be characterized by a lower folding temperature $T_f = 265 \pm 7$ K. Note that this folding temperature is in the biologically relevant temperature regime while helix-formation can also happen at much higher temperatures. The above described two-step folding of our artificial peptide is reminiscent of the framework model [32, 42] and collision-diffusion model [31] which propose that local elements of native local secondary structure, form independently of tertiary structure. As in our example, these elements diffuse until they collide and coalesce to give a tertiary structure.

13.4.2 Structure Predictions of Small Proteins

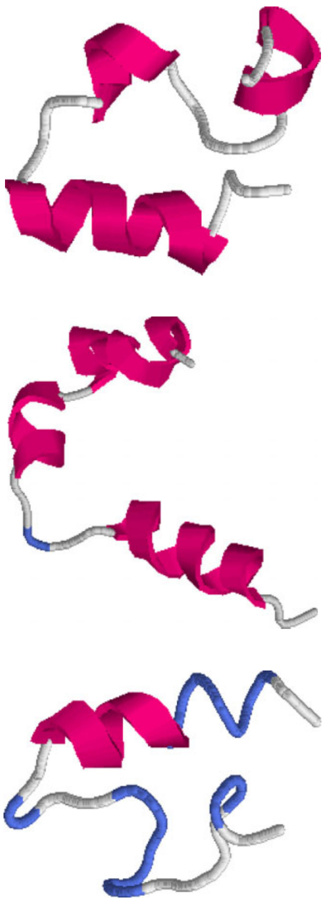


Figure 13.8: Top: Experimental structure of HP-36 as deposited in the PDB data-bank. Middle: Lowest energy structure as obtained in a simulation of the solvated peptide. Bottom: Lowest energy structure of HP-36 as obtained in a simulation in the gas phase.

The second example is the 36-residue villin headpiece subdomain HP-36, one of the smallest peptides that can fold autonomously. HP-36 was chosen by Duan and Kollman for a 1-microsecond molecular dynamics simulation of protein folding [12]. The experimental structure was determined by NMR analysis [37]. Luc Wille (Florida Atlantic University) and I have used this protein to study the efficiency of the ELP algorithm. We have used the approach of Eq. (13.1) to approximate the interaction between protein and water with the parameters σ_i chosen from Ref. [49].

Built up only out of α-helices as secondary structure elements, HP-36 allows in a simple way the definition of an order parameter to characterize configurations other than by their energy. This natural order parameter is the number n_H of residues in the peptide which are part of an α-helix. Throughout the search process we try to deform the energy landscape by means of a histogram $H(E, n_H, t)$ in *both* helicity and energy: $\tilde{E} = E + H(E, n_H, t)$. Operating again at a temperature $T = 50$ K, we find as weights for the search algorithm

$$w(E, n_H, t) = e^{-\beta(E+H(E,n_H,t))} \, . \tag{13.19}$$

Using this weight we performed simulations with 50 000 MC sweeps (starting from random configurations) keeping track of the lowest energy configuration during the search process.

The structure of HP-36 as obtained from the Protein Data Bank (PDB code 1vii) is shown in Figure 13.8. The structure consists of three helices between residues 4–8, 15–18, and 23–32, respectively, which are connected by a loop and a turn. We find for this structure in our model an energy (ECEPP/2 + solvation term) $E_{\text{nat}} = -276$ kcal/mol. Our approach led to a configuration with the lowest energy $E_{\text{min}} = -277$ kcal/mol which we show also in Figure 13.8 [26]. The above structure consists of three helices where the first helix stretches from residue 2 to residue 11 and is more elongated than the corresponding one in the native structure (residues 4–8). The second helix consist of residues 13–17 (compared to residue 15–18 in the native structure) and the third helix stretches from residue 23–33 (residues 23–32 in the PDB structure). The structure has 95% of the native helical content and a radius of gyration $R_\gamma = 10.1$ Å which indicates that the numerically obtained structure is slightly less compact than the experimental structure ($R_\gamma = 9.6$ Å). 60% of native contacts are formed. These values are comparable with the results in Ref. [12] (but required orders of magnitude less computer time) where the optimal structure of a 1 μs molecular dynamic folding simulation showed 80% of native helical content and 62% of native contacts. Similarly comparable were the values of the root-mean-square deviation (RMSD) of both numerically determined conformers to the native structure: 5.8 Å versus 5.7 Å in Ref. [12] (counting only backbone atoms). On the other hand, an ELP simulation of 50 000 sweeps relying only on the ECEPP/2 force field led to a structure with an ECEPP energy of $E_{GP} = -192$ kcal/mol. That structure, shown in the bottom of Figure 13.8, is build out of two helices (between residues 2-16 and 23-33) connected by a loop, differs significantly from the regularized PDB structure with the higher potential energy $E_{\text{nat}} = -176$ kcal/mol. Hence, the native structure of the peptide HP-36 is *not* the global minimum configuration in ECEPP/2. Only the inclusion of the solvation term led to an essentially correct structure as global minimum configuration.

