Summer 2015

Stochastic last-mile delivery problems with time constraints

Stacy Ann Voccia
University of Iowa

STOCHASTIC LAST-MILE DELIVERY PROBLEMS WITH TIME CONSTRAINTS

by

Stacy Ann Voccia

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration
in the Graduate College of
The University of Iowa

August 2015

Thesis Supervisors: Professor Ann Campbell
Associate Professor Barrett Thomas

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Stacy Ann Voccia

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Business Administration at the August 2015 graduation.

Thesis Committee: _____
Ann Campbell, Thesis Supervisor

_____
Barrett Thomas, Thesis Supervisor

_____
Philip Jones

_____
Jeffrey Ohlmann

_____
Xiaodong Wu

ACKNOWLEDGEMENTS

I thank my parents, Carol and Joe, for encouraging and supporting me in all my endeavors. I also thank my brother, Jason, who is always there for me and always willing to share his programming expertise.

I express my sincere gratitude to my advisors Ann Campbell and Barry Thomas. They have provided an immense amount of guidance and support throughout this process. I cannot imagine what this process would have been like without them.

I express my appreciation to several faculty members in my department and on my committee: Sam Burer, Ray de Matta, Phil Jones, Jeff Ohlmann, Nick Street, and Xiaodong Wu. I also extend appreciation to the department secretary, Barb Carr, and to the Ph.D. program coordinator, Renea Jay.

I am grateful for the friendship of my classmates, especially Shu Zhang. I would also like to thank my undergraduate advisor, Jon White, for encouraging me to pursue a graduate degree, and for his continued support.

I gratefully acknowledge the computing resources available through the University of Iowa High Performance Computing Group and the Information Systems and Optimization Research (ISOR) Laboratory in the Department of Management Sciences. Without these resources and the people behind them, I would not have been able to complete the computational experiments necessary for my thesis.

ABSTRACT

When a package is shipped, the customer often requires the delivery to be made within a particular time window or by a deadline. However, meeting such time requirements is difficult, and delivery companies may not always know ahead of time which customers will need a delivery. In this thesis, we present models and solution approaches for two stochastic last-mile delivery problems in which customers have delivery time constraints and customer presence is known in advance only according to a probability distribution. Our solutions can help reduce the operational costs of delivery while improving customer service.

The first problem is the probabilistic traveling salesman problem with time windows (PTSPTW). In the PTSPTW, customers have both a time window and a probability of needing a delivery on any given day. The objective is to find a pre-planned route with an expected minimum cost. We present computational results that characterize the PTSPTW solutions. We provide insights for practitioners on when solving the PTSPTW is beneficial compared to solving the deterministic analogue of the problem.

The second problem is the same-day delivery problem (SDDP). The SDDP is a dynamic and stochastic pick-up and delivery problem. In the SDDP, customers make delivery requests throughout the day and vehicles are dispatched from a warehouse or brick and mortar store to serve the requests. Associated with each request is a request deadline or time window. In order to make better-informed decisions,

our solution approach incorporates information about future requests into routing decisions by using a sample scenario planning approach with a consensus function. We also introduce an analytical result that identifies when it is beneficial for vehicles to wait at the depot. We present a wide range of computational experiments that demonstrate the value of our approaches.

PUBLIC ABSTRACT

When a package is shipped, the customer often requires the delivery to be made within a particular time window or by a deadline. However, meeting such time requirements is difficult, and delivery companies may not always know ahead of time which customers will need a delivery. In this thesis, we present models and solution approaches for two last-mile delivery problems in which customers have delivery time constraints and customer presence is unknown in advance. Our solutions can help reduce the operational costs of delivery while improving customer service.

The first problem is the probabilistic traveling salesman problem with time windows (PTSPTW). In the PTSPTW, customers have both a time window and a probability of needing a delivery on any given day. The objective is to find a pre-planned route with an expected minimum cost. We present computational results that characterize the PTSPTW solutions. We provide insights for practitioners on when solving the PTSPTW is beneficial compared to solving the deterministic analogue of the problem.

The second problem is the same-day delivery problem (SDDP). In the SDDP, customers make delivery requests throughout the day and vehicles are dispatched from a warehouse or brick and mortar store to serve the requests. In order to make better-informed decisions, our solution approach incorporates information about future requests into routing decisions. We also introduce an analytical result that identifies when it is beneficial for vehicles to wait at the depot. We present a wide

range of computational experiments that demonstrate the value of our approaches.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF ALGORITHMS

Algorithm

CHAPTER 1
INTRODUCTION

Time-definite delivery plays a crucial role in the shipping industry. Now more than ever, businesses are operating with lean production strategies and just-in-time inventories. Freight is shipped in smaller lot sizes, and the predictability of arrival times is critical (Schultz, 2008). Randy Guidry, Communications Coordinator for Averitt Express, notes, "Every year, more and more of our customers are requesting appointments within a delivery window" (Terreri, 2011). In 2010, time-definite, day-definite, and same-day delivery services accounted for 51.4% of the United States shipping industry's total market value (Datamonitor, 2010). In 2014, FedEx revenues for overnight package delivery in the U.S. accounted for $8.191 billion, about 72% of FedEx's total U.S. package revenues (FedEx, 2014).

While time-definite services are critical for businesses, retail consumers are also taking advantage of delivery speed and convenience. Where instant gratification was once the largest advantage of brick-and-mortar stores over online retailers, same-day delivery now brings near-instant gratification to online shoppers (Howard, 2014). "There's lots going on in this space, and it's all driven by Amazon," says Tom Allason, founder and chief executive of Shutl, a UK-based same-day delivery service that expanded to the U.S. and was purchased by eBay in 2013 (Clifford and Miller, 2012). Allason predicts that as soon as consumers know that same-day delivery is available, they will start to expect it, and demand it. Amazon offers same-day delivery in several urban markets, but most recently introduced Prime Now in Manhattan where

customers can have products delivered within one hour for $7.99 or two hours for free (Amazon). Other companies, such as Instacart, partner with local brick-and-mortar stores. Instacart allows customers to choose between a one or two hour delivery deadline for their online grocery order for prices of $5.99 and $3.99, respectively, on orders of $35 or more (Instacart).

Meeting customer delivery time requirements is challenging and costly when all the relevant information is known in advance. The problem is even more difficult when uncertainties about the future exist. Variable travel and service times, road conditions, map accuracy, and customer presence are examples of uncertainties that create additional scheduling challenges. One way to decrease the uncertainty that exists is to collect historical data to construct probability distributions about important but uncertain aspects of the future. These distributions can be used to help plan for future events.

This thesis focuses on uncertainty in customer presence. Uncertainty in customer presence occurs when it is not known in advance which customers will need service on any given day. We introduce two vehicle routing problems with stochastic customer presence and customer time constraints in last-mile package delivery settings. The first problem is the probabilistic traveling salesman problem with time windows. In this problem, we consider the situation where vehicle routes are planned before service requests are known. The second problem is the same-day delivery problem. In this problem, we consider the situation where vehicle routes are planned during the day as new service requests arrive. For both problems, we present mod-

els, solution approaches, and results that can help practitioners determine when the computational burden of incorporating stochasticity into the model is justified.

We provide a description of the probabilistic traveling salesman problem with time windows in Section 1.1 and a description of the same-day delivery problem in Section 1.2.

1.1 Description of the Probabilistic Traveling Salesman Problem with Time Windows

Currently, many companies employ a pre-planned, or a priori, route which identifies an ordering of all possible customers that a particular driver may need to visit. The driver then skips those customers on the route who do not require a delivery on that day. A priori routes can be implemented with relative ease and are an alternative to the high cost of re-optimization. In addition, a priori routes offer both drivers and customers consistency and help to improve driver efficiency as the driver becomes familiar with the route. Although businesses place high importance on delivery time windows, the consideration of time windows in a priori routing has received little attention in the literature. Additionally, a priori policies can be important subproblems in dynamic routing problems (Goodson et al., 2013; Manni, 2007). For these reasons, we examine the impact of time windows on a priori route design.

We introduce the probabilistic traveling salesman problem with time windows (PTSPTW). While a number of elements of this problem could be modeled stochas-

tically, such as travel time, we maintain the presence of the customer as the only stochastic element in our model. The PTSPTW can be considered an extension of the probabilistic traveling salesman problem (PTSP). The PTSP is the problem of finding a minimum expected cost a priori tour through a set of customers with probabilities of requiring service on any given day. Also associated with each customer in the PTSPTW is a time window during which service must occur.

The work presented in this thesis on the PTSPTW is associated with a published paper in a peer-reviewed journal (see Voccia et al. (2013)).

1.2 Description of the Same-Day Delivery Problem

Same-day delivery for online purchases is characterized by a fleet of vehicles that serve delivery requests over the course of the service day. The requests arrive dynamically, and the only information that is known in advance is probabilistic. Associated with each request is either a deadline or time window during which the delivery must occur. Vehicles are loaded and dispatched from the depot, either a warehouse or brick-and-mortar store, to serve requests. Vehicles are permitted to carry more than one request at a time. The objective is to maximize the number of requests that can be delivered on time. We call this problem the same-day delivery problem (SDDP).

While same-day delivery is focused in urban areas where there is sufficient customer density to support it, same-day delivery is a logistically complicated and expensive service to operate. With the continuing expansion of same-day delivery

services, there is a need for efficient routing strategies. Routing efficiency can be improved when probabilistic information is incorporated into the solution approach to anticipate future requests. For example, instead of leaving the depot immediately after a request is made, probabilistic information can be used to determine if the vehicle should wait at the depot in anticipation of the arrival of another request in the near future. Similarly, anticipating future requests can help to determine which requests should be loaded onto a vehicle and served immediately and which requests should be left behind to be loaded and served at a future time. Both of these scenarios have the potential to produce more efficient routes that serve more customers.

The SDDP is a dynamic pick-up and delivery problem (DPDP) and differs from most DPDPs in two key ways. A majority of the DPDPs in the literature have unique pick-up and delivery locations for each load (Sheridan et al., 2013; Ghiani et al., 2009; Pureza and Laporte, 2008). This is significant because a vehicle can drop-off a load at one location and pick-up a new load at a near-by location. In the SDDP, however, vehicles always have to make the return back to the depot to pick-up a new load. This makes it costly to serve demand that is far from the depot. Certain dial-a-ride problems (DARP) in the literature do have a single location that is a required stop for all requests. An example is a hospital where patients are shuttled between their homes and the hospital. In such a situation, however, the priority is on patient satisfaction (Cordeau and Laporte, 2007). In most situations, it would not be appropriate to pick up certain patients from the hospital, but leave others waiting for the next vehicle. In the SDDP, strategic loading decisions may mean that some

orders are loaded onto another vehicle at a later time. Further, it is a non-trivial task to decide which requests should be loaded onto the departing vehicle.

1.3 Thesis Outline

In this section, we provide an outline for this thesis. The remainder of this chapter, Sections 1.4 and 1.5, are devoted to discussing the literature relevant to the problems introduced in this thesis and the contribution of this work.

Chapter 2 is devoted to the PTSPTW. In Section 2.1, we formulate a recourse model for the PTSPTW and present an algorithm to pre-compute arrival time probabilities. Section 2.2 describes our variable neighborhood search solution heuristic. Sections 2.3 and 2.4 describe our experimental design and results. Conclusions are presented in Section 2.5.

Chapter 3 is devoted to the SDDP. In Section 3.1, we formulate a dynamic programming model for the SDDP. We describe the state and action spaces is Sections 3.1.1 and 3.1.2, respectively. We present the transitions to the post-decision state in Section 3.1.3. We present the transitions to the pre-decision state in Section 3.1.4. The contribution, criterion, and objective are presented in Section 3.1.5. We present an analytical result in Section 3.1.6 that calculates the latest time a vehicle can leave the depot without deteriorating solution quality. We introduce our sample scenario planning approach in Section 3.2. The sample scenario planning approach uses sampled realizations of requests to guide decision making. It consists of two primary components. One component is the subproblem optimization presented in

Section 3.2.1. The second component is the consensus function presented in Section 3.2.2. We present our computational experiments in Section 3.3. This includes a description of our data sets in Section 3.3.1, parameter setting experiments in Section 3.3.2, and results in Section 3.3.3. Conclusions are presented in Section 3.4.

Chapter 4 presents a summary of the thesis and future work.

1.4 Literature Review

In this section, we present a review of the literature related to the PTSPTW and SDDP. In addition to containing customer delivery time constraints, both problems are stochastic. The problems are stochastic in the sense that customer presence is initially known according to probability distributions and realizations are revealed during the execution of the routes. However, the solution approach for the PTSPTW is static, while the solution approach for the SDDP is dynamic. The solution approach for the PTSPTW is static in the sense that an a priori route is created, and regardless of the customer realizations, the order of the customers in the a priori route is maintained during the execution of the route. The solution approach for the SDDP is dynamic in the sense that routing decisions are made based on the current state of the system, and an a priori order of customers is not maintained. We divide this literature review into two sections: (1) stochastic and static and (2) stochastic and dynamic.

While stochastic customer presence is the primary stochastic element in this thesis, we also discuss literature with stochastic travel or service times, and stochastic

demands. Each of these stochastic elements affects the arrival time of vehicles at customer locations, thus playing a large role in the ability to serve customers within their respective deadlines or time windows.

In Section 1.4.1, we describe the stochastic and static literature. In Section 1.4.2, we discuss the stochastic and dynamic literature.

1.4.1 Stochastic and Static

The probabilistic traveling salesman problem (PTSP) is considered a fundamental stochastic vehicle routing problem and is introduced by Jaillet (1988). In the PTSP, customers have an associated probability of needing service on any given day. The PTSPTW, presented in this thesis, is a time-constrained version of the PTSP. Another time-constrained version of the PTSP is the PTSP with deadlines (PTSPD). Campbell and Thomas (2008) introduce and propose models for the PTSPD, and Campbell and Thomas (2009) offer tractable ways to solve the PTSPD. These papers on the PTSPD summarize related literature prior to 2008.

Recent papers have been published on the PTSP and related problems. A subset of these papers focus on solving the PTSP through local search methods (Marinakis and Marinaki, 2010; Balaprakash et al., 2009; Marinakis and Marinaki, 2009; Liu, 2008; Marinakis et al., 2008). Birattari et al. (2008) and Weyland et al. (2009) use local search methods but also employ sampling techniques for the evaluation of the objective function. Mohan et al. (2008) and Chen et al. (2009) introduce arc-routing variants of the PTSP.

Weyland et al. (2013) introduce a new method for reducing the computational complexity of the objective function of the PTSPD. An approximation for the objective function based on Monte Carlo Sampling is combined with a quasi-parallel evaluation of the samples. The authors show that heuristics which use this new method outperform previous approaches for the PTSPD both in terms of computational time and solution value. Weyland et al. (2012) show that the evaluation of the objective function of the PTSPD is #P-hard.

The remainder of this section focuses on related problems with stochastic travel or service times. Jaillet et al. (2014) introduces an objective function to minimize a lateness index for single-vehicle problems with stochastic travel times and deadlines. This objective function works well in some situations, but our objective function is more appropriate for situations where a penalty payment is made based on late delivery.

Chang et al. (2009) develop a heuristic for a version of the time-dependent, stochastic traveling salesman problem with time windows (STSPTW) where travel and service times are stochastic. The authors use an n-path relaxation of a deterministic traveling salesman problem (TSP) and a convolution-propagation approach to approximate arrival times at each customer.

A related problem to the TSP is the orienteering problem (OP). In the orienteering problem, a profit is associated with each customer. The objective is to construct a tour that maximizes the total profit while observing a specified time limit. Campbell et al. (2011) and Tang and Miller-Hooks (2005) introduce stochas-

tic versions of the OP where travel and service times are random. Campbell et al. formulate a recourse model and present a variable neighborhood search to solve instances of the orienteering problem. Tang and Miller-Hooks formulate their problem as a chance-constrained stochastic program and develop an exact method that solves small to medium-size problems as well as a heuristic method for larger instances.

Another related routing problem is the stochastic vehicle routing problem (SVRP) where customers have demands and vehicles have capacity constraints. Several recent papers highlight the SVRP with time windows (SVRPTW). Lei et al. (2011) model a version of the SVRPTW with stochastic demands as a stochastic program with recourse and propose an adaptive large neighborhood search heuristic. Li et al. (2010) formulate their model with stochastic travel and service times as a constrained programming model and a stochastic programming model with recourse and use a tabu search-based method to solve the two models.

Erera et al. (2009) present a stochastic dynamic vehicle routing problem with time windows (SDVRPTW). The SDVRPTW is a problem in which customers are dynamically inserted into routes. The problem is similar to ours in that customer presence is stochastic and each customer has an associated time window. However, our recourse model penalizes time window violations. The model of Erera et al. maintains time window feasibility and uses a paired-vehicle recourse.

1.4.2 Stochastic and Dynamic

The SDDP is most related to dynamic vehicle routing problems (DVRPs) and dynamic pick-up and delivery problems (DPDPs). In these problems, the common stochastic element is customer service requests, which are revealed dynamically. Also related to the SDDP is the dynamic vehicle routing problem with stochastic demands (DVRPSDs). In the DVRPSD, customer demands are stochastic, and demand quantities are revealed when a vehicle arrives to the customer. Due to capacity constraints, the decision elements in this problem are determining when vehicles should return to the depot and which customer to serve next. Similar decisions need to be made in the the SDDP. However in the SDDP, customers are not known in advance whereas in the DVRPSD, all customers are known ahead of time. A survey of the DVRPSD literature is described in Goodson et al. (2015) and Goodson et al. (2013). The primary focus of this literature review is on DVRPs and DPDPs.

We further divide the stochastic and dynamic literature into two subcategories that use stochastic information in different ways to guide decision policies. The two subcategories are papers that use implicit strategies and papers that use explicit strategies to incorporate stochastic information. Implicit strategies determine decision policies without considering the underlying probability distributions. For example, an implicit strategy assumes that customer service requests will arrive, but it does not make assumptions about the underlying distributions that describe where or when the request will arrive. Explicit strategies, on the other hand, incorporate information about the underlying probability distributions. We discuss literature that

uses implicit and explicit strategies in Sections 1.4.2.1 and 1.4.2.2, respectively.

1.4.2.1 Implicit Strategies

Implicit strategies typically occur in the form of a priori waiting rules where vehicles are permitted to wait at suitable locations for a period of time. Another a priori waiting rule postpones the acceptance decision for new customers.

Gendreau et al. (1999) introduce a DVRP with time windows where waiting is allowed in the form of a least commitment strategy. With this strategy, a vehicle is required to wait at its current location if it would otherwise arrive early at the next customer location. In Kiechle et al. (2009), the objective is to ensure short response times for emergency transport when the same vehicles are also used for scheduled, non-emergency transportation. The problem is a DPDP with hard time windows. The strategy where vehicles depart from their current location immediately unless better coverage of the emergency area can be obtained by waiting is determined to be the best strategy.

Other papers seek to distribute waiting time along the route. The problem in Branke et al. (2005) is a DVRP with hard time windows where only one new customer arrives during the execution of the routes. All other customers are known in advance. The objective is to maximize the probability that the additional customer can be integrated into the a priori routes. In the multi-vehicle case, it is optimal to drive without waiting until the time to drive the remaining route is equal to the slack time and then distribute the waiting time between the remaining customers. Mitrović-

Minić and Laporte (2004) consider several waiting strategies for the DPDP with time windows. The best strategy in terms of the number of vehicles used and the route length is their advanced dynamic waiting strategy. This strategy clusters customers into dynamically changing zones based on the travel distance between customers. When the vehicle is ready to leave one zone for the next, it waits according to some proportion of the total remaining slack time. Mitrović-Minić et al. (2004) introduce the concept of a double horizon objective where distance is minimized in the short term, and sufficient slack time is maintained in the long term to enable vehicles to easily respond to future requests.

Waiting can also be applied to customer acceptance decisions. Pureza and Laporte (2008) use such a strategy for a DVRP with time windows. In this strategy, the decision to accept or reject a customer is delayed if the customer can be feasibly inserted into a route in the next time period.

1.4.2.2 Explicit Strategies

Some of the earliest papers that incorporate explicit strategies into dynamic decision making are Psaraftis (1988) as well as Bertsimas and Van Ryzin (1991, 1993). Recent papers incorporate explicit strategies through sampling methods or waiting strategies. Other papers use a threshold policy or directly incorporate arrival rate expectations into the objective function.

The paper most similar to the SDDP presented in this thesis is Azi et al. (2012). Azi et al. (2012) addresses a same-day delivery problem with time windows, however

the objective is to maximize total expected profits, where profits are customer-specific. Our objective is to maximize the number of served customer requests. Further, in Azi et al. (2012), the length of route segments are controlled by a fixed parameter. We allow the length to be determined explicitly through our consensus function while also determining when it may be beneficial to wait at the depot. In addition, we examine the the impact of different time windows and deadlines.

Bent and Van Hentenryck (2004) use a sampling approach to address the DVRP with hard time windows. The authors present a sample scenario planning approach that generates routing plans for scenarios that include known requests and potential future requests that are sampled from a known probability distribution. A consensus function is used to select the best routing plan generated from the scenarios. A similar approach is used in Hvattum et al. (2007) for a VRPSD with time windows. For the DPDP with soft deadlines presented in Ghiani et al. (2009), instead of using a consensus function, the expected future value of feasible routing plans is estimated using a set of scenarios in order to select the best routing plan. Van Hemert and La Poutré (2004) introduce the concept of fruitful regions for DVRPs. A fruitful region is a group of known customer locations that have a high probability of generating service requests in the near future. Potential routing plans are created by sampling fruitful regions, and a self-adaptive evolutionary algorithm is used to determine when to move to a fruitful region.

Other methods that use stochastic information about future request arrivals incorporate explicit waiting strategies. An explicit waiting strategy incorporates in-

formation about the distribution of future requests to decide when and where to wait. A number of papers combine both implicit and explicit waiting strategies. For example, some papers might implicitly determine when vehicles wait and explicitly consider where vehicles wait. For such papers, a common implicit rule for determining when to wait is to allow a vehicle to always wait when either no customers are assigned to it or when it has assigned customers but would arrive early at the next customer if it were to leave immediately. We borrow the term "idle vehicle" from Larsen et al. (2004) to refer to a waiting vehicle that follows this implicit rule.

Several explicit waiting strategies are examined in Larsen et al. (2004). The authors present a single-vehicle dynamic problem with time windows where the distribution area is divided into regions with Poisson arrival rates for each region, and a specified idle point where the vehicle can wait within each region. The authors consider several waiting strategies including waiting at the idle point with the highest arrival rate, and waiting at the idle point with the highest expected number of requests within the specified idle time. In Sheridan et al. (2013) and Moretti Branchini et al. (2009), it is assumed that higher-density areas will generate more requests, and therefore idle vehicles may be sent to wait in such areas. However, probabilities of request arrivals are not explicitly calculated. Meisel (2011) introduce a single-vehicle DPDP and propose an approximate dynamic programming method as a solution approach. The method allows vehicles to wait at the depot and customer locations, as well as strategically accept and reject new customer requests in order to maximize the number of served customers. The authors present two value approximations that

take future requests into account and demonstrate that the approach outperforms a number of benchmark heuristics.

Rather than using a sampling approach, another way to use stochastic information is to incorporate it into the objective calculation. Sáez et al. (2008) employ a fuzzy clustering method using historical demand data to determine future trip patterns and their corresponding occurrence probabilities. The problem is a capacitated DPDP and the objective function takes into account the calculated probabilities of trip patterns.

Albareda-Sambola et al. (2014) and Ichoua et al. (2006) both use threshold policies. For the dynamic multiperiod VRP with time windows presented in Albareda-Sambola et al. (2014), the probability that a customer near an existing route will request service in a later time period is calculated. If the probability is high enough, service to a current customer may be postponed in order to accommodate a new request. Similarly for the DVRP with time windows in Ichoua et al. (2006), a threshold policy is used to determine when it is beneficial to allow a vehicle to wait in its current zone or to move to its next destination in another zone.

1.5 Contributions

This thesis makes several contributions to the literature related to stochastic vehicle routing problems with time constraints. We introduce two new problems, the PTSPTW and the SDDP. In both cases, customer presence is the stochastic element and we use probabilistic information to plan vehicle routes that are robust

to future customer realizations. For both problems, we present results that can help practitioners determine when the computational burden of incorporating stochasticity into the model is justified. Specific contributions made from the PTSPTW and SDDP are discussed in Sections 1.5.1 and 1.5.2, respectively.

1.5.1 Contributions of the PTSPTW

The primary contribution made from the PTSPTW work is the presentation of a recourse model that incorporates time windows into a routing problem with stochastic customer presence. As in previous work on the probabilistic traveling salesman problem with deadlines (PTSPD), our recourse model accounts for late deadline violations by penalizing any violations in the objective function. However, our model also extends the PTSPD to account for early arrivals, an extension that requires non-trivial derivations to calculate the probability of early arrivals at each customer. A secondary contribution of this paper is the results from computational experiments that identify the circumstances under which PTSPTW solutions differ from the solutions for the PTSPTW's deterministic analog, the TSPTW.

1.5.2 Contributions of the SDDP

The primary contribution of the work on the SDDP is the development of a consensus function for use in a sample scenario planning approach. While sample scenario planning was first introduced by Bent and Van Hentenryck (2004), we develop a new consensus function. The consensus function determines how the sampled information is used to construct a new solution. Our consensus function allows for

orders to be left at the depot in anticipation of future requests and allows for more efficient delivery of the orders left at the depot. We also introduce an analytical result that determines how long a vehicle can wait at the depot without impacting solution quality. We compare the solution approach to a myopic approach and offer insight into when anticipating the future adds value. We also examine the computational trade-off between solution quality and runtime due to the number of scenario samples and the length of the sampling horizon. In addition, we introduce the multi-trip team orienteering problem with time windows (MTTOPTW) as a subproblem for the SDDP. We formulate a mixed integer programming model for the MTTOPTW.

CHAPTER 2
THE PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS

In this chapter, we present our work on the PTSPTW. In Section 2.1, we formulate a recourse model for the problem. In Section 2.2, we present the variable neighborhood heuristic that we use to solve instances of the problem. We describe our experimental design in Section 2.3 and present our results in Section 2.4. Our conclusions are presented in Section 2.5.

2.1 Recourse Model

In this section, we present a recourse model for the PTSPTW. A recourse model is characterized by two stages. In stage one, an a priori solution is determined. Then the random variables are realized and a recourse action is applied to the original solution. In this case, the a priori solution is a route that identifies the order in which customers will be visited. After information becomes available about which customers need to be visited, the recourse action is applied. This recourse action consists of adjusting the route so that customers who do not need to be visited are skipped, but the remaining customers are still visited in the a priori route order.

The PTSPTW is the problem of finding a minimum expected cost a priori tour through a set of customers $N = \{i \mid 1, \ldots, n\}$ with probabilities $P = \{p_i \mid 1, \ldots, n\}$ of requiring service on any given day. The travel time between any two customers i and j is given by $d_{i,j}$, where $d_{i,j} = d_{j,i}$. These travel times also serve as the cost to

traverse each arc. Also associated with each customer $i \in N$ is a time window with an earliest time of service e_i and latest time of service l_i.

In our model, service at each customer cannot begin before e_i. An early arrival requires the driver to wait until e_i. However, the vehicle is permitted to visit a customer after l_i has passed, but a penalty is incurred for doing so. We utilize a per-unit-time penalty charge for arriving after the close of the time window. The penalty is represented by λ. The per-unit-time charge represents cases where the delivery company is charged per unit time of lateness. For instance, FedEx Custom Critical refunds varying percentages of the cost of a shipment based on how late the shipment is delivered (FedEx Custom Critical, Inc, 2005). For additional examples, see Charnsirisakskul et al. (2004) and Slotnick and Sobel (2005). Thus, the objective of the PTSPTW, formulated as a recourse model, is to identify an a priori route where the sum of the expected travel costs between customers and the expected penalty term is minimum.

In the case of our recourse model, we let τ define an order, or tour, in which the customers $i = 1, \ldots, n$ are to be visited. We assume that the customers are indexed according to their position in τ. We assume a tour always begins and ends at a fixed depot, and the depot is indexed as $i = 0$. We assume integer travel times and deadlines and that all tours start at time $t = 0$. Our recourse model provides a formula for evaluating the cost of a given tour τ.

As a result of our assumptions, the expected travel costs can be calculated as they are for the well-known PTSP (see Jaillet (1988) for further reference) with a

straightforward modification for the fixed depot (Campbell and Thomas, 2008):

$$\sum_{j=1}^{n} p_j d_{0,j} \prod_{k=1}^{j-1}(1-p_k) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_i p_j d_{i,j} \prod_{k=i+1}^{j-1} (1-p_k) + \sum_{i=1}^{n} p_i d_{i,0} \prod_{k=i+1}^{n} (1-p_k). \quad (2.1)$$

This formula calculates the probability and the resulting expected cost of any arc that may appear in the tour. The expected cost of an arc (i, j) depends on customers i and j being realized and no customers k, $k = i+1, \ldots, j-1$, being realized.

For both the issue of early and late arrivals, we need to determine the probability that arrival at a customer occurs at a particular time t. To begin, let the random variable

$$X_i = \begin{cases} 0 & \text{if customer } i \text{ is not realized} \\ 1 & \text{if customer } i \text{ is realized.} \end{cases}$$

Also, let A_i be a random variable representing the time of arrival at the customer in position i in τ. We assume that arrival at a customer i cannot be early or late if customer i is not realized. Hence, we are left to compute $P(A_i = t \mid X_i = 1)$, where t represents the time of arrival at customer i. For notational convenience, let $g(i,t) = P(A_i = t \mid X_i = 1)$. We also let $G(i,t) = P(A_i \leq t \mid X_i = 1)$, which can be computed as $G(i,t) = \sum_{k=0}^{t} g(i,k) = G(i,t-1) + g(i,t)$. We assume that arrival at the depot occurs at time $t = 0$. Therefore, $G(0,t) = 1$ for all values of t. Because there is no time window or penalty associated with the depot, it is not necessary to consider the probability that the arrival at the depot on the return trip occurs at any particular time.

To account for the fact that arrival at customer i depends on the departure times at previous customers, we let $h(i,t)$ be the probability of departing from cus-

tomer i at time t. We can then compute the g and h values in an iterative, recursive fashion. Note that arrival and departure times for a customer differ when waiting for the opening of the time window occurs. Before describing the computation procedure in Algorithm 2.1, we first introduce bounds on the earliest and latest times that a customer can be visited.

If the triangle inequality holds for the travel time data, then the times between the earliest possible arrival time and the latest possible departure time that need to be considered for each customer i can be limited based on the location of customer i in the tour. The earliest arrival time at customer i, expressed as T_i^{min}, occurs when all customers prior to i are not realized. Thus, $T_i^{min} = d_{0,i}$ for all i. The latest possible arrival time at customer i is computed by assuming that every customer prior to i is realized. If arrival occurs prior to the start of the time window for any customer prior to i, then this additional waiting time must be accounted for in the computation. Given $T_1^{max} = d_{0,1}$, we can compute the other values recursively:

$$T_i^{max} = \max(T_{i-1}^{max}, e_{i-1}) + d_{i-1,i}. \tag{2.2}$$

All values of t outside of the range of T_i^{min} to T_i^{max} result in g and h values of zero unless $T_i^{max} < e_i$, in which case $h(i, e_i)$ has a value of one.

We now clarify the initialization procedure for Algorithm 2.1. The values $g(i,t)$, $h(i,t)$, and $G(i,t)$ are set to zero for all combinations of i and t, where $i = 0, 1, \ldots, n$ and $t = 0, 1, \ldots, T_n^{max}$, unless explicitly noted. The values $g(0,0)$ and $h(0,0)$ are set to one to represent the probabilities of both arriving and departing from the depot at time zero. Also, $h(i, e_i)$ is set to one if $T_i^{max} < e_i$. Finally, $G(0,t)$

Algorithm 2.1 Pseudocode for computation of g and h values for Recourse Model

Initialization:
  $g(0,0) \leftarrow 1$ and $g(i,t) \leftarrow 0$ for $(i,t) \neq (0,0)$,
  $i = 0, 1, \ldots, n$ and $t = 0, 1, \ldots, T_n^{max}$.
  $h(0,0) \leftarrow 1$ and $h(i,t) \leftarrow 0$ for $(i,t) \neq (0,0)$,
 $i = 0, 1, \ldots, n$ and $t = 0, 1, \ldots, T_n^{max}$.
 $G(0,t) \leftarrow 1$ for $t = 0, 1, \ldots, T_n^{max}$and
 $G(i,t) \leftarrow 0$ for $i = 1, \ldots, n$ and $t = 0, 1, \ldots, T_n^{max}$.
for $i = 1, \ldots, n$ **do**
    if $T_i^{max} < e_i$ **then**
        $h(i, e_i) \leftarrow 1$
for $i = 1, \ldots, n$ **do**
    for $t = T_i^{min}, \ldots, T_i^{max}$ **do**
        for $v = 0, \ldots, i-1$ **do**
            if $t \geq d_{v,i}$ **then**
                $g(i,t) \leftarrow p_v h(v, t - d_{v,i}) \prod_{k=v+1}^{i-1}(1 - p_k) + g(i,t)$
        $G(i,t) \leftarrow G(i,t-1) + g(i,t)$
        if $t < e_i$ **then**
            $h(i,t) \leftarrow 0$
        else if $t = e_i$ **then**
            $h(i,t) \leftarrow G(i,t)$
        else
            $h(i,t) \leftarrow g(i,t)$

is set to one for all values of t.

Using the previously described bounds and initialization, Algorithm 2.1 recursively computes h and g values. Recall that $g(i,t)$ represents the probability of arriving at customer i at a particular time t, given that customer i is realized. Therefore, the algorithm considers the probability that previous customers are realized, as well as the probability of departing at a particular time from these previous customers. Thus $g(i,t) = \sum_{v=0}^{i-1} p_v h(v, t - d_{v,i}) \prod_{k=v+1}^{i-1}(1 - p_k)$, and is computed recursively in Algorithm 2.1. In words, $p_v h(v, t - d_{v,i})$ represents the probability that customer v is realized times the probability that departure from customer v occurs at the current time minus the travel time from customer v to customer i. This departure time implies that customers v and i are realized but no customers k, $k = v + 1, \ldots, i - 1$, are realized. The probability that these customers are not realized is represented by the term $\prod_{k=v+1}^{i-1}(1 - p_k)$ in the formula. Note also that when $t < d_{v,i}$, $t - d_{v,i}$ is negative, and thus represents an impossible situation when considering the original assumptions of the problem. Thus $h(v, t - d_{v,i})$ is set to zero when $t < d_{v,i}$.

Because $G(i,t)$ is the cumulative function for $g(i,t)$, $G(i,t)$ can be computed recursively using the formula $G(i,t-1)+g(i,t)$. We are left to compute $h(i,t)$, which depends on three situations. Either

1. time t is before the start of the time window,
2. time t is equivalent to the start of the time window, or
3. time t is after the start of the time window.

If the first situation occurs, then the probability of departing from customer i at time

t is zero. That is, $h(i,t) = 0$. If the second situation occurs, then the probability of departing from customer i at time t is equivalent to the probability of arriving at customer i at or before time t. Therefore, $h(i,t) = G(i,t)$. If the third situation occurs, $h(i,t) = g(i,t)$. This completes the description of Algorithm 2.1.

To account for the per-unit-time penalty, we complete the objective by adding the following to Formula 2.1:

$$\sum_{i=1}^{n} p_i \sum_{t=l_i+1}^{T_i^{max}} \lambda g(i,t)(t - l_i). \tag{2.3}$$

Thus we have the following objective function:

$$\begin{aligned} &\sum_{j=1}^{n} p_j d_{0,j} \prod_{k=1}^{j-1}(1-p_k) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_i p_j d_{i,j} \prod_{k=i+1}^{j-1}(1-p_k) \\ &\quad + \sum_{i=1}^{n} p_i d_{i,0} \prod_{k=i+1}^{n}(1-p_k) + \sum_{i=1}^{n} p_i \sum_{t=l_i+1}^{T_i^{max}} \lambda g(i,t)(t-l_i). \end{aligned} \tag{2.4}$$

The computation of the objective function is more complex than that of the PTSP, but is similar to the complexity of the PTSPD. The initialization of Algorithm 2.1 is computed in $O(n\ T_n^{max})$ time, while the body of the algorithm is computed in $O(n^2 \max_i\{T_i^{max} - T_i^{min}\})$ time. Without restrictions on distances, this is not necessarily polynomial in n. Once the g and h values are known, the per-unit-time penalty portion of the objective can be computed in $O(n \max_i\{T_i^{max} - l_i\})$ time, and the distance portion of the objective function can be computed in $O(n^2)$ time. The entire objective calculation is dominated by the g and h computations, and thus takes $O(n^2 \max_i\{T_i^{max} - T_i^{min}\})$ time. In addition, Weyland et al.'s (2012) characterization of the PTSPD objective function as #P-hard can be trivially extended to the

PTSPTW.

2.2 Solution Approach

Our solution approach is motivated by the recent work of da Silva and Urrutia (2010) on the TSPTW, the deterministic version of the problem discussed in this chapter. Da Silva and Urrutia combine a Variable Neighborhood Search (VNS) with a Variable Neighborhood Decent (VND) as a local search. The algorithm yields high quality solutions for the data sets from which those used in this section are derived and achieves best-known results on other benchmark data sets. Thus, we chose to implement a VNS/VND heuristic as well.

In broad terms, the VNS we use can be described as a systematic change of neighborhoods during a two phase procedure. The first phase is a perturbation of the current best solution, while the second phase is a descent to find a new local minimum. The purpose of the perturbation is to move the solution out of a local minimum. Note that we do not need the initial construction phase implemented by da Silva and Urrutia (2010) because we penalize time window violations, thus making all solutions feasible.

We now describe the algorithm by beginning with a description of the VNS. As input, the VNS takes an initial seed tour x, a value k_{max} that controls the maximum level of perturbation in the Shake function, and an objective function $f(\cdot)$ that maps a tour to the set of positive real numbers. Specifically, $f(\cdot)$ is the objective function presented in Formula 2.4. After initializing k to 1 and *improvement* to true,

the algorithm perturbs the current solution x by calling the Shake function. The perturbed solution, x', is then passed to the VND to find a local minimum x^*. If the objective value of x^* is less than the objective value of the current solution x, then x is set to x^*, k is reset to 1, and the procedure repeats. If, on the other hand, the objective value of x^* is not less than the objective value of x and k does not equal k_{max}, then k is incremented by 1 and the procedure repeats. When k reaches k_{max}, then no improving solution has been found within the last k_{max} iterations. The parameter *improvement* is set to false, the current solution x is set to *best*, and the algorithm terminates returning *best*, the best tour found. The pseudo code for the VNS algorithm is presented in Algorithm 2.2.

Algorithm 2.2 VNS

Input: x, k_{max},$f(\cdot)$
Output: *best*
Initialization:
$k \leftarrow 1$
improvement $\leftarrow$ True
while *improvement* = True **do**
    $x' \leftarrow Shake(k, x)$
    $x^* \leftarrow VND(x', j_{max})$
    if $f(x^*) < f(x)$ **then**
        $x \leftarrow x^*$
        $k \leftarrow 1$
    else if $k < k_{max}$ **then**
        $k \leftarrow k + 1$
    else
        improvement $\leftarrow$ False
best $\leftarrow x$

We now describe the Shake function that is used to perturb the current best solution. As input, the Shake function requires a tour x and a neighborhood specification k. We denote a set of neighborhoods for the Shake function with $N_k^{Shake} = \{1, \ldots, k_{max}\}$ and with $N_k^{Shake}(x)$ the set of solutions in the k^{th} neighborhood. Specifically, $N_k^{Shake}(x)$ contains all tours that differ from x by a combination of k 1-shift moves. A 1-shift move removes a single customer from a tour and reinserts the customer in a new location on the tour. We assume the solutions from $N_k^{Shake}(x)$ are ordered such that $\{x^1, \ldots, x^{|N_k^{Shake}(x)|}\}$. The perturbation selects a random tour $x^w \in N_k^{Shake}(x)$, sets x' to x^w, and outputs the perturbed solution x'. The pseudo code for the algorithm is presented in Algorithm 2.3.

Algorithm 2.3 Shake

Input: x, k
Output: x'
1: $w \leftarrow [1 + Rand(0, 1) \times |N_k^{Shake}(x)|]$
2: $x' \leftarrow x^w$

In the algorithm presented by da Silva and Urrutia, k_{max} is set to 30. However, such large neighborhoods lead to many iterations of the VNS and VND. Because the computational complexity of the PTSPTW objective calculation is greater than that of the TSPTW objective calculation, a large number of iterations causes the running time to be very large. Additionally, during preliminary experiments, we found that the algorithm often converged to the same solution for values of k_{max} set to 30 and

set to 5. For these reasons, we set k_{max} to 5.

Hansen et al. (2010) propose an implementation of VNS/VND for combinatorial optimization problems which differs from the implementation proposed by da Silva and Urrutia in the procedure for changing neighborhoods within the VND. In preliminary experiments, we found that both algorithms generally converged to the same solutions, but the algorithm proposed by Hansen et al. converged faster. For this reason, we base our VND implementation on the algorithm proposed by Hansen et al..