In order to understand more the differences between the gas-phase results and that with a solvent accessible surface term, Chai-Yu Lin, Chin-Ku Hu (both Academia Sinica, Taiwan) and I have simulated recently HP-36 with parallel tempering on 20 nodes of a cluster of IBM 4-ways 375MHZ SMP Thin Nodes [35]. We have chosen as temperatures $T = 1000$, 900, 800, 700, 610, 560, 530, 510, 495, 485, 475, 465, 450, 420, 390, 360, 330, 300, 275, 250 K. On each node, we performed 150 000 MC sweeps, and a replica exchange move was attempted after each sweep. Both gas-phase simulations and such relying on a solvent-accessible surface term with the parameter set OONS of Ref. [40] were performed.

From these parallel tempering simulations we have calculated the number of helical residues as a function of temperature. Figure 13.9 displays our results. Little difference is found at high temperatures. However, below the transition temperature $T \approx 490$ K, the data for both simulations diverge. The helicity grows rapidly with decreasing temperature in the OONS simulation while it stays small in the gas phase. Configurations in the gas phase and in OONS simulations differ also in their compactness. We display in Figure 13.10, for HP-36, two quantities that measure the compactness of protein configurations. The main graph is a plot of the average radius of gyration $\langle r_{gy}\rangle(T)$ as a function of temperature. The corresponding values for the total number of contacts $\langle n_{TC}(T)\rangle$ are shown in the inset. Both plots indicate that configurations in the gas phase are substantially more compact than the ones in the OONS simulation. For instance, at $T = 300$ K, we find $r_{rgy} = 9.6(1)$ Å in the gas phase compared to $r_{rgy} = 12.5(1)$ Å in OONS simulations. Note that, even at $T = 1000$ K, the peptide in the gas phase has a radius of gyration $r_{gy} = 15.6(1)$ Å and is substantially more compact than in the OONS simulation ($r_{gy} = 19.2$ Å). We conjecture that this bias toward compact configurations inhibits the formation of α-helices, and that the low-energy states of HP-36 in the gas phase are characterized by large density and low helicity.

Our simulations of HP-36 demonstrate that the simulation techniques described in this review allow one, not only to predict the structure of small peptides, but also to evaluate the limitations of the utilized energy functions. For instance, in our example, we were able to determine the reasons behind the failure of gas-phase simulations when compared to those with simple solvent approximations. Since presently available energy functions are often parameterized for small molecules, their limitations may become more obvious as one proceeds toward larger systems. The described modern simulation techniques may open ways to unveil and finally overcome these limitations.

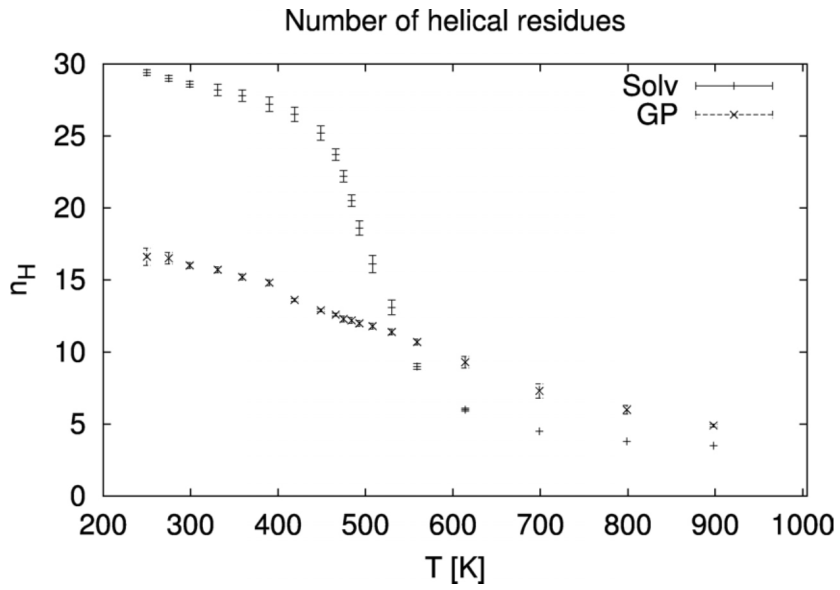


Figure 13.9: Average number of helical residues $\langle n_H\rangle(T)$ of HP-36 as a function of temperature for both the solvated protein and in the gas phase. The figure is taken from Ref. [35].