The VND is used to optimize the recently perturbed solution. The input for the VND includes a tour x, a value j_{max}, and the previously described objective function $f(\cdot)$. We denote a set of neighborhoods for the VND with $N_j^{VND} = \{1, \ldots, j_{max}\}$ and with $N_j^{VND}(x)$ the set of solutions in the j^{th} neighborhood. The value j_{max} represents the last neighborhood that the VND iterates through. We set j to 1 to specify the the previously discussed 1-shift neighborhood and we set j to 2 to specify the 2-Opt neighborhood. A 2-Opt move deletes two edges from a tour so that the tour is broken into two paths, then reconnects the paths in the only other possible way. Because we specify two neighborhoods for the VND, we set j_{max} to 2. Our choices of the 1-shift and 2-Opt neighborhoods are also motivated by the work of da Silva and Urrutia (2010) and the descriptions of the neighborhoods can be found therein.

After initializing j to 1 and *improvement* to true, a locally optimal solution x' is found by iterating through the entire neighborhood N_j^{VND}. If the objective value of x' is less than the objective value of the current solution x, then x is set to x^*, j is reset to 1, and the procedure repeats. If, on the other hand, the objective value

of x' is not less than the objective value of x and j does not equal j_{max}, then j is incremented by 1 and the procedure repeats. When j reaches j_{max}, then no improving solution has been found in either neighborhood. The parameter *improvement* is set to false, the current solution x is set to x^*, and the algorithm terminates returning x^*, the best locally optimal tour found. The pseudo code for the VND algorithm is presented in Algorithm 2.4.

Algorithm 2.4 VND

Input: x, j_{max}, $f(\cdot)$
Output: x^*
Initialization:
$j \leftarrow 1$
$improvement \leftarrow$ True
while $improvement =$ True **do**
    $x' \leftarrow \arg\min_y\{f(x^y)\}$
    if $f(x') < f(x)$ **then**
        $x \leftarrow x'$
        $j \leftarrow 1$
    else if $j < j_{max}$ **then**
        $j \leftarrow j + 1$
    else
        $improvement \leftarrow$ False
$x^* \leftarrow x$

2.3 Experimental Design

Next, we investigate what instance characteristics result in different tours when customer presence is modeled stochastically instead of deterministically. To address this question, we focus on the effects of customer realization probabilities, time win-

dow widths, per-unit-time penalties, and the number of customers.

The data sets we use are a subset of the TSPTW instances first proposed by Dumas et al. (1995). We use the 20-, 40- and 60-customer instances with time-window widths of 20, 60, and 100 units. These instances are labeled "Feasible" in the results tables. We also generate new data sets from each of the existing instances. These new instances differ in the starting and ending times of the time windows, and represent the situations where feasible solutions with respect to time windows are unlikely to exist if all customers are realized. These situations are likely to occur as the demand for time-definite services grows. Our results, which are presented in the next section, indicate that large cost savings are possible when these situations are modeled stochastically compared to deterministically. We denote the new early and late deadlines as e_i' and l_i', respectively. In general, we set l_i' equal to the early deadline. Then we set e_i' equal to l_i' minus the width of the corresponding time window, unless that time is negative in which case e_i' is set to zero. As an example, consider an instance with a time window width of 20 where $e_i = 15$ and $l_i = 35$. Then $l_i' = 15$, and since $15 - 20 = -5$, set $e_i' = 0$. Sometimes, however, l_i' is equal to zero. In this case, we instead let l_i' equal the late deadline and e_i' equal l_i' minus the width of the corresponding time window. Suppose we again have a time window width of 20. If $e_i = 0$ and $l_i = 20$, then since l_i' would equal 0, we let $l_i' = 20$ and $e_i' = 0$. These instances will be referred to in the tables by the label "Infeasible."

Similar to the design of the PTSPD experiments in Campbell and Thomas (2008) (see Campbell and Thomas (2006) for PTSPD benchmark data sets), we con-

sider four different probability settings for each instance. Three of these settings are homogeneous with customer presence probabilities set to 0.1, 0.5, and 0.9, respectively. These instances are referred to as 0.1, 0.5, and 0.9 in the results tables. The probabilities represent the likelihood that each customer will be realized. A probability of 0.1 may be suitable to represent a home or small business because packages are unlikely to be delivered every day. Probabilities of 0.5 may be suitable to represent small or medium businesses that receive packages a few days a week, while probabilities of 0.9 may be suitable to represent medium or large businesses that receive packages almost every day. The fourth probability setting is heterogeneous, where the probability of each customer is randomly assigned a probability of either 0.1 or 1. This case represents the situation in which both small and large businesses are served by the same vehicle. This data set will be referred to in the tables by the label "Mixed."

For each instance with 20-, 40- and 60- customers, we consider two different per-unit-time penalties. As was done in the PTSPD work of Campbell and Thomas (2008), one penalty is set to 5, while in the other case the penalty is set to 50. These penalties represent small and large costs of failing to satisfy customer time windows. In total, we solve 720 different instances.

Because our goal is to determine differences between TSPTW and PTSPTW solutions, for feasible instances we seed the VNS/VND heuristic described in Section 2.2 with the best-known TSPTW solution (see da Silva and Urrutia (2010)) and solve the corresponding PTSPTW instance with homogeneous and mixed probability

settings. For the infeasible instances, we construct corresponding TSPTW solutions to use as seed solutions for the VNS/VND heuristic. To construct these TSPTW solutions, we assign the same per-unit-time penalties as the corresponding PTSPTW set, and set all probabilities to one. We run the VNS/VND heuristic 10 times using the corresponding best-known TSPTW solution with feasible time windows, converted to infeasible time windows, as a starting solution. We select the best TSPTW solution out of the 10 runs for each instance to use as the seed solutions for the VNS/VND algorithm to solve the corresponding PTSPTW infeasible instances with homogeneous and mixed probability settings.

In evaluating our computational results, it is necessary to be able to compare two solutions. One useful metric of comparison is the Hamming distance introduced by Bierwirth et al. (1996) for the Job Shop Scheduling problem. Ehmke et al. (2012) use this measure with vehicle routing problems. In terms of tours, the Hamming distance is the ratio of the precedence relationships of customers being preserved between two tours. In order to calculate the Hamming distance, a tour must be converted to a bit-string representation. A tour with n customers has a bit-string representation of $\frac{n^2-n}{2}$ bits. If customer i is located before customer j on a tour, then the bit is set to one. On the other hand, if customer j is located prior to i, then the bit is set to zero. The normalized Hamming distance for two tours represented as bit-strings x and y is calculated by $d_{x,y} = \frac{1}{l} \sum_{k=1}^{l} xor(x_k, y_k)$, where l is the length of the bit-strings. The "exclusive-or" operator is represented by $xor(\cdot)$ and results in one if the bits x_k and y_k differ, and zero otherwise. A Hamming distance of zero indicates

two identical tours, while a Hamming distance of one indicates that no customers share the same precedence relationship between two tours. A Hamming distance of one occurs only in the case where one tour is the exact inverse of the other tour.

Another metric that we use to compare solutions is the percent change in objective value. In the case of feasible instances, the initial solution is the best-known TSPTW solution. For infeasible instances, the initial solution is the best TSPTW solution we found using our heuristic technique. Both the initial and final solutions are evaluated with the PTSPTW objective function. The percent change in objective value is calculated as

$$\left(\frac{Initial\,objective\,value - Final\,objective\,value}{Initial\,objective\,value}\right) \times 100\%. \tag{2.5}$$

Because the initial objective value is either equal to or greater than the final objective value, the percent change always represents a decrease in value from the initial seed tour.

Because of the probabilistic nature of the algorithm, each instance is run 10 times. In the results section, we provide tables with summarized results. Before discussing the results, we describe the column labels in the tables. "Avg. Δ (%)" represents the average percent change in objective value for each of the five instances across the 10 run for each instance. Similarly, "Avg. Ham." and "Avg. Time" represent the average Hamming distance and the average CPU seconds of runtime, respectively. The abbreviation "NC" within the tables stands for "No Change," as opposed to "0.00" which means a very small change has occurred.

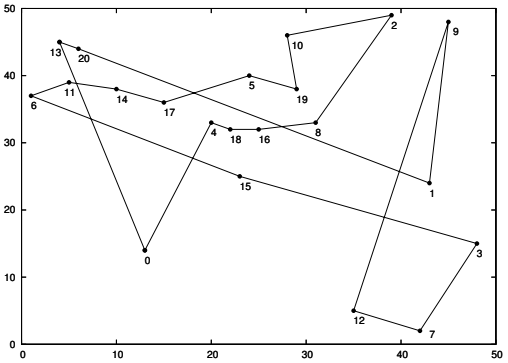
2.4 Results

The summarized results in Tables 2.1 and 2.2 indicate that the 0.1 homogeneous setting has greater differences in tour structure from the TSPTW solution than the other homogeneous and mixed settings. Meanwhile, the 0.9 setting displays the smallest differences. This outcome is logical, because as customer realization probabilities increase towards one, the problem gets closer to the deterministic TSPTW. An example of this trend is displayed in Figure 2.1. The PTSPTW solution for the 0.9 probability instance is visually similar to the TSPTW solution and has a Hamming distance of 0.02. On the other hand, the 0.1 instance looks very different from the TSPTW solution and has a larger Hamming distance of 0.14. The key difference is the number of times that the tour crosses itself. The increased crossings in instances with smaller probabilities occur as a way to improve routing cost when it is possible to skip customers.

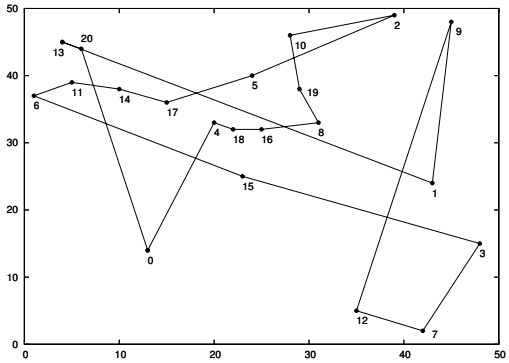
Out of all the probability settings, the mixed setting displays the largest differences from the TSPTW solutions for cost. This is because of the PTSPTW's ability to prioritize customers with larger probabilities in the tour. As noted earlier, however, the mixed setting does not display the largest differences in structure. This indicates that even small changes in tour structure can produce large changes in cost.

It is intuitive that time windows drive the tour construction for the TSPTW. Our results support the idea that time windows are also a significant contributor to tour construction for the PTSPTW. In a majority of cases, instances with time window widths of 100 have greater percent changes in objective value and also greater

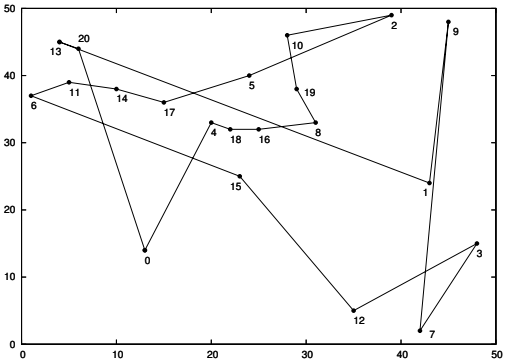
Figure 2.1: Solution tours for homogeneous probabilities for the third feasible instance with 20 customers, time window widths of 100, and a per-unit-time penalty of 50.



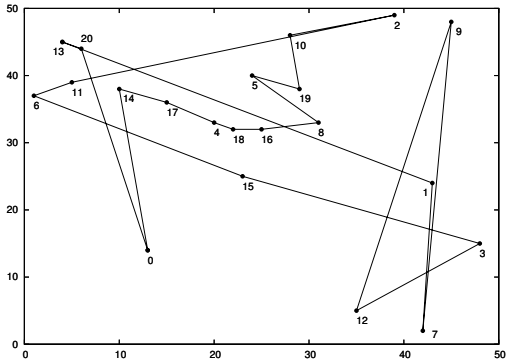
(a) TSPTW solution



(b) PTSPTW solution for 0.9 probabilities, Hamming distance: 0.02, change in objective value: 0.06%



(c) PTSPTW solution for 0.5 probabilities, Hamming distance: 0.03, change in objective value: 0.48%



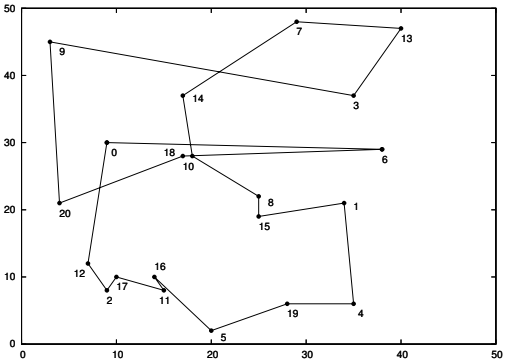
(d) PTSPTW solution for 0.1 probabilities, Hamming distance: 0.14, change in objective value: 0.39%

Hamming distances than instances with time window widths of 20. This is because larger time windows allow greater flexibility in the construction of the tours.

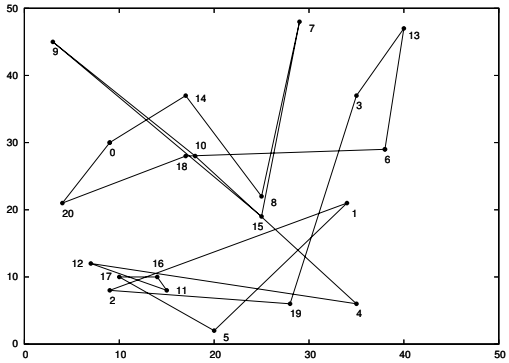
Another result comes from comparing the percent changes in objective value of the feasible instances to those of the infeasible instances. The average percent changes in objective value for the feasible instances are much smaller than those of the infeasible instances. The average change across all feasible instances is 0.92%, while the average for the infeasible instances is 11.86%. An example of the type of change in tour structure that creates a large decrease in objective value is displayed in Figure 2.2. The PTSPTW tour chooses to serve customers 7 and 9, which are far from the other customers, in a very different way. If customers 7 and 9 are not realized, the PTSPTW tour will be much shorter and allow more customers to be visited feasibly than the TSPTW tour. This result indicates that when infeasibility exists in terms of time windows, it may be very beneficial to use the PTSPTW formulation as opposed to the TSPTW formulation.

Per-unit-time penalty increasing from 5 to 50 affects the solution tours for all the instances as expected. For the feasible instances, the average percent change in objective value decreases as the penalty increases. Because these are feasible instances, it is possible to serve many customers within their respective time windows. It is likely that incurring a large penalty cost for missing a deadline is more costly than traveling a longer distance to arrive within the respective time window. Thus, larger penalties push the solutions toward the optimal TSPTW solutions. All of the Hamming distances either decrease or do not demonstrate change within two significant

Figure 2.2: Solution tour for 0.1 probabilities for the second infeasible instance with 20 customers, time window widths of 100, and a per-unit-time penalty of 50.



(a) TSPTW solution



(b) PTSPTW solution for 0.1 probabilities, Hamming distance: 0.43, change in objective value: 53.41%

digits.

For infeasible instances, the opposite trend occurs. The average percent change in objective value generally increases as the penalty increases from 5 to 50. By construction of these instances, it is unlikely that a tour exists such that the driver will be able to serve all customers within their respective time windows. Therefore, penalty charges are unavoidable. As the charge increases, the cost of the tour increases. Any percent change in objective value that occurs at a low penalty level is multiplied at the higher level.

An increase in the number of customers also causes an increase in the average percent change in objective value for both feasible and infeasible instances. Hamming distance, on the other hand, tends to decrease and is most evident with time window widths of 100. It is possible that because of the larger number of customers on the

tour, more precedence relationships are preserved even when customer positions differ and cause changes in the objective value.

Lastly, it is worthwhile to mention two factors contributing to increased runtimes. The most noticeable factor is problem size. As the number of customers increases, so does the runtime. This is due to the fact that as the solution space grows, the number of times that the objective value is calculated increases. The second factor is the use of small or mixed probabilities. Instances with these two probability types deviate more from their TSPTW solutions than instances with larger probabilities. This implies that the increased runtime is caused by more local search moves.

2.5 Conclusions

In this chapter, we have presented a recourse model for the PTSPTW that requires construction of a tour that visits each realized customer but pays a penalty if the late deadline of the time window constraint is violated. We described a VNS/VND heuristic to solve instances of the PTSPTW. Finally, we presented computational results that offer the following key insights when comparing PTSPTW solutions to their TSPTW counterparts:

- Large probabilities of customer presence have less impact on the solution than small probabilities.
- When customers have a combination of high and low probabilities, solutions are significantly impacted by modeling customers stochastically. This is because of the stochastic model's ability to prioritize customers with larger probabilities

in the solution.

- Large time window widths have a greater impact on the solution than small time window widths when customers are modeled stochastically. An explanation is that as time window widths increase, the algorithm has more flexibility for placing customers on the tour.
- Solutions to the PTSPTW come with large computation times due to the complexity of the objective calculation. Factors that contribute to larger runtimes are large problem sizes and problems with low or mixed probabilities.

Table 2.1: Summary of Feasible Experiments.

		AVG. Δ (%)				AVG. Ham.				AVG. Time			
	Probability	0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed
$n = 20$	Penalty = 5												
	$w = 20$	0.03	0.04	NC	0.02	0.01	0.00	NC	0.00	6.09	5.36	5.45	5.07
	$w = 60$	0.50	2.00	0.46	0.47	0.05	0.03	0.01	0.02	6.08	5.99	5.79	4.79
	$w = 100$	0.46	0.20	0.06	0.98	0.10	0.01	0.01	0.05	6.31	4.66	4.81	4.88
$n = 20$	Penalty = 50												
	$w = 20$	0.03	NC	NC	0.01	0.01	NC	NC	0.00	5.84	5.12	5.45	4.89
	$w = 60$	0.40	1.01	0.13	0.42	0.04	0.02	0.01	0.01	6.28	6.15	5.40	4.80
	$w = 100$	0.31	0.11	0.02	0.97	0.09	0.01	0.01	0.04	6.68	5.00	4.89	4.94
$n = 40$	Penalty = 5												
	$w = 20$	0.12	0.30	0.03	0.57	0.01	0.00	0.00	0.00	156.08	138.82	132.23	120.88
	$w = 60$	0.58	0.96	0.01	2.65	0.03	0.01	0.00	0.02	167.98	124.56	113.12	120.56
	$w = 100$	1.73	1.13	0.00	4.98	0.05	0.02	0.00	0.04	164.72	124.49	103.83	153.78
$n = 40$	Penalty = 50												
	$w = 20$	0.11	0.20	0.04	0.37	0.01	0.00	0.00	0.00	154.54	131.82	135.48	127.85
	$w = 60$	0.50	0.69	0.01	2.62	0.03	0.01	0.00	0.02	170.77	120.24	115.17	120.41
	$w = 100$	1.43	1.11	0.00	3.87	0.05	0.02	0.00	0.03	169.63	120.33	104.94	133.84
$n = 60$	Penalty = 5												
	$w = 20$	0.19	0.62	0.06	2.65	0.01	0.00	0.00	0.01	823.93	705.98	668.30	960.30
	$w = 60$	0.59	0.96	0.78	3.65	0.02	0.01	0.00	0.01	932.91	716.23	677.40	850.16
	$w = 100$	1.52	1.88	0.17	4.54	0.04	0.02	0.00	0.03	1,013.68	783.61	632.49	970.53
$n = 60$	Penalty = 50												
	$w = 20$	0.16	0.33	0.00	1.61	0.01	0.00	0.00	0.00	990.25	827.87	819.30	936.87
	$w = 60$	0.45	0.45	0.03	2.76	0.02	0.00	0.00	0.01	1,192.77	868.17	766.85	810.23
	$w = 100$	1.31	1.31	0.17	4.41	0.04	0.02	0.00	0.02	1,303.77	937.40	741.76	989.51

Table 2.2: Summary of Infeasible Experiments.

		AVG. Δ (%)				AVG. Ham.				AVG. Time			
	Probability	0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed
$n = 20$	Penalty = 5												
	$w = 20$	2.93	0.60	NC	1.63	0.01	0.00	NC	0.00	6.02	5.38	5.23	4.95
	$w = 60$	3.82	3.40	0.04	4.98	0.07	0.06	0.00	0.04	6.02	5.24	4.36	4.95
	$w = 100$	5.46	6.00	0.51	22.58	0.13	0.06	0.01	0.17	6.78	4.53	3.85	6.24
$n = 20$	Penalty = 50												
	$w = 20$	13.03	2.20	NC	5.52	0.02	0.01	NC	0.00	5.99	5.61	5.06	5.51
	$w = 60$	21.30	8.93	2.74	6.61	0.08	0.05	0.01	0.04	6.70	5.09	4.63	5.26
	$w = 100$	28.98	13.89	1.74	39.60	0.17	0.08	0.01	0.17	6.93	5.06	3.99	6.13
$n = 40$	Penalty = 5												
	$w = 20$	0.80	3.93	1.75	3.42	0.02	0.01	0.00	0.01	158.54	129.41	120.97	126.99
	$w = 60$	4.53	7.16	0.20	3.16	0.05	0.02	0.00	0.02	165.50	110.67	94.76	109.27
	$w = 100$	13.16	14.14	1.53	24.99	0.08	0.04	0.01	0.11	163.05	113.88	91.78	157.81
$n = 40$	Penalty = 50												
	$w = 20$	5.54	12.77	3.58	8.54	0.01	0.01	0.00	0.00	155.76	133.02	123.90	157.76
	$w = 60$	24.45	12.77	0.30	3.71	0.06	0.01	0.00	0.02	196.67	110.97	95.29	143.38
	$w = 100$	25.51	20.95	0.44	3.94	0.09	0.04	0.01	0.03	182.84	119.47	91.93	167.07
$n = 60$	Penalty=5												
	$w = 20$	0.66	1.12	0.05	4.79	0.01	0.00	0.00	0.01	833.44	705.66	686.59	1,157.12
	$w = 60$	18.94	23.46	12.22	37.42	0.05	0.03	0.04	0.07	1,073.15	825.71	843.04	1,397.34
	$w = 100$	6.67	6.87	0.72	22.46	0.06	0.02	0.00	0.10	1,098.85	703.37	561.38	1,362.16
$n = 60$	Penalty = 50												
	$w = 20$	7.44	6.02	0.02	14.31	0.01	0.00	0.00	0.01	999.66	832.76	769.01	1,153.81
	$w = 60$	40.86	33.69	18.68	54.01	0.04	0.02	0.03	0.07	1,307.59	930.00	890.20	1,259.74
	$w = 100$	48.12	36.69	4.04	33.28	0.06	0.02	0.01	0.11	1,655.80	876.45	671.45	1,968.37

CHAPTER 3
THE SAME-DAY DELIVERY PROBLEM

In this chapter, we present our work on the SDDP. We describe our dynamic programming formulation in Section 3.1. We describe the state and action spaces is Sections 3.1.1 and 3.1.2, respectively. We present the transitions to the post-decision state in Section 3.1.3. We present the transitions to the pre-decision state in Section 3.1.4. The contribution, criterion, and objective are presented in Section 3.1.5. We present an analytical result in Section 3.1.6 that calculates the latest time a vehicle can leave the depot without deteriorating solution quality. In Section 3.2, we provide an overview of our sample scenario planning method and describe the subproblem for the method in Section3.2.1 and the consensus function in Section 3.2.2. We present our experimental design, parameter setting, and computational results in Section 3.3. We discuss conclusions in Section 3.4.

3.1 Model Formulation

The SDDP is characterized by a fleet of vehicles operating from a depot and by a set of locations. Let $\mathcal{M} = \{1, \ldots, M\}$ be a set of M identical vehicles initially located at the depot at time 0. Let the set $\mathcal{I} = \{0, 1, \ldots, I\}$ represent all the locations in the problem. In set $\mathcal{I}$, the depot is represented by 0, and customer locations are represented by $1, \ldots, I$. Each customer location may represent an individual customer or an aggregation of customers. For example, in a city scenario, customers might be aggregated by city blocks. We assume the travel time between any two locations

$i, j \in \mathcal{I}$ is deterministic and is represented by τ_{ij}.

Customers request service throughout the day until a fixed cut-off time L. The arrivals of requests are described by a known arrival rate and distribution. The arrival rate of requests at location i is represented by λ_i. Multiple requests may arrive from the same location over the course of the day.

When a request is made, the location and time of that request is denoted by ω and rt, respectively. Also associated with a request is a known service time μ, and a delivery time window $[e, l]$. The service time is assumed to be a function of the location making the request and is known in advance, but all other information associated with a request becomes known once the request is realized. The delivery of a request must occur at or after time e, but no later than time l. The restriction on beginning service after e means that a vehicle may need to wait at the request location. When only a delivery deadline is present, e is set to rt, the time that the request is made.

Requests can be served either by the existing fleet of vehicles or by a third party. We assume that it costs more to serve a request via the third party than through the existing fleet. We do not allow a request's time constraints to be violated. We assume that once a request is loaded on a vehicle, then the request will be delivered by that vehicle.

A vehicle $m \in \mathcal{M}$ can leave the depot immediately when a request is assigned to it, or the vehicle can wait for additional time before leaving. Once a vehicle leaves the depot, the route for that vehicle is fixed and the vehicle returns to the depot when

it has made all its assigned deliveries. Vehicles are required to return to the depot by a depot deadline equal to $L + \Delta$. The objective is to maximize the expected number of requests that can be fulfilled by the existing fleet within the given time constraints.

We formulate the SDDP as a dynamic program. A dynamic program models sequential decisions. Before presenting the details of our model, we summarize the sequence of events that occurs at each decision epoch. A decision epoch is triggered when at least one vehicle is scheduled to depart from the depot at the current time or is waiting at the depot when a new request arrives. Decision epoch k occurs at the random time t_k and indicates the end of period $k-1$ and the beginning of period k. At the beginning of epoch k, the pre-decision state s_k is observed. The transition to the pre-decision state is random because the time of request arrivals is unknown in advance. For each vehicle, an action is selected and executed. For a vehicle arriving to or waiting at the depot, the action directs the vehicle to leave the depot at the current time with a set of requests or wait at the depot. When a vehicle leaves the depot, requests are marked as served and the contribution of serving each request is recorded as $C_k(s_k, a)$. These events are captured in the deterministic transition from the pre-decision state s_k to the post-decision state s_k^a. The next epoch then occurs at time t_{k+1} and the process repeats until the time of the depot deadline is reached.

3.1.1 State Space

The state of the system contains all the information necessary to make request and routing decisions at decision epoch k. The state includes the current time, re-

quest locations, request time windows, vehicle arrival times to the depot, and vehicle departure times from the depot. Let t_k represent the current time and the time at which decision epoch k occurs. Let the set $\mathcal{N}_k$ represent the set of realized requests at epoch k that have neither been loaded onto a vehicle in $\mathcal{M}$, nor assigned to a third party. We represent attributes of request $n \in \mathcal{N}_k$ by the tuple $(\omega_k^n, e_k^n, l_k^n)$.

Let ω_k^n be the location of request n. Let e_k^n be the beginning of the time window and l_k^n be the time window deadline for request n. Let $(\omega_k, e_k, l_k)_{n \in \mathcal{N}_k} = (\omega_k^n, e_k^n, l_k^n)$ denote the vector of request attributes.

We represent attributes of vehicle $m \in \mathcal{M}$ at epoch k by the tuple (p_k^m, d_k^m). The attribute p_k^m and d_k^m represent the next scheduled arrival and departure times to and from the depot, respectively. The arrival and departure times are updated when vehicle m leaves the depot. Additionally, the departure time can be updated when vehicle m arrives at the depot or while the vehicle is waiting at the depot. Let $(p_k, d_k)_{m \in \mathcal{M}} = (p_k^m, d_k^m)$ denote the vector of vehicle attributes. Combining this information into the tuple $(t_k, (\omega_k, e_k, l_k), (p_k, d_k))$, we have the state of the system.

3.1.2 Action Space

Let $s_k = (t_k, (\omega_k, e_k, l_k), (p_k, d_k))$ be the state at decision epoch k. Given state s_k, decision epoch k occurs at time t_k. Let $\mathcal{M}'_k = \{m | p_k^m \leq t_k, \forall m \in \mathcal{M}\}$ be the set of vehicles located at the depot at epoch k. The k^{th} epoch is triggered at time t_k when at least one of the following is true:

1. $\{m | d_k^m = t_k, \forall m \in \mathcal{M}\} \neq \emptyset$, or

2. $\mathcal{M}'_k \neq \emptyset \wedge rt^n_k = t_k, \exists n \in \mathcal{N}_k$.

Condition 1 states that at least one vehicle has a scheduled departure time from the depot that is equal to the current time. Condition 2 states that at least one vehicle is located at the depot, and at least one new request is realized at the current time.

An action is selected at each decision epoch and is the assignment of requests in $\mathcal{N}_k$ to vehicles in $\mathcal{M}$. Requests may be assigned to vehicles located at the depot, but are not allowed to be assigned vehicles that are en route. For vehicles that are assigned a non-empty set of requests, a departure rule determines if each vehicle should leave the depot immediately, or wait. One departure rule requires vehicles to leave immediately. Another departure rule is presented in Section 3.1.6. If immediate departure is required, vehicles can load the assigned requests and leave the depot. We assume that requests that are loaded onto a vehicle are served in the sequence of the minimum duration route that satisfies request time windows and the depot deadline. Our assumption of the minimum duration route is addressed in Section 3.2.1. Additionally, we require that request n is assigned, at most, to one vehicle.

Action a is an M-dimensional vector where the m^{th} element, a_m, is the action associated with vehicle $m \in \mathcal{M}$. Let $\phi(a_m)$ be a function that returns the time that vehicle m returns to the depot given the set of customers specified by action a_m. The

set of available actions in state s_k for vehicle m is

$$\{a_m \subseteq \mathcal{N}_k|$$

$$a_m = \emptyset, \text{ if } m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}, \tag{3.1}$$

$$a_m \subseteq \mathcal{N}_k, \text{ if } m \in \mathcal{M}'_k, \tag{3.2}$$

$$\bigcap_{\forall c \in \mathcal{M}} a_c = \emptyset, \tag{3.3}$$

$$\phi(a_m) \leq L + \Delta\}. \tag{3.4}$$

Condition 3.1 states that if vehicle m is en route, then no additional requests are assigned to that vehicle. Condition 3.2 allows a subset of requests in $\mathcal{N}_k$ to be assigned to vehicle m if vehicle m is at the depot. However, it is possible that the selected subset is the empty set. Condition 3.3 requires that each request is assigned at most to one vehicle. Condition 3.4 requires that vehicle m returns to the depot before the depot deadline. Denote by $\mathcal{A}(s_k)$ the set of actions available in state s_k. Thus, $\mathcal{A}(s_k)$ consists of all feasible action vectors.

3.1.3 Transition to Post-Decision State

Given state s_k and the selected action $a \in \mathcal{A}(s_k)$, a deterministic transition is made to post-decision state $s_k^a = (t_k^a, (\omega_k^a, e_k^a, l_k^a), (p_k^a, d_k^a))$. In this transition, the current time remains unchanged. Thus,

$$t_k^a = t_k. \tag{3.5}$$

In transitioning to the post-decision state, a request n transitions out of the set $\mathcal{N}_k$ when it is either loaded onto a vehicle in $\mathcal{M}$ or assigned to a third party. A

request is assigned to a third party when it is not assigned to a vehicle in $\mathcal{M}$, but a vehicle needs to be dispatched in order to achieve a feasible delivery time. For requests that remain in $\mathcal{N}_k$, request locations and time windows remain unchanged. Thus,

$$\omega_k^{na} = \omega_k^n, \forall n \in \mathcal{N}_k \tag{3.6}$$

$$e_k^{na} = e_k^n, \forall n \in \mathcal{N}_k \tag{3.7}$$

$$l_k^{na} = l_k^n, \forall n \in \mathcal{N}_k. \tag{3.8}$$

If the chosen action a_m for vehicle m at a given decision epoch contains requests, then vehicle m is required to leave the depot with the requests and serve them in the sequence specified by the minimum duration route. Consequently, the arrival time back to the depot for vehicle m can be calculated based on the departure time and the route duration. Given an action a_m that requires vehicle m to leave the depot, let the function $\phi(a_m)$ return the next depot arrival time for vehicle m. If vehicle $m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}$, then the depot arrival time remains unchanged. Thus, depot arrival times are updated as

$$p_k^{ma} = \begin{cases} \phi(a_m), & \forall m \in \mathcal{M}'_k, \\ p_k^m, & \forall m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}. \end{cases} \tag{3.9}$$

We also update the time of the scheduled vehicle departure from the depot for each vehicle. If a_m contains requests for vehicle $m \in \mathcal{M}'_k$, then the next depot departure time is set according to the next arrival time such that $d_k^{ma} = p_k^{ma} = \phi(a_m)$. If instead a_m does not contain requests for vehicle $m \in \mathcal{M}'_k$, then vehicle m is required to wait at the depot for additional time. Let η be the additional amount of time to

wait. Then $d_k^{ma} = t_k + \eta$. If vehicle $m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}$, then the depot departure time remains unchanged. Thus, the depot departure times are updated as

$$d_k^{ma} = \begin{cases} \phi(a_m), & \text{if } a_m \neq \emptyset,\ \forall m \in \mathcal{M}'_k, \\ t_k + \eta, & \text{if } a_m = \emptyset,\ \forall m \in \mathcal{M}'_k, \\ d_k^m, & \forall m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}. \end{cases} \tag{3.10}$$

3.1.4 Transition to Pre-Decision State

At decision epoch $k+1$, a transition is made from post-decision state s_k^a to pre-decision state $s_{k+1} = (t_{k+1}, (\omega_{k+1}, e_{k+1}, l_{k+1}), (p_{k+1}, d_{k+1}))$. During the transition, the arrival of new requests are observed. The transition is probabilistic because the realization times for new requests are unknown prior to t_{k+1}.

We first describe the transition for the time attribute. Recall that a decision epoch is triggered when at least one vehicle is scheduled to depart from the depot at the current time or is waiting at the depot when a new request arrives. Thus, there are two cases to consider:

1. all vehicles are en route, and
2. at least one vehicle is located at the depot.

In the first case, t_{k+1} can be calculated deterministically because the next decision epoch depends on the earliest scheduled arrival time to the depot. (Recall that arrival and departure times are equal for vehicles that are en route.) In the second case, t_{k+1} depends on the earliest scheduled departure time and on the probabilistic arrival of new requests. Therefore in the second case, t_{k+1} is calculated probabilistically.

In the first case, t_k^a is updated according to known departure times. Thus,

$$t_{k+1} = \min_{\forall m \in \mathcal{M}} \{d_k^{ma}\}. \tag{3.11}$$

It is useful to write t_{k+1} in terms of t_k^a. Let

$$t_{k+1} = t_k^a + \min_{\forall m \in \mathcal{M}} \{d_k^{ma}\} - t_k^a = t_k^a + \epsilon, \tag{3.12}$$

where $\epsilon = \min_{\forall m \in \mathcal{M}} \{d_k^{ma}\} - t_k^a$.

In the second case, the time of the next decision epoch is probabilistic. The epoch occurs at the time of the earliest scheduled vehicle departure or when the next new request arrives, whichever event occurs first. Let B be a random variable that represents the inter arrival of the next request. Let b be a realization of B. Then t_k^a is updated as

$$t_{k+1} = \min\{\min_{\forall m \in \mathcal{M}} \{d_k^{ma}\}, t_k^a + b\} = \min\{t_k^a + \epsilon, t_k^a + b\}. \tag{3.13}$$

We now describe the transition probabilities of t_{k+1} for the second case. Inter arrival times of events in a Poisson process with an arrival rate of λ are exponentially distributed with mean $1/\lambda$. Let $\lambda = \sum_{i=1}^{I} \lambda_i$. Therefore, the random variable B is exponentially distributed with mean $1/\lambda$.

Using a discrete time model, assume b represents b units of time. Also assume $P\{B = b\} = P\{B > b-1\} - P\{B \geq b\}$, for all b where $b = 1, 2, \ldots \infty$. Therefore, the probability mass function f_B is

$$f_B = \begin{cases} \exp^{-\lambda(b-1)} - \exp^{-\lambda(b)}, & b = 1, \ldots, \infty, \\ 0, & \text{o.w.} \end{cases} \tag{3.14}$$

Let Z be the random variable representing the time until the next decision epoch, and let z be a realization of Z. The probability mass function for Z is identical to that of B for $z = 1, \ldots, \epsilon - 1$. When $Z = \epsilon$, we have

$$P\{Z = \epsilon\} = P\{B = \epsilon\} + P\{B > \epsilon\} = \exp^{-\lambda(\epsilon-1)} - \exp^{-\lambda(\epsilon)} + \exp^{-\lambda(\epsilon)} = \exp^{-\lambda(\epsilon-1)}. \tag{3.15}$$

Using the calculated probabilities for when at least one vehicle is located at the depot, we define the probability mass function f_Z that governs the transition of t_k^a to the realization t_{k+1} in the second case as

$$f_Z = \begin{cases} \exp^{-\lambda(\epsilon-1)}, & z = \epsilon, \\ \exp^{-\lambda(z-1)} - \exp^{-\lambda(z)}, & z = 1, \ldots, \epsilon - 1, \\ 0, & \text{o.w.} \end{cases} \tag{3.16}$$

Writing the update for t_k^a in terms of the random realization z, we have

$$t_{k+1} = t_k^a + z. \tag{3.17}$$

We now describe the probabilistic transitions for requests. New requests that arrive transition into $\mathcal{N}_{k+1}$ and their state attributes are created. The arrival of new requests is probabilistic according to a Poisson process and varies according to location.

In the first case when all vehicles are en route, several requests from the same location may arrive between t_k^a and t_{k+1}. Therefore, we calculate the probability that x_i requests arrive from location $i \in (I) \setminus \{0\}$ during a time period of length ϵ. Let X_i^ϵ be a random variable representing the number of requests arriving from location i during ϵ. The assumption of stationary increments implies that X_i^ϵ is also Poisson

distributed with mean $\lambda_i \epsilon$. Therefore, the probability of x_i new requests arriving from location i during a time period of length ϵ is

$$P\{X_i^\epsilon = x_i\} = \exp^{-\lambda_i \epsilon} \frac{\lambda_i \epsilon_i^x}{x!}, x_i = 0, 1, \ldots \infty. \tag{3.18}$$

In the second case, when at least one vehicle is at the depot, the probability of one new request transitioning into $\mathcal{N}_{k+1}$ is the probability that the request occurs before $t_k^a + \epsilon$. This is equivalent to $P\{B \leq \epsilon\} = 1 - P\{B > \epsilon\} = 1 - \exp^{-\lambda\epsilon}$. Let κ be the random variable associated with the next request location. Given that a new request arrives before ϵ, the probability that the new request comes from a particular location $i \in \mathcal{I} \setminus \{0\}$ can be calculated as

$$P\{\kappa = i | B \leq \epsilon\} = (\frac{\lambda_i}{\lambda})(1 - \exp^{-\lambda\epsilon}). \tag{3.19}$$

The probability that a new request does not arrive before ϵ is $P\{B > \epsilon\} = \exp^{-\lambda\epsilon}$.

Attributes for requests in $\mathcal{N}_k^a$ remain unchanged as they transition to $\mathcal{N}_{k+1}$. Vehicle attributes also remain unchanged.

3.1.5 Contribution, Criterion, and Objective

Choosing action a when in state s_k results in a reward. Let $\bar{\mathcal{N}}_k^a = \{n | n \in \bigcup_{\forall m \in \mathcal{M}} a_m\}$ be the set of requests assigned to a vehicle in $\mathcal{M}$ at decision epoch k. Then the contribution function is

$$C_k(s_k, a) = |\bar{\mathcal{N}}_k^a|. \tag{3.20}$$

Let K be a random variable that represents the total number of decision epochs. A decision rule at epoch k is a function $\delta(s_k)$ that selects an available action

$a \in \mathcal{A}(s_k)$. A policy π is a sequence of decision rules. Let Π represent the set of all possible policies. The criterion is

$$v^{\pi}(s_0) = \mathbb{E}^{\pi}\{\sum_{k=0}^{K} C_k(s_k, \delta^{\pi}(s_k))|s_0\}. \tag{3.21}$$

The objective is to find an optimal policy π^* such that $v^{\pi^*}(s_0) \geq v^{\pi}(s_0)$ for all $\pi \in \Pi$.

3.1.6 Maximum Delay Time

For vehicles located at the depot, the question of when to leave the depot arises. One possibility is to leave immediately if the vehicle's planned route is non-empty. Another possibility is to wait for a period of time before departing. On one hand, doing work is preferred to idle time. On the other hand, waiting at the depot can result in more efficient routes because more requests are allowed to accumulate before departing. We introduce a departure strategy that allows vehicles to wait at the depot without sacrificing the ability to serve requests.