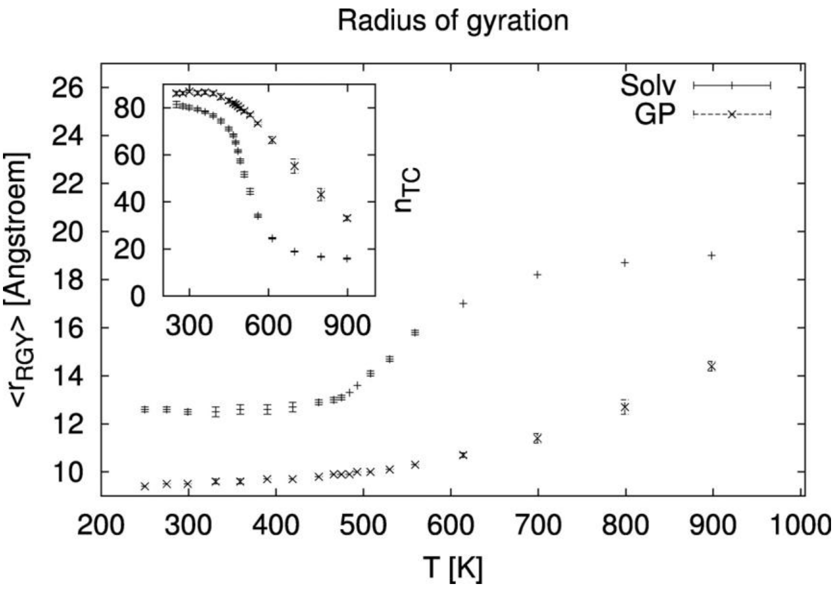


Figure 13.10: Average radius of gyration $\langle r_{gy} \rangle (T)$ of HP-36 as a function of temperature for both the solvated protein and in the gas phase. The figure is taken from Ref. [35].

13.5 Conclusion

I gave a brief introduction into a few modern techniques used in simulations of the protein-folding problem. These examples demonstrate that modern simulation algorithms are well-suited for investigations both of the thermodynamics of proteins and the prediction of their structure. It seems now that all-atom simulations and structure predictions of large proteins are more restricted by the accuracy of the present energy functions than by the efficiency of the search algorithms.

Acknowledgments

The presented work was done in collaboration with Nelson Alves (FFCLRP, University of Sao Paulo, Brazil), Luc Wille (Florida Atlantic University), Chai-Yu Lin and Chin-Ku Hu (both, Academia Sinica, Taiwan). Financial supports from a research grant (CHE-9981874) of the National Science Foundation (USA) is gratefully acknowledged.

References

[1] N.A. Alves and U.H.E. Hansmann, *Helix Formation and Folding in an Artificial Peptide*, J. Chem. Phys. **118**, 2374 (2003).

[2] N.A. Alves and U.H.E. Hansmann, *Partition Function Zeros and Finite Size Scaling of Helix-Coil Transitions in a Polypeptide*, Phys. Rev. Lett. **84**, 1836 (2000).

[3] C.B. Anfinsen, *Principles that Govern the Folding of Protein Chains*, Science **181**, 223 (1973).

[4] H. Arkin and T. Celik *Comparison of the ELP and Multicanonical Methods in Simulations of the Heptapeptide Deltorphin*, Eur. Phys. J. B **30**, 577 (2002).

[5] B.A. Berg and T. Neuhaus, *Multicanonical Algorithms for First Order Phase Transitions*, Phys. Lett. B **267**, 249 (1991).

[6] B.A. Berg, *Algorithmic Aspects of Multicanonical Simulations*, J. Stat. Phys. **82**, 323 (1996).

[7] G. Besold, J. Risbo, and O. G. Mouritsen, *Efficient Monte Carlo Sampling by Direct Flattening of Free Energy Barriers*, Comp. Mat. Sci. **15**, 311 (1999).