How long the vehicle should wait at the depot relies on the computation of the maximum delay time (MDT). The MDT is computed based on given set of requests that would be served according to the shortest duration route that is feasible. Because of the existence of time windows, a vehicle may arrive to a request location before the beginning of the time window and will therefore have to wait before beginning service. Given a planned route, the MDT calculation considers both this and deadline constraints to determine how much longer a vehicle can remain at the depot without violating deadlines and without changing the return time to the depot. The MDT formula is presented in Proposition 1. Our main result is presented in Proposition 2,

which states that by maintaining the same depot return time and delaying departure time by the MDT, at least as many customers can be served as without delaying the departure time. The results of this section mean that given an assignment of requests to a vehicle, if it is not necessary to depart immediately to complete the tour, then the assignment should not be made at the current time.

Before presenting Propositions 1 and 2, we define additional notation and quantities. Define a tour as an ordering of requests such that the ordering starts and ends at the same depot. Associate the number $j = 0, 1, \ldots, J, J+1$ with the position of a request on the tour, where the depot is always in positions 0 and $J+1$. Let $[e_j, l_j]$ and μ_j be the time window and service time associated with the request in position j on a tour. Similarly, let $\tau_{j,j+1}$ be the travel time from the request in position j on the tour to the request in position $j+1$ on the tour. We also define the following quantities:

D_j	the vehicle departure time at the request in position j,
A_j	the vehicle arrival time at the request in position j,
W_j	the time a vehicle spends waiting at the request in position j to begin service,
CW_j	the cumulative wait time of a vehicle starting with position 0 and summing up to and including position j, and
DS_j	the deadline slack at the request in position j, which is the amount of time that A_j can be delayed without violating l_j.

The quantities e_j, l_j and μ_j are known in advance for j, $1 \leq j \leq J+1$ and $\tau_{j,j+1}$ is known in advance for j, $0 \leq j \leq J$. In addition, we assume $W_0 = 0$ and $e_{J+1} = 0$ so that waiting does not occur at the depot. We also assume that D_0 is a known value.

A_j and D_j are computed recursively using the following formulas.

$$A_j = D_{j-1} + \tau_{j-1,j}, \quad \forall j \geq 1. \tag{3.22}$$

$$D_j = \begin{cases} \max\{A_j, e_j\} + \mu_j, & 1 \leq j \leq J, \\ 0, & j = 0. \end{cases} \tag{3.23}$$

Wait time is computed using the following formula.

$$W_j = \begin{cases} \max\{e_j - A_j, 0\}, & \forall j \geq 1, \\ 0, & j = 0. \end{cases} \tag{3.24}$$

Alternatively, the departure time can be rewritten using wait time.

$$D_j = A_j + W_j + \mu_j, \quad 1 \leq j \leq J. \tag{3.25}$$

Cumulative wait times and deadline slacks are calculated with the the following formulas.

$$CW_j = \sum_{k=0}^{j} W_k, \quad \forall j. \tag{3.26}$$

$$DS_j = l_j - A_j, \quad \forall j \geq 1. \tag{3.27}$$

Given this notation, we present Proposition 1.

Proposition 1. *Given a feasible tour, let D_0 be the departure time from the depot at the beginning of the tour and A_{J+1} be the arrival time at the depot at the end of the tour. The tour can be delayed from leaving the depot by g units of time such that time window feasibility is maintained and the new return time at the depot $A_{J+1}^{new} = A_{J+1}$. The maximum delay time (MDT) is the maximum such g and is computed by the following formula.*

$$MDT = \min\{\min_{1 \leq j \leq J}\{DS_j + CW_{j-1}\}, CW_J\} \tag{3.28}$$

Formula 3.28 states that the MDT is the minimum of the cumulative wait time through J and the amount of time that A_j can be shifted forward for j, $1 \leq j \leq J$ without deadline violation.

The proof of Proposition 1 is a direct consequence of Lemmas 1 – 4. We now present Lemmas 1 – 4.

Lemma 1. *A shift forward of g units of time such that $D_1^{new} = D_0 + g$ results in a new arrival time at the request in position j of*

$$A_j^{new} = \begin{cases} A_j, & \text{if } g \leq CW_{j-1}, \\ A_j + h - CW_{j-1}, & \text{if } g > CW_{j-1}. \end{cases} \tag{3.29}$$

Proof. We prove formula 8 by induction. First, we show that the result is true for $j = 1$ and $j = 2$.

Let $j = 1$. Suppose the departure time from the depot is shifted by g units. Then by the given definitions,

$$\begin{aligned} A_1^{new} &= D_0 + g + \tau_{0,1} \\ &= A_1 + g. \end{aligned} \tag{3.30}$$

Because $g \geq CW_0 = 0$, the result holds for $j = 1$.

Let $j = 2$. If $A_1^{new} \leq e_1$, then $D_1^{new} = D_1$. Thus, no further arrival times are shifted and $A_j^{new} = A_j, j \geq 2$. Because a positive wait time at location j indicates that $A_j \leq e_j$, we can determine whether or not $A_1^{new} \leq e_1$ by using the wait time

formula for W_1^{new} and comparing g to CW_1 .

$$\begin{aligned} W_1^{new} &= \max\{e_1 - A_1^{new}, 0\} \\ &= \max\{e_1 - (A_1 + g - CW_0), 0\} \\ &= \max\{(e_1 - A_1) - (g - CW_0), 0\} \\ &= \max\{W_1 - g + CW_0, 0\} \\ &= \max\{CW_1 - g, 0\}. \end{aligned} \tag{3.31}$$

Therefore, $A_1^{new} \leq e_1$ when $g \leq CW_1$.

However, if $A_1^{new} > e_1$, or equivalently, $g > CW_1$, then $D_1^{new} \neq D_1$. Hence, the arrival time at the request in position 2 is shifted such that the new arrival time is

$$\begin{aligned} A_2^{new} &= D_1^{new} + \tau_{1,2} \\ &= A_1 + g + s_1 + \tau_{1,2}. \end{aligned} \tag{3.32}$$

The difference between A_2^{new} and A_2 is

$$\begin{aligned} A_2^{new} - A_2 &= A_1 + g + s_1 + \tau_{1,2} - (D_1 + \tau_{1,2}) \\ &= A_1 + g + s_1 + \tau_{1,2} - (A_1 + W_1 + s_1 + \tau_{1,2}) \\ &= g - W_1 \\ &= g - CW_1. \end{aligned} \tag{3.33}$$

Therefore, the new arrival time at the request in position 2 can be expressed as $A_2^{new} = A_2 + g - CW_1$. Thus, the result holds for $j = 2$.

Now, suppose the result holds for $j = 3, \ldots, h-1$. We show the the result is true for $j = h$.

If $g \leq CW_{h-1}$, then $A_{h-1}^{new} \leq e_{h-1}$ and $D_{h-1}^{new} = D_{h-1}$ and no further arrival times are shifted. Hence, $A_j^{new} = A_j, j \geq k$. The fact that $g \leq CW_{h-1}$ implies

$A_{h-1}^{new} \leq e_{h-1}$ can be demonstrated using the wait time formula.

$$\begin{aligned} W_{h-1}^{new} &= \max\{e_{h-1} - A_{h-1}^{new}, 0\} \\ &= \max\{e_{h-1} - (A_{h-1} + g - CW_{h-2}), 0\} \\ &= \max\{(e_{h-1} - A_{h-1}) - (g - CW_{h-2}), 0\} \\ &= \max\{W_{h-1} - g + CW_{h-2}, 0\} \\ &= \max\{CW_{h-1} - g, 0\}. \end{aligned} \tag{3.34}$$

Therefore, if $g \leq CW_{h-1}$, then $A_{h-1}^{new} \leq e_{h-1}$. Note that the analysis depends on g and CW_{h-2}. If $g \leq CW_{h-2}$, then $g \leq CW_{h-1}$, which implies $A_{h-1}^{new} \leq e_{h-1}$. If $g > CW_{h-2}$, then $A_{h-1}^{new} = A_{h-1} + g - CW_{h-2}$.

If $g > CW_{h-1}$, then $A_{h-1}^{new} > e_{h-1}$ and $D_{h-1}^{new} \neq D_{h-1}$. Hence, the arrival time at the request in position h is shifted such that the new arrival time is

$$\begin{aligned} A_h^{new} &= D_{h-1}^{new} + \tau_{h-1,h} \\ &= A_{h-1} + g - CW_{h-1} + \tau_{h-1,h}. \end{aligned} \tag{3.35}$$

The difference between A_h^{new} and A_h is

$$\begin{aligned} A_h^{new} - A_h &= A_{h-1} + h + s_{h-1} + \tau_{h-1,h} - (D_{h-1} + \tau_{h-1,h}) \\ &= A_{h-1} + g + CW_{h-2} + s_{h-1} + \tau_{h-1,h} \\ &\quad - (A_{h-1} + W_{h-1} + s_{h-1} + \tau_{h-1,h}) \\ &= g - (CW_{h-1} + W_{h-1}) \\ &= g - CW_{h-1}. \end{aligned} \tag{3.36}$$

Thus, the new arrival time at location h can be expressed as $A_h^{new} = A_h + g - CW_{h-1}$. Therefore, by the induction hypothesis, the result holds for $j = h, \forall h \geq 1$. □

Lemma 2. *$DS_j + CW_{j-1}$ is the maximum amount of time that departure from the depot can be delayed without violating l_j.*

Proof. The proof follows from Lemma 1. If $g \leq CW_{j-1}$, then $A_j^{new} = A_j \leq l_j$. Therefore, g can be increased to at least CW_{j-1} without violating l_j.

If $g > CW_{j-1}$, then $A_j^{new} = A_j + g - CW_{j-1}$. Therefore, $A_j^{new} = l_j$ if

$$\begin{aligned} j &= l_j - A_j + CW_{j-1} \\ &= (l_j - A_j) + CW_{j-1} \\ &= DS_j + CW_{h-2}. \end{aligned} \tag{3.37}$$

And any g greater than $DS_j + CW_{j-1}$ leads to an arrival time at j that violates l_j. □

Lemma 3. *A shift forward of $\min_{1 \leq j \leq J}\{DS_j + CW_{j-1}\}$ is deadline feasible for j, $1 \leq j \leq J$.*

Proof. The proof is a direct consequence of Lemma 2. □

Lemma 4. *CW_J is an upper bound for the MDT.*

Proof. The proof follows from Lemma 1. If $g \leq CW_J$, then $A_{J+1}^{new} = A_{J+1}$.

If $g > CW_{J+1}$, then $A_{J+1}^{new} = A_{J+1} + g - CW_J > A_{J+1}$, contradicting that the shift be such that $A_{J+1}^{new} = A_{J+1}$. Therefore, the maximum g such that $A_{J+1}^{new} = A_{J+1}$ holds true is CW_J.

This result and our beginning assumption of a feasible tour implies that l_{J+1} is not violated with a shift of CW_J. □

Lemma 5. *Equation 3.28 is deadline feasible for all j.*

Proof. The proof is a direct consequence of Lemmas 3 and 4. □

Proposition 2. *Given a feasible tour that leaves the depot at D_0, there exists a policy that serves at least as many customers when departing at $D_0 + MDT$ compared to the tour departing at D_0.*

Proof. The proof is straightforward. □

3.2 Scenario-Based Planning

In scenario-based planning, different scenarios are constructed based on random samples of the stochastic parameter set. We refer to these scenarios as scenario samples, where a scenario sample includes the set of realized requests and a set of random requests sampled from the stochastic parameter set. We refer to requests that are sampled as sampled requests. Sampled requests represent requests that may occur in the future. We develop heuristic partial policies using the scenario samples to inform action selection. Our scenario-based planning algorithm is presented in Algorithm 3.1.

Algorithm 3.1 takes several pieces of information as input. It takes the set $\mathcal{N}_k$, the set of realized but unassigned requests. It also takes the set $\mathcal{M}'_k$, which is the set of vehicles currently located at the depot. The list *depart_times* stores the earliest time that each vehicle is able to depart from the depot. Let *depart_times* be indexed such that *depart_times*$[m]$ is the earliest departure time for vehicle $m \in \mathcal{M}$. The earliest departure times are calculated as

$$depart\_times[m] = \begin{cases} t_k, & \forall m \in \mathcal{M}'_k, \\ p_k^m, & \forall m \in \{\mathcal{M} \setminus \mathcal{M}'_k\}. \end{cases} \tag{3.38}$$

Algorithm 3.1 Scenario-Based Planning

Input: $\mathcal{N}_k$, $\mathcal{M}'_k$, *depart_times*, *num_ss*, *ss_params*
Output: *Distinguished_Plan*
1: **Initialization:** *all_ss* = [], *all_ssp* = []
2: **for** i in 1 to *num_ss* **do**
3: *ss* ← GenerateSS($\mathcal{N}_k$, *ss_params*)
4: *all_ss*.append(*ss*)
5: **for** i in 1 to *num_ss* **do**
6: *ssp* ← SubOpt(*all_ss*[i], *depart_times*)
7: *all_ssp*.append(*ssp*)
8: *Distinguished_Plan* ← Consensus(*all_ssp*, $\mathcal{M}'_k$)

This means that for vehicles located at the depot, the earliest departure time is equal to the current time. For vehicles en route, the earliest departure time for vehicle m is equal to the scheduled arrival time for vehicle m.

Also specified as input is *num_ss*, the number of scenario samples to be generated. Important to the scenario sample generation is the list *ss_params*. The list *ss_params* stores the information necessary to generate the set of sampled requests. Included information is the current time, the sampling horizon, the set of locations, request arrival rates, and the information necessary to generate time windows for sampled requests. It is important to know the current time because sampled requests represent requests that may occur in the future. The sampling horizon controls how far in the future sampled requests are generated.

The algorithm begins with the initialization of two lists: *all_ss* and *all_ssp*. The first list, *all_ss*, stores all the scenario samples that are generated. The second list, *all_ssp*, stores all the scenario sample routing plans that are created for each scenario sample. A routing plan specifies a route for each vehicle.

Next, beginning on line 2, Algorithm 3.1 repeatedly calls function GenerateSS in order to generate the number of scenario samples specified by num_ss. The function GenerateSS takes $\mathcal{N}_k$ and ss_params as input because each scenario sample contains unassigned realized requests and sampled requests. Each scenario sample is stored in the list all_ss.

Beginning on line 5, Algorithm 3.1 constructs routing plans for each scenario sample using the function SubOpt. SubOpt is our subproblem optimization method, which will be described in Section 3.2.1. SubOpt takes a scenario sample $all_ss[i]$ as input as well as the list $depart\_times$ containing the earliest departure times for each vehicle. Knowing the earliest departure time for each vehicle is important because vehicles en route are not available to begin serving additional requests until they complete their current route and return to the depot. SubOpt returns a routing plan that contains all realized and sampled requests in the scenario sample. Each resulting scenario sample routing plan, ssp, is stored in the list all_ssp.

The final step of Algorithm 3.1, which occurs on line 8, sends all the scenario sample routing plans to the function Consensus. Consensus uses the information from the scenario sample routing plans to develop a routing plan that is used to select actions. The routing plan that is developed is called the distinguished plan. In addition to taking all_ssp as input, Consensus also takes $\mathcal{M}'_k$ as input. Knowing which vehicles are located at the depot is important because only those vehicles may be scheduled to leave from the depot. Therefore, in constructing the distinguished plan, the consensus function only considers information about routes for vehicles

located at the depot. We address our consensus function in Section 3.2.2. The distinguished plan is the output for Algorithm 3.1.

3.2.1 Subproblem Optimization

The subproblem optimization optimizes routing plans for each of the scenario samples. Each scenario sample contains realized requests, which have not been assigned to either a vehicle in the existing fleet or a third party, and sampled requests. It is unlikely for a routing plan to exist that allows all requests in a scenario sample to be served within each request's time window. Hence, the subproblem is an orienteering problem in which a subset of requests that can be feasibly served on the next route segment is chosen. The subproblem that is solved at each decision epoch is a deterministic multi-trip team orienteering problem with time windows (MTTOPTW).

The MTTOPTW belongs to the class of orienteering problems (OP) and is an extension of the team orienteering problem with time windows (TOPTW). A survey of literature for the OP and its variants is presented in Vansteenwegen et al. (2011). To the best of the authors' knowledge, the MTTOPTW has not been presented in the literature.

In the TOPTW, associated with each customer is a reward and a time window. A team of vehicles is dispatched from the depot to collect the largest possible reward from a subset of customers. If a reward is collected at a customer, that reward must be collected during the customer's associated time window. If the vehicle arrives early at a customer, then the vehicle must wait until the beginning of the time window to

begin collecting the reward. The time it takes to collect the reward is considered a service time. All vehicles are required to return to the depot by a depot deadline. Unlike the TOPTW, in the MTTOPTW, each vehicle is permitted to make multiple trips from the depot before returning for the final time. Also new in the MTTOPTW is a ready time associated with each customer. A vehicle cannot visit a customer on a trip if the time of the start of the trip occurs before the ready time associated with the customer.

We formulate the MTTOPTW as a mixed integer program using the following notation. Let the set of customers and the depot be indexed such that $i = 0, 1, \ldots, n, n+1$, where the depot is represented by 0 at the beginning of a trip and $n+1$ at the end of the trip. For convenience, when we refer to customer i, we include 0 and $n+1$ (the depot) as possible values for i. Let R be the non-negative reward for serving a customer. Let e_i and l_i represent the start and end of the time window for customer i. We assume the time window for the depot represents the time at which trips are permitted to begin ($e_0 = e_{n+1}$) and the time by which all trips must be completed ($l_0 = l_{n+1}$). Let μ_i be the service time for customer i. Let τ_{ij} be the travel time between customer i and customer j, where $i, j = 0, 1, \ldots, n, n+1$. Let q_i be the ready time associated with customer i. Let the set of vehicles be indexed from $m = 1, \ldots, M$. Let H be the allowable number of possible trips for each vehicle and let the trips on a vehicle be indexed by $h = 1, \ldots, H$. Let $\mathcal{D}_m$ be a constant that represents the time that vehicle m is permitted to start the first trip. Let Y be a large constant.

We also use the following decision variables to formulate the MTTOPTW. Let $x_{ijmh} = 1$ if customer i is followed by customer j on vehicle m on trip h, and 0 otherwise. Let w_{imh} be the start time of the service at customer i on vehicle m on trip h.

The following assumptions are made to solve the MTTOPTW as the subproblem for the SDDP. The set of customers in the MTTOPTW is the set of realized requests and sampled requests in a scenario sample. The ready time is the realization time for the corresponding request. The parameter H is an upper bound on the number of trips that each vehicle needs to make in order to maximize the reward. We set H equal to the number of sampled requests plus one. If there are no sampled requests in the data set, then H is set to one in order to allow one trip for each vehicle. On the other hand, if there are sampled requests in the set, we allow one trip for each vehicle to serve realized requests and one trip for each of the sampled requests. Because H is an upper bound, the solution may include empty trips. The time that vehicle m is permitted to start the first trip, $\mathcal{D}_m$, is set to the earliest time that vehicle m is permitted to depart from the depot in the SDDP model, given the current epoch time. Note that these times may be different for each vehicle. We now present the integer program.

$$\max \quad \sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{h=1}^{H} R \sum_{j=1}^{n+1} x_{ijmh} \tag{3.39}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{0jmh} = \sum_{i=1}^{n} x_{i(n+1)mh} \leq 1, \quad m = 1, \ldots, M; h = 1, \ldots, H \tag{3.40}$$

$$\sum_{i=0}^{n} x_{ikmh} = \sum_{j=1}^{n+1} x_{kjmh}, \quad k = 1 \ldots, n; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.41}$$

$$w_{imh} + \mu_i + \tau_{ij} - w_{jmh} \leq Y(1 - x_{ijmh}),$$

$$i = 0, \ldots, n; j = 1, \ldots, n+1; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.42}$$

$$\sum_{j=1}^{n+1}\sum_{m=1}^{M}\sum_{h=1}^{H} x_{ijmh} \leq 1, i = 1, \ldots, n \tag{3.43}$$

$$w_{(n+1)m(h-1)} \leq w_{0mh}, \quad m = 1, \ldots, M; h = 2, \ldots, H \tag{3.44}$$

$$\mathcal{D}_m \leq w_{0m1}, \quad m = 1, \ldots, M \tag{3.45}$$

$$q_i \sum_{j=1}^{n+1} x_{ijmh} \leq w_{0mh}, \quad i = 1, \ldots, n; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.46}$$

$$\sum_{j=1}^{n} x_{0jmh} \leq \sum_{j=1}^{n} x_{0jm(h-1)}, \quad m = 1, \ldots, M; h = 2, \ldots, H \tag{3.47}$$

$$e_i \leq w_{imh}, \quad i = 0, \ldots, n+1; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.48}$$

$$w_{imh} \leq l_i, \quad i = 0, \ldots, n+1; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.49}$$

$$w_{imh} \geq 0, \quad i = 0, \ldots, n+1; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.50}$$

$$x_{ijmh} \in 0, 1,$$

$$i = 0, \ldots, n; j = 1, \ldots, n+1; m = 1, \ldots, M; h = 1, \ldots, H \tag{3.51}$$

The objective function 3.39, maximizes the total collected reward. Constraints 3.40 guarantee that all trips start and end at the depot. Constraints 3.41 ensure

the connectivity of each trip, and constraints 3.42 ensure feasible service start times based on prior customer start times. Constraints 3.43 ensure that every customer $(i \neq 0, n+1)$ is visited at most once. Constraints 3.44 require that the starting time for each trip begins after the ending of the previous trip. Constraints 3.45 set the earliest time each vehicle can begin the first trip. Constraints 3.46 ensure that the start time for each trip begins after the ready times for all customers on the trip. Constraints 3.47 are symmetry constraints and require that trips are associated with the lowest possible trip indices for the particular vehicle. Constraints 3.48 and 3.49 enforce time windows. Constraints 3.50 ensure the w_{imh} variables are positive, and constraints 3.51 define the x_{ijmh} variables as binary.

Because orienteering problems in general are challenging to solve exactly and our problem introduces the added challenge of multiple trips, we implement a variable neighborhood search (VNS) as our subproblem optimization method. A standard implementation of VNS is described in Section 2.2. The standard implementation of VNS is designed for a single vehicle, where a vehicle route is encoded as an ordered list of requests. The MTTOPTW is a multi-vehicle problem, and therefore we encode all of the vehicle routes associated with a routing plan into one ordered list of requests. We include dummy requests to separate the different vehicle routes.

Another implementation issue concerns the multi-trip aspect of the subproblem. Some sampled requests have request times that occur in the future. This means that at the current time, they are not available for pick up from the depot. The subproblem optimization therefore needs to schedule times for vehicles to return to

the depot. We handle this scheduling issue in the VNS by hard-coding returns to the depot into the objective function. For example, if the next scheduled request has not been picked up from the depot, the objective calculation will include the travel time to the depot from the the vehicle's current location, the time that the vehicle spends waiting at the depot for the sampled request's ready time to occur, the travel time from the depot to the sampled request, and the waiting time if the vehicle arrives before the beginning of the time window.

We solve a relaxation of subproblem instances by allowing deadline violations. A penalty of one is used for deadline violations of requests, while a large penalty is applied for deadline violations at the depot. A two-tier objective function is implemented. The primary objective is to minimize the number of requests with violated time constraints. The secondary objective is to minimize total route duration. The VNS terminates if the solution has not been updated in the last five iterations. Such a relaxation is necessary to reduce the otherwise large solution space, and thus reduce the computation time. If deadline violations were not allowed, the VNS would need to search a solution space with routing plans containing many combinations of subsets of the full set of requests. A single subset of requests can have many different vehicle assignments and routing orderings. Therefore, it would be difficult to find the largest subset of requests that can be feasibly served. We impose the secondary objective of minimizing the total route duration because we assume that by minimizing the duration, there will be greater opportunity in the future to serve additional requests.

3.2.2 Consensus Function

For each scenario sample routing plan, we use the consensus function to choose the 'best' solution. Defining a good consensus function is difficult. A classical example of a consensus function is presented by Bent and Van Hentenryck (2004). The problem presented by Bent and Van Hentenryck (2004) is a dynamic multi-vehicle routing problem with time windows. The consensus function used by Bent and Van Hentenryck (2004) identifies the plan that is most similar to the other plans. In Bent and Van Hentenryck (2004), the most similar plan is based on the next customer for which each vehicle in the plan will depart.

For Bent and Van Hentenryck's problem, it is intuitively appealing to use such a consensus function because selecting a plan that is most similar to all other plans means vehicle routes should be able to accommodate a robust set of future scenarios. The results in Bent and Van Hentenryck (2004) demonstrate that the sample scenario planning approach with their consensus function achieves significantly better results compared to methods that do not incorporate information about future requests. For these reasons, we also use a consensus function that selects the most similar plan to all other plans. However, we adapt the consensus function to fit the unique characteristics of the SDDP.

Because all requests need to be picked up from the depot before the delivery can be made, the SDDP is more constrained than Bent and Van Hentenryck's problem. In the SDDP, requests accumulate while waiting at the depot to be picked up. For this reason, several requests may be assigned to a vehicle and loaded at one time.

This leads to two questions:

1. Should a vehicle leave the depot immediately or wait for more requests to arrive?
2. If a vehicle is going to leave the depot immediately, which requests should it load?

Our consensus function heuristically answers both of these questions. In contrast, the consensus function of Bent and Van Hentenryck (2004) helps to answer the simpler question of which one customer should be scheduled next for each vehicle.

We present our consensus function in Algorithm 3.2. The consensus function takes *all_ssp* and $\mathcal{M}'$ as input. The list *all_ssp* contains all the scenario sample plans and $\mathcal{M}'$ is the set of vehicles located at the depot. The initialization consists of initializing an empty list called *all_scores*. The number of scenario samples is calculated by determining the length of list *all_ssp* and stored in *num_ss*.

Algorithm 3.2 Consensus

Input: *all_ssp*, $\mathcal{M}'$
Output: *Distinguished_Plan*
1: **Initialization:** *all_scores* = [], *num_ss* ←len(*all_ssp*)
2: **for** i in 1 to *num_ss* **do**
3: *score* ← Score(*all_ssp*[i], *all_ssp*, $\mathcal{M}'$)
4: *all_scores*.append(*score*)
5: i ← Select(*all_scores*)
6: *Distinguished_Plan* ← Construct(*all_ssp*[i])

In lines 2 through 4, each scenario sample plan is assigned a score by the function called Score. The function Score takes a scenario sample plan *all_ssp*[i] as

input as well as all_ssp and $\mathcal{M}'$ and assigns a real number, the score, to $all\_ssp[i]$. The resulting scores are stored in the list all_scores. The score assigned to each scenario sample plan is the mechanism by which the consensus function identifies the plan most similar to all other plans.

In line 5, Algorithm 3.2 calls the Select function on the list of all_scores. The Select function finds the highest score in the list and returns the index associated with the score. If there is a tie for the highest score, then one of the associated indices is selected at random.

In line 6, Algorithm 3.2 calls the Construct function that takes the scenario sample plan associated with the high score and returns a distinguished plan. The distinguished plan is used to select the next actions.

We now discuss the mechanics of the Score and Construct functions. The Score function maps a scenario sample plan onto the set of real numbers. The Score function counts the number of scenario sample plans that are similar to the scenario sample plan that is being evaluated. Two plans are considered similar if their partial plans are identical.

A partial plan consists of:

1. routes associated with vehicles located at the depot, and
2. the exact ordering of requests that occur on a route up to the first position on the route that is associated with a sampled request.

The first restriction makes sense because actions selected at the current decision epoch affect only vehicles located at the depot. The second restriction makes sense because

the presence of a sampled request on a route indicates that a new request is expected to arrive in the future. In order to serve any request that occurs in the future, vehicles are required to return to the depot to pick up such requests. Therefore, the position of the sampled request is indicating that it would be appropriate to schedule a return to the depot in that position. This indicates the end of the upcoming route segment. Therefore, decisions about requests that appear on a route after the sampled request can be considered at a later decision epoch. Note also that vehicles located at the depot are considered identical. Thus, within a routing plan, a route associated with a vehicle located at the depot can be associated with any vehicle that is located at the depot.

The Construct function takes the selected scenario sample plan and constructs the distinguished plan. To construct the distinguished plan, sampled requests are removed from routes associated with vehicles that are en route. For routes associated with vehicles that are located at the depot, the sampled request that occurs earliest in each vehicle route is replaced with the depot. Sampled requests that occur later in the route are removed. If there are no realized requests on the route, then all sampled requests are removed and the vehicle route will be empty. For all routes, the ordering of the realized requests are preserved from the scenario sample plan to the distinguished plan.

When constructing the distinguished plan, the depot may be placed in the first position of the vehicle route. If this occurs, the scenario samples are suggesting that a new request will arrive soon and it will be beneficial to wait at the depot to

see if a new request materializes. Therefore, the vehicle will be scheduled to wait at the depot for an additional unit of time. At the end of the time unit, a new decision epoch occurs.

The distinguished plan is used to explicitly identify when a vehicle should wait at the depot in anticipation of future requests rather than departing immediately. The information also determines a loading policy that allows vehicles to leave behind requests that could be served later by either the same vehicle or another vehicle.

Given that our subproblem solution method presented in Section3.2.1 solves a relaxation of the original problem, it may be necessary to adjust the distinguished plan so that all deadlines are met for requests that are scheduled to leave the depot. The depot deadline also needs to be maintained. Therefore, before a vehicle leaves the depot, the route is checked for deadline violations. A request with a deadline violation is removed from the route segment and temporarily placed on a different route segment to see if it can be feasibly served at a later time. This procedure is repeated until there are no request deadline violations that would occur. In the situation that the depot deadline is violated, requests are similarly removed until there is not a depot deadline violation.

We reiterate that the consensus function is used to select actions. The consensus function explicitly determines when waiting at the depot is beneficial based on probabilities of future request arrivals. However, the MDT presented in Section 3.1.6 is an implicit departure rule based only on a sequence of requests. We augment the decision making of the consensus function with the MDT. Once the consensus

function selects an action that assigns a non-empty set of requests to a vehicle, the MDT can be applied to determine whether waiting will or will not decrease the ability to serve current and future requests.

To clarify how the consensus function works, we present an example. Table 3.1 displays three scenario sample plans and their associated partial plans. There are two vehicles in the example. The asterisk symbol next to vehicle V1 indicates that V1 is located at the depot. Vehicle V2 is en route. In the plans, realized requests are represented by a number. Sampled requests are represented by an 'S' followed by a number.

Table 3.1: An example using the consensus function to create the distinguished plan.

	Scenario Sample Plan	**Partial Scenario Sample Plan**	**Distinguished Plan**
Plan A	V1*: 2, 1, S1, 4 V2: S2, 3	V1*: 2, 1	V1*: 2, 1, 0, 4 V2: 3
Plan B	V1*: S3, 2, S4, 1, 4 V2: 3	V1*:	
Plan C	V1*: 2, 1, S5 V2: 3, 4	V1*: 2, 1	

For each scenario sample plan, the Score function returns a score by comparing the partial scenario sample plans. In this example, Plan A receives a score of 1 because the partial plan is identical to exactly one other partial plan. For the same reason, Plan C also receives a score of 1. Plan B receives a score of 0. Plans A and C are tied for the highest score, and therefore one of them is randomly selected. Suppose Plan

A is selected. The distinguished plan is constructed from the scenario sample plan associated with Plan A. To construct the distinguished plan, we first consider vehicle V1, which is located at the depot. Sampled request 'S1' is the sampled request that occurs in the earliest position on the route. Therefore, 'S1' is replaced by 0. Next, consider vehicle V2, which is en route. All sampled requests are removed from the route.

3.3 Computational Experiments

In this section, we present computational experiments using data sets with a variety of different location geographies and time constraints. Section 3.3.1 describes the construction of the data sets that are used in the experiments. We present our parameter setting procedure and results in Section 3.3.2. The final results are presented in Section 3.3.3.

3.3.1 Data Sets

The data sets for the SDDP are adapted from the well-known data sets of Solomon (2005) for the VRPTW as well as the extended Solomon data sets of Gehring and Homberger (1999). We use only the location information in these data sets and generate our own request times, request locations, and time windows. Each data set contains customer location geographies that are randomly dispersed (R), clustered (C), or both randomly dispersed and clustered (RC). We now describe our location data sets. For each geography, we create nine SDDP location data sets with 100 customer locations. We describe the SDDP location data sets for the R geography,

Table 3.2: Sources for SDDP R location data sets.

SDDP	**VRPTW Benchmark**		
Data Set	**Author**	**Data Set**	**# of Cust.**
R_1	Solomon	R	100
R_2	G & H	R1_2	200
R_3	G & H	R1_4	400
R_4	G & H	R1_6	600
R_5	G & H	R1_8	800
R_6	G & H	R2_2	200
R_7	G & H	R2_4	400
R_8	G & H	R2_6	600
R_9	G & H	R2_8	800

but the C and RC location data sets are similar. Table 3.2 summarizes the source data sets for the SDDP R location data sets. We use the Solomon R data set that contains 100 customers. We refer to this data set as R_1. Because there is only one unique Solomon R data set in terms of customer locations, we also use the extended Solomon data sets of Gehring and Homberger. We create eight SDDP data sets from the extended data sets by sampling 100 customer locations from each of two data sets with 200, 400, 600, and 800 customer locations. We refer to these data sets as R_2 through R_9. Including the R, C, and RC geographies, there are a total of 27 SDDP location data sets.

We translate all the distances to travel times, τ_{ij}, such that no customer is more than one hour from the depot. To do this, we first calculate the Manhattan distance between locations. Manhattan distances resemble the distances travelled when using the road structure in a city. To obtain the travel times in a data set, we adjust all the distances by the same factor such that the farthest customer from the

depot in the data set can be reached within one hour.

We use a Poisson process to generate the times of requests. Let I represent the number of locations. Then, λ_i represents the request arrival rate of location i, $i = 1, \ldots, I$. The arrival time of the next request is determined by the sum of arrival rates for all locations: $\lambda = \sum_{i=1}^{I} \lambda_i$. The inter-arrival times are distributed $\exp(\lambda)$. Because we are using a discrete-time model, the inter-arrival times are rounded to the nearest integer.

The location associated with the request arrival time is selected by drawing a random number u from the uniform distribution $U[0, 1]$. The cumulative probabilities are calculated based on the following formula:

$$P_i = \begin{cases} 0, & \text{if } i = 0, \\ \sum_{h=1}^{i} p_h, & i = 1, \ldots, I \end{cases}$$

where $p_h = \frac{\lambda_h}{\sum_{i=1}^{I} \lambda_i}$. Thus, the selected location is i if u is such that $P_{i-1} \leq u < P_i$.

We generate data sets with one of four time window types. One of the types is a 2-hour deadline, while the other three types are 1-hour time windows. We describe the construction of the time windows for each data set.

- TW.d: A 2-hour deadline is created by setting $e_n = rt_n$ and $l_n = e_n + 2$ hours.
- TW.f: Based on the request time, the time window begins at a fixed time in the future. A 1-hour time window is created by setting $e_n = rt_n + 1$ hour and setting $l_n = e_n + 1$ hours.
- TW.h: Based on the request time and L, a uniform distribution is used to select e_n from one of the remaining hours of the day. Time windows only begin on

the hour. A 1-hour time window is created by setting $l_n = e_n + 1$ hours. For example, if a request arrives six hours and 51 minutes into a nine-hour day ($L = 8$ and $\Delta = 1$), the uniform distribution is used to assign e_n to either 7 or 8. If 8 is selected, then $e_n = 8$ and $l_n = 9$. The construction of these time windows match those of Azi et al. (2012).

- TW.r: Based on the request time and L, a uniform distribution is used to select e_n. Unlike TW.h, e_n is allowed to fall between hours. This data set can be thought of as having randomly dispersed e_n times. A 1-hour time window is created by setting $l_n = e_n + 1$ hours.

Data sets are generated for full-day operations where requests arrive through the first eight hours ($L = 8$) and must return to the depot by the ninth hour ($L+\Delta = 9$) for TW.d, TW.h, and TW.r time window types and by the tenth hour ($L+\Delta = 10$) for type TW.f. Type TW.f is different because of the time window construction where e_n and l_n are allowed to equal hours 9 and 10, respectively. We also generate data sets for half-day operations. Requests arrive during the first three hours of the day ($L = 3$) and vehicles are required to return to the depot at the end of four hours ($L + \Delta = 4$) for the TW.d, TW.h, and TW.r time window types and five hours ($L + \Delta = 5$) for the TW.f type. We refer to these time window types as TW3.d, TW3.f, TW3.h, and TW3.r.

The SDDP data sets are named according to the time window type and the location data set. For example, TW.f_R_1 refers to the full-day operation schedule for time window type TW.f and location data set R_1.

For all of the location data sets, 25 random request streams are generated with request times and locations. For half-day operations, the first three hours of requests are taken from the full-day request streams. For each of the time window types for both full-day and half-day operations, time windows are generated for each request in the request streams. For each data set type, we report average results across the 25 trials.

Experiments that include sampling require additional trials for each data set type in order to construct the sample scenarios. We refer to trials used in sampling procedures as scenario samples. We run a series of parameter setting experiments in Section 3.3.2 to determine the number of scenario samples that we will use as well as the sampling horizon duration that we will use in all subsequent experiments. Scenario samples are generated from the same distributions as are used to generate requests for the 25 test scenarios.

3.3.2 Parameter Setting

In this section, we run a series of experiments to identify reasonable parameter settings. There are two parameters in which we are interested: the duration of the sampling horizon and the number of scenario samples. The duration of the sampling horizon indicates how far in the future sampled requests are generated for the scenario samples. We consider sampling horizons of a 1/2 hour, 1 hour, and 2 hours and 10, 25, and 50 scenario samples. For purposes of these parameter setting experiments, we use three vehicles and set $\lambda_i = 0.002$ requests per minute for all customer locations.

Table 3.3: Frequency table for the best combinations of sampling horizon (1/2, 1, and 2 hours) and number of scenario samples (10, 25, and 50).

	1/2, 10	1/2, 25	1/2, 50	1, 10	1, 25	1, 50	2, 10
Top 1	2	**3**	**3**	0	2	1	1
Top 2	**7**	5	6	1	2	2	1

This corresponds to an overall arrival rate of 0.2 requests per minute, meaning that the expected number of requests is 48 for half-day operations. We use time window types TW3.d, TW3.f, TW3.h, and TW3.r and location data sets R_1, C_1, and RC_1.

A frequency table of the best combinations of sampling horizon and number of scenario samples from each data set is presented in Table 3.3. Considering the best combination for each data set, there is a tie between a sampling horizon of a 1/2 hour with 25 scenario samples and a sampling horizon of a 1/2 hour with 50 scenario samples. Both combinations occur three times. Because there is no clear winning combination, we instead consider the best two combinations for each data set. Out of the best two combinations, a sampling horizon of a 1/2 hour performs well with the most frequent combination being a sampling horizon of a 1/2 hour with 10 scenario samples. Therefore, we select to run the remaining experiments with this setting.

One question that arises is why sampling horizons of 1 and 2 hours perform worse than the shorter 1/2-hour sampling horizon. We ran additional experiments that indicate that longer sampling horizons require a larger number of scenario samples. More scenario samples are required because a longer sampling horizon means

that the number of possible routing plans increases. Therefore, a larger number of scenario samples are needed for a consensus of the best plan. When more scenario samples are used, solution quality does not deteriorate as the sampling horizon lengthens. However, using less than 10 scenario samples resulted in poorer solution quality.

3.3.3 Results

For the results presented in this section, we define a set of default settings. The settings are as follows:

- full-day operations,
- 1/2-hour sampling horizon,
- 10 scenario samples, and
- 25 trials for each data set type.

Before discussing the experiments and results in detail, we describe the headings presented in the results tables. The column labelled "Avg Epoch CPU" displays the average CPU time per decision epoch. The "Delay" column indicates whether or not the MDT calculation is used. A "D" indicates that the MDT is used, and "ND" indicates that it is not used. In the remainder of this section, we refer the MDT calculation as the delay. For time window type TW.d, the delay was not used because the delay is always zero when e_n equals the request time. The "Sampling" column indicates whether or not sampling is used. An "S" indicates that sampling is used, and "NS" indicates that sampling is not used. The "% Filled" column indicates the

average percent of requests that are filled within their respective time windows. The column labelled "S-NS" displays the difference in the percent filled with sampling minus the percent filled without sampling. Differences are calculated separately for the D and ND cases. For example, consider Table 3.4 and data set TW.f_R. The difference between the ND and S case and the ND and NS case is $50.80 - 48.60 = 2.20\%$. Similarly, the column labelled "D-ND" displays the difference in the percent filled with the MDT minus the percent filled without the MDT. Again, consider data set type TW.f_R. The difference between the D and NS case and the ND and NS case is $49.66 - 48.60 = 1.06\%$. All results are rounded to two decimal places.

Section 3.3.3.1 shows baseline results using three and five vehicles. In Sections 3.3.3.2 and 3.3.3.3, we show the effects of using a variety of different numbers of vehicles and of different homogeneous arrival rates, respectively. In Section 3.3.3.4, we present results using heterogeneous request arrival rates.

3.3.3.1 Baseline Results

In this section, we present detailed results for three and five vehicles to observe basic patterns. In addition to using the default settings, we also use a request arrival rate of $\lambda_i = 0.002$ requests per minute for each customer location. With 100 customer locations, this arrival rate corresponds to an overall arrival rate of $\lambda = 0.2$ and an expected number of requests of 96 for the full day of operation. We use all the time window types. We use location data sets R, C, and RC numbers 1-9. Results include situations with and without sampling and with and without delaying depot departure

using the delay.