[8] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus, *CHARMM: A Program for Macromolecular Energy, Minimization and Dynamics Calculations*, J. Comp. Chem. **4**, 187 (1983).

[9] D. Cvijovic and J. Klinowski, *Taboo Search: An Approach to the Multiple Minima Problem*, Science **267**, 664 (1995).

[10] E.M.F. Curado and C. Tsallis, *Possible Generalization of Boltzmann-Gibbs Statistics*, J. Phys. A: Math. Gen. **27**, 3663 (1994).

[11] K.A. Dill and H.S. Chan, *From Levinthal to Pathways to Funnels*, Nature Structural Biology **4**, 10 (1997).

[12] Y. Duan and P.A. Kollman, *Pathways to a Protein Folding Intermediate Observed in a 1-microsecond Simulation in Aqueous Solution*, Science **282**, 740 (1998).

[13] F. Eisenmenger and U.H.E. Hansmann, *Variation of the Energy Landscape of a Small Peptide under a Change from the ECEPP/2 Force Field to ECEPP/3*, J. Phys. Chem. B **101**, 3304 (1997).

[14] F. Eisenmenger, U.H.E. Hansmann, Sh. Hayryan, C.-K. Hu, *[SMMP] A Modern Package for Simulation of Proteins*, Comp. Phys. Comm. **138**, 192 (2001).

[15] A.M. Ferrenberg and R.H. Swendsen, *New Monte Carlo Technique for Studying Phase Transitions*, Phys. Rev. Lett. **61**, 2635 (1988); Optimized Monte Carlo Data Analysis, Phys. Rev. Lett. **63** , 1658(E) (1989), and references given in the erratum.

[16] U.H.E. Hansmann and Y. Okamoto, *Prediction of Peptide Conformation by Multicanonical Algorithm: A New Approach to the Multiple-Minima Problem*, J. Comp. Chem. **14**, 1333 (1993).

[17] U.H.E. Hansmann and Y. Okamoto, *Comparative Study of Multicanonical and Simulated Annealing Algorithms in the Protein Folding Problem*, Physica A **212**, 415 (1994).

[18] U.H.E. Hansmann, Y. Okamoto, and F. Eisenmenger, *Molecular Dynamics, Langevin and Hybrid Monte Carlo Simulations in a Multicanonical Ensemble*, Chem. Phys. Lett. **259**, 321 (1996).

[19] U.H.E. Hansmann and Y. Okamoto, *Generalized-Ensemble Monte Carlo Method for Systems with Rough Energy Landscape*, Phys. Rev. E **56**, 2228 (1997).

[20] U.H.E. Hansmann, *An Effective Way for Determination of Multicanonical Weights*, Phys. Rev. E **56**, 6200 (1997).

[21] U.H.E. Hansmann, *Parallel Tempering Algorithm for Conformational Studies of Biological Molecules*, Chem. Phys. Lett. **281**, 140 (1997).

[22] U.H.E. Hansmann and Y. Okamoto, *The Generalized-Ensemble Approach for Protein Folding Simulations*, In: D. Stauffer (ed.), *Annual Reviews in Computational Physics VI*, 129 (World Scientific, Singapore, 1999).

[23] U.H.E. Hansmann and Y. Okamoto, *Finite-size Scaling of Helix-coil Transitions in Polyalanine Studied by Multicanonical Simulations*, J. Chem. Phys. **110**, 1267 (1999); **111**, 1339(E) (1999).

[24] U.H.E. Hansmann and Y. Okamoto, *New Monte Carlo Algorithms for Protein Folding*, Curr. Opin. Struct. Biol. **9**, 177 (1999).

[25] U.H.E. Hansmann, *Protein Folding Simulations in a Deformed Energy Landscape*, Eur. Phys. J. B **12**, 607 (1999).

[26] U.H.E. Hansmann and L.T. Wille, *Global Optimization by Energy Landscape Paving*, Phys. Rev. Lett. **88**, 068105 (2002).

[27] Alex Hansen, private communication.

[28] J. Higo, N. Nakajima, H. Shirai, A. Kidera, and H. Nakamura, *Two-component Multicanonical Monte Carlo Method for Effective Conformational Sampling.* J. Comp. Chem. **18**, 2086 (1997).

[29] H.P. Hsu, S.C. Lin and U.H.E. Hansmann, *Energy Landscape Paving for X-Ray Structure Prediction of Macromolecules*, Acta Cryst. A **58**, 259 (2002).