We first present results for the three vehicle case. In Table 3.4, we report average results for each data set type. Because we report averages across the nine data sets per type, we label average results by time window type and geography type only.

The percent of filled requests for time window types TW.d and TW.f are larger than for TW.h and TW.r. There are two things to note with the TW.d and TW.f types. The TW.d type has 2-hour deadlines while the other time window types have 1-hour time windows. The TW.f type has requests arriving up to nine hours, but due to the construction of the time windows, it has a 1-hour extended day compared to the other time window types. However, the TW.h and TW.r types have significantly lower fulfillment percentages. The data sets are constructed in such a way that many time windows occur towards the end of the day. Due to time constraints and the number of vehicles, it is likely that the subset of requests that can be feasibly served is small in comparison to the total number of requests. Therefore, given that the number of vehicles is set to three, the optimal fulfillment rates for TW.h and TW.r are lower than for TW.d and TW.f.

In all cases, the percent of filled requests is larger when sampling is used compared to not using sampling. Sampling has a larger effect on the TW.h and TW.r time window types compared to the TW.d and TW.f types. Sampling provides an average improvement of 1.18% for the TW.d and TW.f types, while for the TW.h and TW.r types, sampling provides an average improvement of 7.73%. One explanation is

Table 3.4: Summary results for three vehicles.

Data Set	Avg Epoch CPU	Delay	Sampling	% Filled	S-NS	D-ND
TW.d_R	1.77	ND	NS	50.29	-	-
	53.87	ND	S	51.96	1.67	-
TW.d_C	2.24	ND	NS	50.62	-	-
	52.39	ND	S	52.38	1.77	-
TW.d_RC	1.65	ND	NS	50.41	-	-
	46.16	ND	S	51.68	1.27	-
TW.f_R	1.96	ND	NS	48.60	-	-
	59.79	ND	S	50.80	2.20	-
	1.59	D	NS	49.66	-	1.06
	74.35	D	S	51.32	1.67	0.53
TW.f_C	2.05	ND	NS	48.91	-	-
	51.81	ND	S	51.39	2.48	-
	1.53	D	NS	50.11	-	1.20
	70.21	D	S	51.92	1.81	0.53
TW.f_RC	1.67	ND	NS	48.31	-	-
	52.12	ND	S	50.33	2.02	-
	1.34	D	NS	49.42	-	1.10
	64.68	D	S	50.81	1.39	0.48
TW.h_R	5.92	ND	NS	26.41	-	-
	200.41	ND	S	36.06	9.65	-
	7.44	D	NS	30.16	-	3.74
	394.16	D	S	36.54	6.38	0.47
TW.h_C	6.51	ND	NS	26.14	-	-
	169.17	ND	S	36.04	9.90	-
	8.49	D	NS	30.58	-	4.44
	383.80	D	S	36.31	5.73	0.27
TW.h_RC	4.95	ND	NS	25.51	-	-
	156.34	ND	S	35.44	9.93	-
	4.26	D	NS	30.39	-	4.88
	338.79	D	S	36.00	5.61	0.56
TW.r_R	5.64	ND	NS	27.43	-	-
	154.59	ND	S	37.55	10.13	-
	4.87	D	NS	32.05	-	4.62
	343.33	D	S	37.52	5.47	-0.03
TW.r_C	6.30	ND	NS	28.05	-	-
	169.03	ND	S	37.78	9.73	-
	6.47	D	NS	32.46	-	4.41
	384.35	D	S	37.89	5.43	0.10
TW.r_RC	6.27	ND	NS	27.87	-	-
	203.44	ND	S	37.64	9.77	-
	3.25	D	NS	32.31	-	4.44
	313.11	D	S	37.31	4.99	-0.33

that the TW.d and TW.f types are less constrained. The deadlines and time windows are spread evenly throughout the day because they correspond directly to the time of the request arrivals. Thus, it is easier to find routes that feasibly serve many requests. In these situations, the amount of requests that can be feasibly served is constrained more by the number of vehicles than by time windows.

In the TW.h and TW.r time window types, many requests' time windows occur near the end of the day. This means many requests have overlapping time windows, making it difficult to serve all of the requests within their respective time windows. Without the use of sampling, vehicles leave the depot with all the requests that can be feasibly served on the trip. The routing plans that are constructed when sampling is used, however, allow vehicles to leave on the current trip with a smaller number of requests. Shorter trips at the current time occur as a result of anticipating the possibility for better routing in the future. Sampling provides information about when vehicles should return to the depot based on expected request arrivals. This means that the requests served on route segments and the duration of the route segments are dynamically adjusted during the day based on current information and expected future requests.

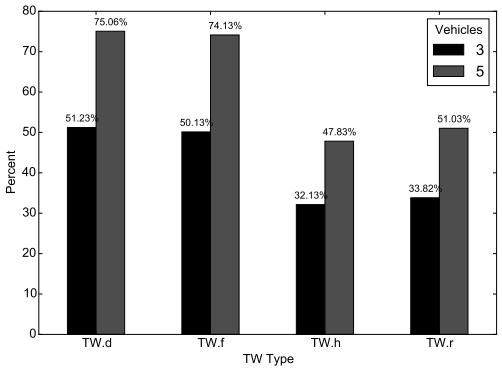
The gains made by the delay in addition to sampling are close to zero and sometimes negative. Therefore, sampling does a good job of delaying when it is valuable. In addition, the average CPU time per epoch is 262.98 seconds using both sampling and the delay compared to 114.09 seconds for sampling alone. This is a difference that is more than two times greater when using both sampling and the

delay compared to using sampling alone. Thus, the delay does not add additional value when used in conjunction with sampling.

Using the delay without sampling for the TW.f time window type offers a similar improvement of the percent filled compared to sampling alone, but with significantly reduced computation times. With sampling alone, the average CPU time is 54.57 seconds compared to 1.49 seconds with delay alone. For the TW.h and TW.r types, using sampling alone results in an average percent filled improvement of 9.85% while the average CPU time per epoch is 175.5 seconds. Using the delay without sampling offers an average percent filled improvement of 4.42% and an average CPU time of 5.80 seconds. Therefore, the delay offers reasonable improvements in terms of the percent filled without the computational burden of sampling. This suggests that the delay could be used in place of sampling when faster computation times are necessary.

The structure of the time windows in the TW.h and TW.r types plays a significant role in why the delay increases the percent of filled requests by a large amount. Without the delay, a vehicle may be sent at the beginning of the day to serve requests that have time windows that occur near the end of the day. The late-occurring time windows force the vehicle to idle at the customer location, thus preventing the vehicle from serving other requests. Using the delay, however, the vehicle is allowed to wait at the depot location until requests that have earlier time windows arrive. Therefore, the delay defers routing decisions about requests that cannot be served until later. The result is that more urgent requests can be served

Figure 3.1: Average percent of filled requests for three and five vehicles.



when they arrive, thus increasing the overall percent of filled requests.

We now present results for five vehicles. Results for R, C, and RC numbers 1 through 9 are displayed in Table 3.5. The most noticeable difference between using three vehicles and five vehicles is that the percent of requests filled when using five vehicles is larger for five vehicles. This results from keeping the arrival rate the same but increasing the number of vehicles. A comparison of the percent of filled requests of three vehicles versus five vehicles broken down by time window type is is displayed in Figure 3.1. The other trends observed for the three vehicle case extend to the five vehicle case.

3.3.3.2 Number of Vehicles

In this section, we consider how the number of vehicles affects the results for one to thirteen vehicles. We use the default settings and run R_1 and C_1 for all time window types. The request arrival rate for each customer location, λ_i, is set to 0.002 requests per minute, an equivalent of 96 expected requests per day. Figure 3.2 shows

Table 3.5: Summary results with five vehicles.

Data Set	Avg. Epoch CPU	Delay	Sampling	% Filled	S-NS	D-ND
TW.d_R	1.56	ND	NS	74.51	-	-
	44.64	ND	S	75.66	1.15	-
TW.d_C	1.36	ND	NS	74.58	-	-
	42.81	ND	S	75.90	1.32	-
TW.d_RC	1.44	ND	NS	74.43	-	-
	44.14	ND	S	75.25	0.82	-
TW.f_R	1.48	ND	NS	73.00	-	-
	50.62	ND	S	74.37	1.36	-
	1.33	D	NS	73.87	-	0.86
	63.40	D	S	75.33	1.47	0.97
TW.f_C	1.31	ND	NS	73.24	-	-
	47.45	ND	S	74.76	1.52	-
	1.16	D	NS	74.35	-	1.10
	58.80	D	S	75.33	0.99	0.57
TW.f_RC	1.42	ND	NS	72.44	-	-
	48.83	ND	S	74.24	1.79	-
	1.23	D	NS	73.74	-	1.29
	61.85	D	S	74.95	1.21	0.71
TW.h_R	3.94	ND	NS	42.06	-	-
	181.57	ND	S	51.11	9.05	-
	5.99	D	NS	46.58	-	4.52
	418.51	D	S	52.11	5.53	1.00
TW.h_C	3.85	ND	NS	41.48	-	-
	164.23	ND	S	51.11	9.63	-
	5.38	D	NS	47.45	-	5.98
	414.61	D	S	52.36	4.91	1.26
TW.h_RC	4.03	ND	NS	40.68	-	-
	177.48	ND	S	50.69	10.01	-
	5.74	D	NS	46.76	-	6.08
	448.24	D	S	51.61	4.85	0.92
TW.r_R	3.59	ND	NS	44.09	-	-
	168.16	ND	S	54.43	10.33	-
	4.04	D	NS	49.96	-	5.87
	351.74	D	S	54.92	4.95	0.49
TW.r_C	3.34	ND	NS	44.63	-	-
	157.36	ND	S	54.93	10.30	-
	3.99	D	NS	49.90	-	5.27
	339.59	D	S	54.99	5.09	0.06
TW.r_RC	3.71	ND	NS	44.92	-	-
	175.25	ND	S	54.37	9.46	-
	4.47	D	NS	50.35	-	5.44
	369.89	D	S	54.88	4.52	0.50

the average results. Subfigure 3.3(a) shows the average percent of filled requests as the number of vehicles increases. For a 95% fulfillment rate, a company needs 11 vehicles using the TW.d time window type, and eight vehicles using the TW.f type. The number of vehicles needed to achieve a 95% service level for both the TW.h and TW.r types is much larger compared to the number needed for the TW.d and TW.f types. The results demonstrate that with 13 vehicles, just under an 85% fulfillment rate can be reached for both the TW.h and TW.r types.

Regarding the differences in the fulfillment rates of the TW.d and TW.f types, it is important to recall that data sets with the TW.f time window type have a 10-hour depot deadline while the other time window types have a 9-hour depot deadline. This extra hour in the day accounts for the fulfillment rate gap between TW.d and TW.f. Additional experiments where the depot deadline is increased to ten hours for the TW.d type supports this conclusion. It is valuable for practitioners to understand the circumstances under which adding additional operational time can be beneficial. In this case, the additional hour would allow for a drastic reduction in the number of vehicles required to serve 95% of all requests.

Continuing to look at the fulfillment curves in Subfigure 3.2(a), the curves for time window types TW.d and TW.f are similar, while the curves for time window types TW.h and TW.r are almost identical. Looking at the TW.d and TW.f types, adding an additional vehicle has a large pay-off until reaching a certain number of vehicles. For the TW.d and TW.f types, this point is at eight and seven vehicles, respectively. When adding an additional vehicle beyond these points, the law of

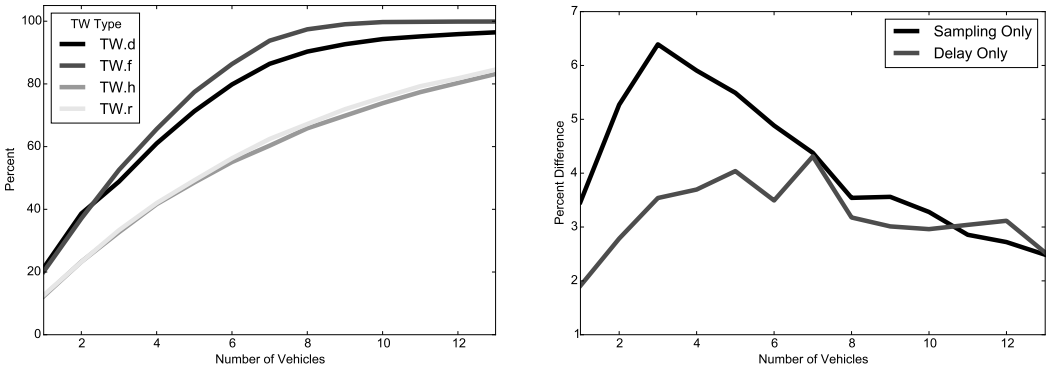
diminishing returns takes effect. A practitioner would not add an additional vehicle when cost of adding that vehicle is larger than the returns of the vehicle.

For the TW.h and TW.r time window types, the curve appears linear. This suggests that there is not a decreasing marginal return for each additional vehicle up to 13 vehicles. The prevalence of overlapping time windows, which make it difficult to feasibly serve customers, is likely the reason for a relatively fixed return when an additional vehicle is added.

Based on the results in this section, it is beneficial for companies to promise dissimilar delivery times to customers and to adopt a time window scheme similar to the TW.d and TW.f time window types. This is because fewer vehicles are necessary to achieve higher fulfillment rates. In some situations, it may be necessary to use the TW.h and TW.r time window types. In such situations, one option is to hire additional vehicles to serve the end-of-the-day demand. The additional vehicles at the end of the day can help serve requests that would otherwise be infeasible to serve, but without the additional cost of hiring a full-time vehicle.

We also examine the value of sampling and the delay as the number of vehicles increases. Subfigure 3.2(b) displays the average percent improvement across all time window types for sampling without the delay in one series and the delay without sampling in the other series. Considering the sampling only series, the value from sampling initially increases as the number of vehicles increases. At three vehicles, the value of sampling begins to decrease as the number of vehicles increases. This trend is observed because initially vehicle capacity is limited, and thus the improvement

Figure 3.2: Average results as the number of vehicles increases for data sets R_1 and C_1.



(a) Percent of requests filled as the number of vehicles increases.

(b) Percent improvement with sampling only and delay only as the number of vehicles increases.

from sampling is also limited. With a certain ratio of requests to vehicles, sampling helps to build efficient routes. However, as more and more vehicles are added but the number of requests remains fixed, the need for efficient routes decreases because there is more flexibility in how those requests are served. A similar trend is seen when delay only is used. Thus, the value of future information increases or decreases depending on the ratio of requests to vehicles.

3.3.3.3 Homogeneous Arrival Rates

We also consider the effect of sampling when different homogeneous arrival rates are used. We use the R and C data sets numbers 1 through 9. We set λ_i to 0.001, 0.002, 0.003, or 0.004 requests per minute for each customer location in the

Table 3.6: Request arrival rates and the corresponding expected total number of requests per day.

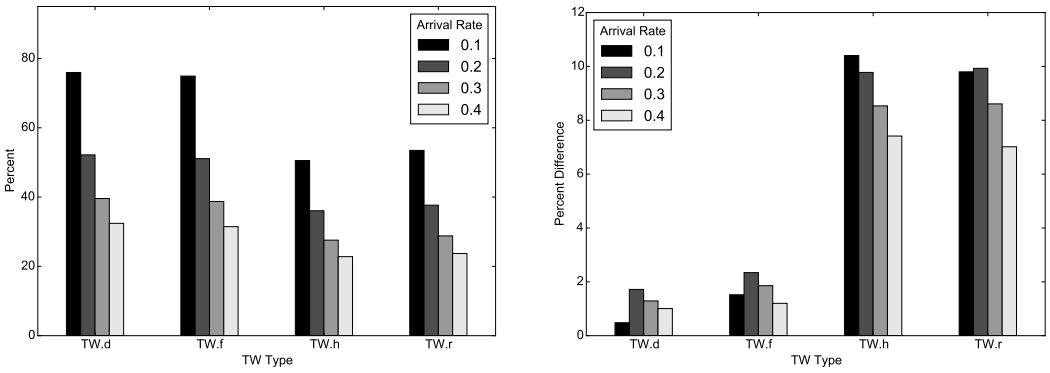
Location Arrival Rates (λ_i) (Requests per Min.)	**Overall Arrival Rate (λ) (Requests per Min.)**	**Expected Total Number of Requests**
0.001	0.1	48
0.002	0.2	96
0.003	0.3	144
0.004	0.4	192

data set. These arrival rates correspond to overall expected number of requests of 48, 96, 144, and 192, respectively. A table describing these arrival rates is displayed in Table 3.6. We use the default settings with three vehicles.

Figure 3.3 displays the average results for the homogeneous arrival rate experiments with sampling only. Subfigure 3.4(a) shows the percent of filled requests using sampling only. As the arrival rate increases, the percent of requests filled decreases. This occurs because the number of requests increases while the number of vehicles remains fixed. Therefore, the proportion of requests that can be served with the given capacity as the arrival rate increases tends to decrease.

Subfigure 3.3(b) displays the percent differences when using sampling only compared to using neither sampling nor delay. There is a distinct pattern that appears for the TW.d, TW.f, and TW.r time window types. Initially, the percent difference increases and then begins to decrease as the arrival rate increases. With the 0.1 overall arrival rate, sampling improves the solutions by only a small amount because there are few requests compared to the available number of vehicles. With a high

Figure 3.3: Average results for homogeneous arrival rates with sampling only.



(a) Percent of requests filled with sampling only.

(b) Percent difference of sampling only compared to not using sampling or delay.

overall arrival rate of 0.4, the percent improvement from sampling is also small, but the small impact is due to the large number of requests compared to vehicles. When more requests are available, there is more flexibility in how routing can occur, and therefore decreases the value of including sampling in the solution approach. Such results support the results from Section 3.3.3.2 where we discussed results for a range of numbers of vehicles. The TW.h time window type demonstrates a pattern of decreasing percent differences as the arrival rate increases. This different pattern likely occurs because of the structure of the time windows. The TW.h type is the only time window type that has time window start times that occur on the hour. Because there are less options for time windows, scenario samples represent scenarios that are close to the actual realizations, thus increasing the performance of sampling. We note that the TW.r type has a similar structure near the end of the day. We see

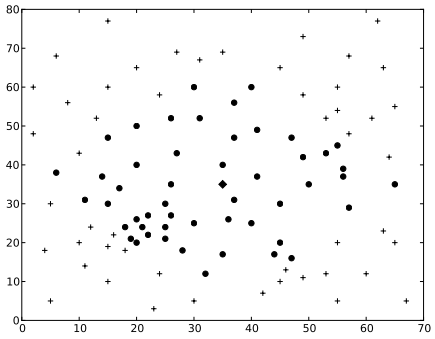
a reflection of this in the TW.r results because the trend is very close to a purely decreasing trend.

3.3.3.4 Heterogeneous Arrival Rates

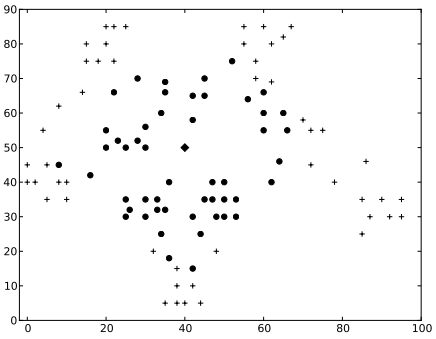
All of our previous experiments have used homogeneous arrival rate settings. We now discuss results for experiments using heterogeneous arrival rate settings. We choose heterogeneous arrival rate settings such that the overall arrival rate, λ, is equal to 0.2 requests per minute. This arrival rate corresponds to an expected number of requests of 96 requests per day. This overall arrival rate is consistent with previous experiments, except those presented in Section 3.3.3.3. For all heterogeneous experiments, we assign 50 customer locations to have arrival rates of $\lambda_i = 0.001$ requests per minute (low) and 50 locations to have arrival rates of $\lambda_i = 0.003$ requests per minute (high). We use the R and C data sets numbers 1 through 9.

We assign arrival rates to locations in four ways. We refer to each of these ways as heterogeneous types 1, 2, 3, and 4. For heterogeneous type 1, we randomly assign the high and low arrival rates to customer locations. In the remaining experiments, we create two clusters of 50 customer locations each. The clusters are shown in Figure 3.4. In each subfigure, the depot is represented by a diamond near the center of the space. For heterogeneous types 2 and 3, one cluster is formed with the locations nearest to the depot and the other cluster is formed with the locations farthest from the depot. For type 2, we assign the low arrival rates to locations near the depot and high arrival rates to locations far from the depot. For type 3, we reverse the

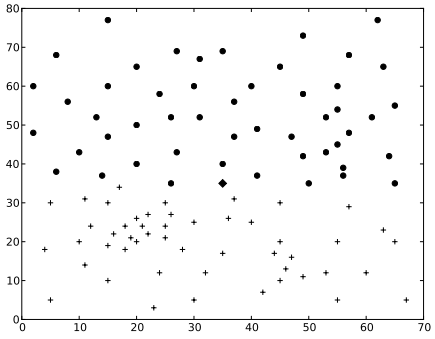
Figure 3.4: Clusters for heterogeneous types 2 through 4.