[30] K. Hukushima and K. Nemoto, *Exchange Monte Carlo Method and Applications to Spin Glass Simulations*, J. Phys. Soc. (Jpn.) **65**, 1604 (1996); G.J. Geyer, *Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference*, J. Am. Stat. Assn **90 (431)**, 909 (1995)

[31] M. Karplus and D.L. Weaver, *Protein-folding Dynamics - The Diffusion-Collision Model and Experimental Data*, Protein Sci. **3**, 650 (1994).

[32] P.D. Kim and R.L. Baldwin, *Intermediates in the Folding Reactions of small Proteins.*, Ann. Rev. Biochem. **59**, 631 (1990).

[33] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, *Optimization by Simulated Annealing*, Science **220**, 671 (1983).

[34] S. Kumar, P.W. Payne, and M. Vásquez, *Method for Free-energy Calculations using Iterative Techniques.* J. Comp. Chem. **17**, 1269 (1996).

[35] C.-Y. Lin, C.-K. Hu and U.H.E. Hansmann, *Parallel Tempering Simulations of HP-36*, Proteins **52**, 436 (2003).

[36] A.P. Lyubartsev, A.A. Martinovski, S.V. Shevkunov, P.N. Vorontsov-Velyaminov, *New Approach to Monte Carlo Calculations of the Free Energy: Method of Expanded Ensembles*, J. Chem. Phys., **96**, 1776 (1992); E. Marinari, G. Parisi, *Simulated Tempering: A new Monte Carlo Scheme*, Europhys. Lett., **19**, 451 (1992).

[37] C.J. McKnight, D.S. Doehring, P.T. Matsudaria and P.S. Kim, *A Thermostable 35-Residue Subdomain within Villin Headpiece*, J. Mol. Biol. **260**, 126 (1996).

[38] Y. Okamoto and U.H.E. Hansmann, *Thermodynamics of Helix - Coil Transitions Studied by Multicanonical Algorithms*, J. Phys. Chem. **99**, 11276 (1995).

[39] J.N. Onuchic, Z. Luthey-Schulten and P.G. Wolynes, *Theory of protein folding: the energy landscape perspective*, Ann. Rev. Phys. Chem. **48**, 545 (1997).

[40] T. Ooi, M. Obatake, G. Nemethy, and H.A. Scheraga, *Accessible Surface Areas as a Measure of the Thermodynamic Parameters of Hydration of Peptides*, Proc. Natl. Acad. Sci. USA **8**, 3086 (1987).

[41] Y. Peng and U.H.E. Hansmann, *Solvation Model Dependency of Helix-coil Transition in Polyalanine*, Biophys. J. **82**, 3269 (2002).

[42] O.B. Ptitsyn, *Kinetic and Equilibrium Intermediates in Protein Folding*, Protein Eng. **7**, 593 (1994).

[43] M.J. Sippl, G. Némethy, and H.A. Scheraga, *Intermolecular Potentials from Crystal Data. 6. Determination of Empirical Potentials for O-H$\cdots$O=C Hydrogen Bonds from Packing Configurations*, J. Phys. Chem. **88**, 6231 (1984), and references therein.

[44] G. Taubes, *Misfolding the Way to Diseases*, Science **271**, 1493 (1996).

[45] G.M. Torrie and J.P. Valleau, *Nonphysical Sampling Distributions in Monte Carlo Free-Energy Estimation: Umbrella Sampling*, J. Comp. Phys. **23**, 187 (1977).

[46] F. Wang and D.P. Landau, *Efficient, Multiple-range Random Walk Algorithm to Calculate the Density of States*, Phys. Rev. Lett. **86**, 2050 (2001).

[47] S.J. Weiner, P.A. Kollman, D.T. Nguyen, and D.A. Case, *An All-atom Force Field for Simulation of Proteins and Nucleic Acids*, J. Comp. Chem. **7**, 230 (1986).

[48] W. Wenzel and K. Hamacher, *Stochastic Tunneling Approach for Global Minimization of Complex Potential Energy Landscapes*, Phys. Rev. Lett. **82**, 3003 (1999).

[49] L. Wesson and D. Eisenberg, *Atomic Solvation Parameters Applied to Molecular Dynamics of Proteins in Solution*, Protein Science **1**, 227 (1992).

Index

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger
Copyright © 2004 Wiley-VCH Verlag GmbH & Co. KGaA
ISBN: 3-527-40406-6