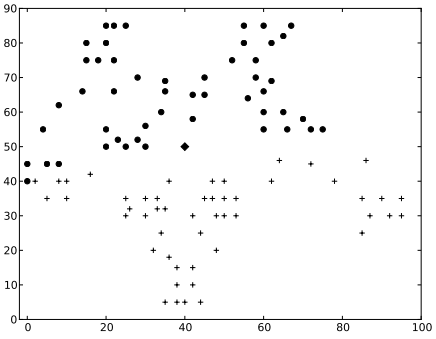
(a) R_1, heterogeneous types 2 and 3



(b) C_1, heterogeneous types 2 and 3



(c) R_1, heterogeneous type 4



(d) C_1, heterogeneous type 4

Table 3.7: Summary of homogeneous and heterogeneous arrival rate types.

Type	**Description**
Homogeneous	All rates equal for each location
Heterogeneous Type 1	Random
Heterogeneous Type 2	High rates for locations far from the depot
Heterogeneous Type 3	High rates for locations near the depot
Heterogeneous Type 4	High rates "above" the depot

arrival rates so that locations that are near to the depot have high arrival rates, and locations that are far from the depot have low arrival rates. For heterogeneous type 4, we divide customers into two clusters using a sweep algorithm (see Gillett and Miller (1974)). As a result, one cluster is approximately located above the y-coordinate of the depot, and the other cluster is approximately located below the y-coordinate of the depot. We assign high arrival rates to locations in the upper cluster and low arrival rates to locations in the lower cluster. For convenience, we summarize the heterogeneous types in Table 3.7 and include the homogeneous case for comparison.

We present a summary of results in Figure 3.5. In addition to the heterogeneous experiments, we compare results to the homogeneous results with data sets R and C numbers 1 through 9 with $\lambda = 0.2$. Subfigure 3.5(a) shows the percent of filled requests with sampling only. For the homogeneous case, the percent filled is 44.25%. In comparison, the average percent filled for heterogeneous types 1 through 4 are 43.32%, 39.47%, 48.41%, and 44.41%, respectively. Heterogeneous types 1 and 4 do not differ significantly from the homogeneous experiments. However, heterogeneous

type 2 has a lower percent of filled requests, while type 3 has a higher percent of filled requests. This is expected because type 2 has high arrival rates for locations far from the depot while type 3 has high arrival rates for locations near the depot. More requests can be served when the locations of the requests are near the depot because less time is necessary for travel.

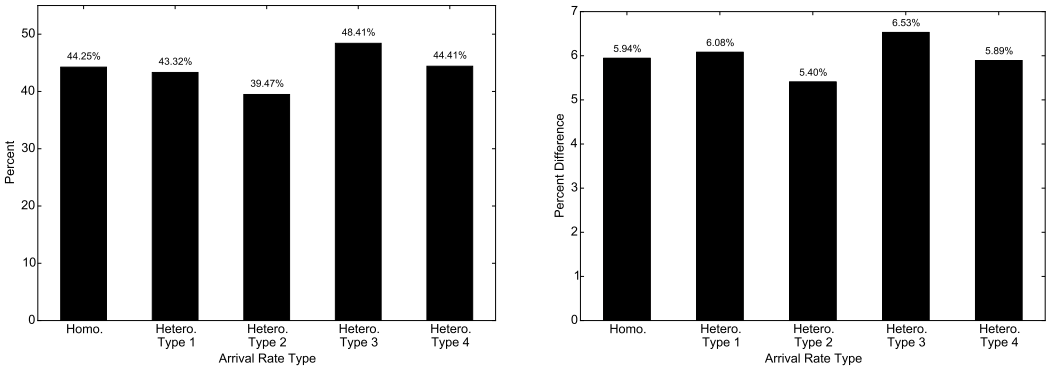
We also consider the average percent difference between using sampling alone compared to not using sampling or the delay. The results are presented in Subfigure 3.5(b). For the homogeneous experiments, the average difference is is 5.94%. For heterogeneous types 1 through 4, the average improvements are 6.08%, 5.40%, 6.53%, and 5.89%, respectively. We see that Heterogenous type 2 has the least amount of improvement from sampling while heterogenous type 3 has the most amount of improvement from sampling. These results indicate that the location of high probability customers can play a significant role on the impact that sampling has on solutions.

3.4 Conclusion

In this chapter, we introduce a dynamic programming model for the SDDP, a dynamic and stochastic pick-up and delivery problem. At each decision epoch, a subproblem of the SDDP is solved. This subproblem is called the MTTOPTW. The MTTOPTW is a new problem in the literature, and therefore, we present a mixed integer programming formulation for the MTTOPTW.

We incorporate two ways to make better-informed decisions at decision epochs. A common implicit rule in the literature is to delay vehicle departures when possible.

Figure 3.5: Average results for homogeneous and heterogeneous arrival rates with sampling only.



(a) Percent of requests filled with sampling only.

(b) Percent difference of sampling only compared to not using sampling or delay.

In delaying vehicle departures, decisions are postponed until a later time when more information about the problem is realized. We introduce the MDT calculation, which determines how long a vehicle can wait at the depot without decreasing the ability to serve current and future requests.

We use a sample scenario planning method as another way to make better-informed decisions. In sample scenario planning, different scenarios are constructed based on random samples of the stochastic parameter set. The stochastic element in the SDDP is customer request arrivals. Based on these scenarios, routing decisions are made. We introduce a consensus function specifically designed for the SDDP that takes advantage of the sampled information to guide vehicle route construction such that vehicles are able to accommodate more future requests once the requests are

realized. Our consensus function identifies when waiting at the depot in anticipation of future requests is beneficial, as well as selecting which requests should be assigned to specific vehicles to be served immediately while delaying decisions about other requests.

We run experiments for a variety of different location geographies, time constraints, number of vehicles, and request arrival rates. Our results offer several insights. Overall, more requests can be filled when time windows occur near the time that requests are made (TW.d and TW.f time window types) compared to when many time windows occur late in the day (TW.h and TW.r time window types). This is because it is more difficult to feasibly serve requests with respect to time windows when many time windows occur near the same time. However, sampling and the MDT delay are most valuable when many time windows occur late in the day. From our three vehicle baseline experiments for time window types TW.h and TW.r, sampling without the MDT delay offers an average increase of filled requests of 9.85%, while using the MDT delay without sampling offers an average increase of 4.42%. While the quality of the solutions with the MDT delay is not as high compared to sampling, the MDT delay can be used when faster computational times are required. Using sampling in conjunction with the MDT delay does not result in solutions that are better than using sampling alone, yet it requires significantly longer computation times. The results suggest that sampling delays making routing decisions when it is beneficial to do so.

Additional experiments demonstrate that the value of sampling and the MDT

delay decreases when either the number of vehicles increases or the arrival rate of requests increases. This is a result of having increased flexibility. When the number of vehicles is large, there is more capacity, and thus more flexibility in choosing how to serve the requests. When the arrival rate of requests is high, there is more flexibility in choosing which requests to serve, based on a limited capacity. These increases in flexibility decreases the value of future information.

Heterogeneous experiments demonstrate that the location of high arrival rate customers in relation to the depot has a large impact on the percent of requests that are filled and the improvements achieved through sampling. In particular, it is beneficial to have high arrival rate customers located near the depot, while it is undesirable to have high arrival rate customers located far from the depot.

CHAPTER 4
SUMMARY AND FUTURE WORK

Motivated by the high demand for last-mile express package delivery, we model and present solution methodologies for stochastic vehicle routing problems with time constraints. We consider two problems, the PTSPTW and the SDDP, and propose different solution approaches that incorporate probabilistic information for each problem. These solutions can help reduce the operational costs of delivery while improving customer service.

The PTSPTW is the problem of finding an expected minimum cost a priori route through a set of customers with probabilities of requiring service on any given day. Also associated with each customer is a time window during which service must occur. We present computational results that characterize the PTSPTW solutions in comparison to the TSPTW, the deterministic analogue of the PTSPTW. The results can help practitioners determine the when complexity of the solving the stochastic problem might be preferred to solving the deterministic problem.

One difficulty with the PTSPTW is the computational complexity of the objective function. For the related PTSPD, Weyland et al. (2013) use an approximation for the objective function based on a sampling procedure and parallel evaluation of the samples. This method outperforms previous approaches for the PTSPD in terms of computation time and solution value. In the future, we would like to apply these techniques to the PTSPTW. By reducing the computational time to solve the PTSPTW, we will be able to solve larger instances, as would be used in real-world

delivery applications.

The SDDP is a pick-up and delivery problem where customer delivery requests arrive dynamically during the day according to a known probability distribution. We propose a dynamic programming model and use a sample scenario planning solution approach. At each decision epoch a subproblem called the multi-trip team orienteering problem with time windows is solved. We introduce this problem and present an integer programming model for it. We introduce a consensus function that uses the sampled information to aid vehicle loading decisions as well as vehicle departure decisions when the vehicle is at the depot. We also introduce an analytical result that determines how long a vehicle can wait at the depot without impacting solution quality. We present computational experiments that demonstrate the value of using the sampled information and the value of our calculation for determining how long a vehicle should wait at the depot. The results can help practitioners determine when sampling may be valuable or to determine what type of delivery time window options customers should be offered.

There are many ways to extend the work on the SDDP. The heterogeneous experiments in this thesis have different request arrival rates based on location. We propose to introduce a new set of experiments that account for varying likelihood of customers to be assigned to different time windows. Sampling may prove to be even more useful if we know the time windows that are more likely to be chosen by different customers.

We propose extensions that incorporate additional real-world elements. Stochas-

ticity frequently occurs in the real-world. However, our current model only accounts for stochastic customer presence. We would also like to incorporate stochastic travel and service times or stochastic customer demands. Another occurrence in the real-world is rush-hour traffic, where travel times increase to varying degrees along certain routes. We could represent this in the model by including time-dependent travel times. The delivery of requests would be impacted because of the travel times.

Another extension would be to assign dollar values to requests to represent the revenue received when the request is served. When heterogeneous dollar values are assigned, these amounts could be used to help prioritize certain customers and increase revenues.

Like the PTSPTW, the SDDP is computationally expensive. We propose to parallelize the evaluation of the sample scenarios. The decrease in computation time from the parallelization will allow us to use larger instances and incorporate real-world data sets that may be large.

REFERENCES

Maria Albareda-Sambola, Elena Fernández, and Gilbert Laporte. The dynamic multiperiod vehicle routing problem with probabilistic information. *Computers & Operations Research*, 48:31–39, 2014.

Amazon. Same-day delivery. URL http://www.amazon.com/b?node=8729023011. Accessed on February 1, 2015.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1): 103–112, 2012.

Prasanna Balaprakash, Mauro Birattari, Thomas Stützle, Zhi Yuan, and Marco Dorigo. Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. *Swarm Intelligence*, 3(3):223–242, 2009.

Russell W. Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6): 977–987, 2004.

Dimitris J. Bertsimas and Garrett Van Ryzin. A Stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research*, 39(4):601–615, 1991.

Dimitris J. Bertsimas and Garrett Van Ryzin. Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles. *Operations Research*, 41 (1):60–76, 1993.

Christian Bierwirth, Dirk C. Mattfeld, and Herbert Kopfer. On permutation representations for scheduling problems. In *In 4th PPSN*, pages 310–318. Springer-Verlag, 1996.

Mauro Birattari, Prasanna Balaprakash, Thomas Stutzle, and Marco Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: A case study on the probabilistic traveling salesman problem. *INFORMS Journal on Computing*, 20(4):644–658, 2008.

Jürgen Branke, Martin Middendorf, Guntram Noeth, and Maged Dessouky. Waiting strategies for dynamic vehicle routing. *Transportation Science*, 39(3):298–312, 2005.

Ann M. Campbell and Barrett W. Thomas. PTSPD benchmark data sets, 2006. URL http://myweb.uiowa.edu/bthoa/PTSPDBenchmarkDataSets.htm. Accessed on November 22, 2011.

Ann M. Campbell and Barrett W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21, 2008.

Ann M. Campbell, Michel Gendreau, and Barrett W. Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, 2011.

Ann Melissa Campbell and Barrett W. Thomas. Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines. *Computers & Operations Research*, 36(4):1231–1248, 2009.

Tsung-Sheng Chang, Yat-wah Wan, and Wei Tsang OOI. A stochastic dynamic traveling salesman problem with hard time windows. *European Journal of Operational Research*, 198(3):748–759, 2009.

Kasarin Charnsirisakskul, Paul M. Griffin, and Pinar Keskinocak. Order selection and scheduling with leadtime flexibility. *IIE Transactions*, 36:697–707, 2004.

Si Chen, Bruce Golden, Richard Wong, and Hongsheng Zhong. Arc-routing models for small-package local routing. *Transportation Science*, 43(1):43–55, 2009.

Stephanie Clifford and Claire Cain Miller. Instantly yours, for a fee. *The New York Times*, page B1, December 28, 2012.

Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

Rodrigo Ferreira da Silva and Sebastián Urrutia. The traveling salesman problem with time windows (TSPTW) - approaches & additional resources, 2010. URL http://homepages.dcc.ufmg.br/~rfsilva/tsptw. Accessed on November 22, 2011.

Rodrigo Ferreira da Silva and Sebastián Urrutia. A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010.

Datamonitor. Express logistics industry profile: United States. Technical report, 2010. URL http://www.datamonitor.com. Accessed on May 13, 2011.

Yvan Dumas, Jacques Desrosiers, Eric Gelinas, and Marius M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

Jan Fabian Ehmke, André Steinert, and Dirk Christian Mattfeld. Advanced routing for city logistics service providers based on time-dependent travel times. *Journal of Computational Science*, 4(3):193–205, 2012.

Alan L. Erera, Martin Savelsbergh, and Emrah Uyar. Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks*, 54(4):270–283, 2009.

FedEx. Fedex annual report 2014, May 2014. URL http://annualreport.van.fedex.com/2014/docs/FedEx_2014_Annual_Report.pdf. Accessed on February 1, 2015.

FedEx Custom Critical, Inc. Surface mode optimization blended services freight forwarder service guide FDFF 201-C, 2005. URL http://customcritical.fedex.com/us/serviceinfo/documents/pdf/fdff201c.pdf. Accessed on November 3, 2011.

H. Gehring and J. Homberger. Extended SOLOMON's VRPTW instances, 1999.

Michel Gendreau, François Guertin, Jean-Yves Potvin, and Éric Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

Gianpaolo Ghiani, Emanuele Manni, Antonella Quaranta, and Chefi Triki. Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):96–106, 2009.

Billy E. Gillett and Leland R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

Justin C. Goodson, Jeffrey W. Ohlmann, and Barrett W. Thomas. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research*, 61(1):138–154, 2013.

Justin C. Goodson, Barrett W. Thomas, and Jeffrey W. Ohlmann. Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Science*, 2015. To appear.

Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Perez. Variable neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, 146, pages 61–86. Springer US, 2nd edition, 2010.

Rob Howard. Same-day delivery: A checklist for retailers seeking an antidote to Amazon: Part one, May 2014. URL http://www.parcelindustry.com. Accessed on February 1, 2015.

Lars Magnus Hvattum, Arne Løkketangen, and Gilbert Laporte. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49(4):330–340, 2007.

Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225, 2006.

Instacart. Frequently asked questions. URL https://www.instacart.com/faq. Accessed on January 22, 2015.

Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.

Patrick Jaillet, Jin Qi, and Melvyn Sim. Routing optimization under uncertainty. 2014. Working paper.

Guenter Kiechle, Karl F. Doerner, Michel Gendreau, and Richard F. Hartl. Waiting strategies for regular and emergency patient transportation. In *Operations Research Proceedings 2008*, pages 271–276, 2009.

Allan Larsen, Oli B. G. Madsen, and Marius M. Solomon. The a priori dynamic traveling salesman problem with time windows. *Transportation Science*, 38(4): 459–472, 2004.

Hongtao Lei, Gilbert Laporte, and Bo Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38 (12):1775–1783, 2011.

Xiangyong Li, Peng Tian, and Stephen C.H. Leung. Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, 125(1):137–145, 2010.

Yu-Hsin Liu. Diversified local search strategy under scatter search framework for the probabilistic traveling salesman problem. *European Journal of Operational Research*, 191(2):332–346, 2008.

Emanuele Manni. *Topics in real-time fleet management*. PhD thesis, Univerity of Calabria, 2007.

Yannis Marinakis and Magdalene Marinaki. A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem. In *IEEE Congress on Evolutionary Computation, 2009*, pages 1762–1769. IEEE, 2009.

Yannis Marinakis and Magdalene Marinaki. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers & Operations Research*, 37(3):432–442, 2010.

Yannis Marinakis, Athanasios Migdalas, and Panos M. Pardalos. Expanding neighborhood search–GRASP for the probabilistic traveling salesman problem. *Optimization Letters*, 2(3):351–361, 2008.

Stephan Meisel. *Anticipatory Optimization for Dynamic Decision Making*, volume 51 of *Operations Research/Computer Science Interfaces Series*. Springer New York, 2011.

Snežana Mitrović-Minić and Gilbert Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655, 2004.

Snežana Mitrović-Minić, Ramesh Krishnamurti, and Gilbert Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685, 2004.

Srimathy Mohan, Michel Gendreau, and Jean-Marc Rousseau. The stochastic Eulerian tour problem. *Transportation Science*, 42(2):166–174, 2008.

Rodrigo Moretti Branchini, Vinícius Amaral Armentano, and Arne Løkketangen. Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36(11):2955–2968, 2009.

Harilaos N. Psaraftis. Dynamic vehicle routing problems. In Bruce L. Golden and Arjang A. Assad, editors, *Vehicle Routing: Methods and Studies, Studies in Management Science and Systems*, volume 16, pages 223–248. North-Holland, Amsterdam, 1988.

Vitória Pureza and Gilbert Laporte. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR: Information Systems and Operational Research*, 46(3):165–176, 2008.

Doris Sáez, Cristián E. Cortés, and Alfredo Núñez. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers & Operations Research*, 35(11):3412–3438, 2008.

John D. Schultz. 2008 Is it trucking's bounce back year? *Logistics Management*, 47 (3):S65 – S71, 2008.

Patricia Kristine Sheridan, Erich Gluck, Qi Guan, Thomas Pickles, Barış Balcıog˜lu, and Beno Benhabib. The dynamic nearest neighbor policy for the multi-vehicle pick-up and delivery problem. *Transportation Research Part A: Policy and Practice*, 49:178–194, 2013.

Susan A. Slotnick and Matthew J. Sobel. Manufacturing lead-time rules: Customer retention versus tardiness costs. *European Journal of Operational Research*, 163(3): 825–856, 2005.

Marius Solomon. VRPTW benchmark problems, March 2005. URL http://w.cba.neu.edu/~msolomon/problems.htm. Accessed on February 1, 2015.

H. Tang and E. Miller-Hooks. Algorithms for a stochastic selective travelling salesperson problem. *Journal of the Operational Research Society*, 56:439–452, 2005.

April Terreri. Technology powers LTL leaders. *World Trade, WT 100*, 24(4):18–23, 2011.

Jano I. van Hemert and J.A. La Poutré. Dynamic routing problems with fruitful regions: Models and evolutionary computation. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 692–701. Springer Berlin Heidelberg, 2004.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

Stacy A. Voccia, Ann M. Campbell, and Barrett W. Thomas. The probabilistic traveling salesman problem with time windows. *EURO Journal on Transportation and Logistics*, 2(1-2):89–107, 2013.

Dennis Weyland, Leonora Bianchi, and Luca Maria Gambardella. New approximation-based local search algorithms for the probabilistic traveling salesman problem. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2009*, volume 5717 of *Lecture Notes in Computer Science*, pages 681–688. Springer Berlin Heidelberg, 2009.

Dennis Weyland, Roberto Montemanni, and Luca Maria Gambardella. Hardness results for the probabilistic traveling salesman problem with deadlines. In A. Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Th. Paschos, editors, *Combinatorial Optimization*, volume 7422 of *Lecture Notes in Computer Science*, pages 392–403. Springer Berlin Heidelberg, 2012.

Dennis Weyland, Roberto Montemanni, and Luca Maria Gambardella. Heuristics for the probabilistic traveling salesman problem with deadlines based on quasi-parallel Monte Carlo sampling. *Computers & Operations Research*, 40(7):1661–1670, 2013.