Rollout Policies for Dynamic Solutions to the Multi-Vehicle Routing Problem with Stochastic Demand and Duration Limits

Justin C. Goodson Jeffrey W. Ohlmann Barrett W. Thomas

October 2, 2012

Abstract

We develop a family of rollout policies based on fixed routes to obtain dynamic solutions to the *vehicle routing problem with stochastic demand and duration limits* (VRPSDL). In addition to a traditional one-step rollout policy, we leverage the notions of the pre- and post-decision state to distinguish two additional rollout variants. We tailor our rollout policies by developing a dynamic decomposition scheme that achieves high quality solutions to large problem instances with reasonable computational effort. Computational experiments demonstrate that our rollout policies improve upon the performance of a rolling horizon procedure and commonly employed fixed-route policies, with improvement over the latter being more substantial.

Keywords: rollout policy; approximate dynamic programming; stochastic vehicle routing; fixed routes

1 Introduction

Vehicle routing problems (VRPs) with stochastic demand underlie many operational challenges in logistics and supply chain management. For example, in less-than-truckload operations, an estimate of customer demands may be available at the time vehicle routes are planned, but actual demands often differ from the initial estimate (Bertsimas and Simchi-Levi, 1996). Similarly, in vendor-managed beverage distribution systems, actual inventory levels are first observed upon arrival to customers, which may consist of a network of restaurants, grocery stores, and fueling stations (Erera et al., 2009). In such cases, vehicle capacity may be inadequate to fully serve demand,

resulting in a *route failure* requiring the vehicle to return to a central depot to replenish capacity before continuing to service customers.

A common approach to handle uncertainty in customer demands is to restrict attention to *a priori*, or *fixed-route*, policies (see Campbell and Thomas (2008) for a review). A fixed-route policy requires vehicles to visit customers in the order specified by a set of pre-defined routes. Fixed routes are routinely used in industry and create a regularity of service that can be beneficial for both the customers and the drivers – customers may be served at roughly the same time each day they require service, and the drivers become familiar with their routes. Because fixed-route policies offer limited recourse actions in response to observed customer demands, finding a high-quality fixed-route policy is often computationally attractive (e.g., see Yang et al. (2000) or Laporte et al. (2002)).

While fixed-route policies can perform well on average, additional improvements may be obtained by dynamically modifying the routing plan as customer demands become known. Recent advances in communication technologies make such modifications possible, but the body of literature addressing dynamic routing schemes for VRPs with stochastic demand is small. With operating margins between two and four percent in the trucking industry (American Trucking Association, 2009), even small improvements in productivity are significant. While there are always concerns about operationalizing new methods, trucking companies are interested in even one to two percent gains (personal communication, Zahir Balaporia, Director of Intermodal Operations at Schneider National, August 15, 2011). The flat rates and rising costs characteristic of today's trucking industry (Wilson, 2011) further motivate the exploration of dynamic routing techniques as a means to increase productivity.

The overarching contribution of this paper is the development of rollout policies to dynamically manage multi-vehicle fleets when customer demands are stochastic. The literature describes rollout policies to make dynamic decisions in related single-vehicle problems (e.g., Secomandi (2001) and Novoa and Storer (2008)). However, the combinatorial growth in problem size experienced in the multi-vehicle case makes it difficult to extend techniques from the single-vehicle literature to multi-vehicle problems. Our work aims to fill this gap by exploring a family of rollout policies for the *vehicle routing problem with stochastic demand and duration limits* (VRPSDL), a problem that serves as a model for a variety of logistics and distribution problems encountered in practice.

We achieve promising results for the VRPSDL with a traditional, *one-step* rollout policy. However, because the state and action spaces grow combinatorially with problem size, our one-step rollout policy is only practical for problems with 50 or fewer customers. To mitigate this computational challenge, we develop a *post-decision* rollout policy that looks ahead from each post-decision state, rather than from each pre-decision state at the next decision epoch (see Powell

(2007) for a discussion of pre- and post-decision states). We also introduce a *hybrid* rollout policy that reduces the computational burden of post-decision rollout by looking ahead from the current pre-decision state and from select post-decision states.

The dynamic policies we obtain via one-step, post-decision, and hybrid rollout improve on benchmark static fixed-route policies and on dynamic policies achieved by a rolling horizon updating procedure equivalent to applying our rollout algorithm in the pre-decision state. Improvement over static fixed-route policies is more substantial than improvement over rolling horizon policies. Yet, despite the introduction of the post-decision and hybrid methods, for the largest problem instances, our rollout policies are computationally prohibitive to implement. To address this concern, we explore decomposition schemes that reduce computation by implementing rollout policies in single-vehicle problems, then using the resulting policies to manage a multi-vehicle fleet. In particular, we propose a *dynamic* decomposition scheme that builds on the *static* decomposition scheme of Fan et al. (2006). In static decomposition, the set of customers to be served (although not necessarily the order of service) by a particular vehicle is fixed at the beginning of the time horizon. Our dynamic decomposition procedure potentially adjusts the decomposition at each decision epoch. We show that dynamic decomposition is able to achieve high quality solutions for large problem instances with reasonable run times, thereby suggesting that dynamic decomposition, coupled with our rollout algorithms, is a strong candidate for making real-time dynamic routing decisions in large-scale operations.

An important feature of a rollout algorithm is the heuristic used to approximate the reward-to-go from a current state. The simple cyclic heuristic used for single-vehicle problems (Secomandi, 2001; Novoa and Storer, 2008) does not effectively extend to multi-vehicle problems. Therefore, as an additional contribution, we base the look-ahead mechanism in our rollout policies on a local search over a space of fixed-route policies. The development of this fixed-route heuristic includes methods to both estimate and exactly calculate the expected demand served by a fixed-route policy.

Although our work focuses on the VRPSDL, it may be possible to adapt some of our problem-specific ideas to general stochastic dynamic programs. In particular, the concept of a hybrid rollout policy is not necessarily tied to the routing domain. Further, although our dynamic decomposition scheme is tailored specifically for multi-vehicle routing problems, other types of problems may contain structures that permit decomposition amenable to a rollout methodology. In Goodson et al. (2012), we begin to explore these issues and propose a generalized rollout policy framework for applying rollout methods to stochastic dynamic programs.

The remainder of the paper is organized as follows. In §2, we discuss literature addressing dynamic policies for VRPs with stochastic demand. In §3, we formulate the VRPSDL as a Markov decision process (MDP). In §4, we develop a fixed-route heuristic for the VRPSDL. In §5, we

describe our rollout policies and decomposition procedures. We present computational results in §6 and conclude the paper in §7.

2 Related Literature

The formulation underlying much of the literature for dynamic VRPs with stochastic demand is that of Dror et al. (1989) and Dror (1993), who formulate a single-vehicle model as a MDP. The objective of the formulation is to obtain a policy that minimizes expected travel cost subject to a constraint on vehicle capacity. Beyond incorporating a route duration limit, we deviate from this model in two additional ways. First, our model allows for multiple vehicles, which leads to a larger state space and a significantly more complex action space. Second, as we discuss in §3, our objective is to maximize expected demand served, rather than to minimize expected travel costs.

For problem instances of any practical size, obtaining optimal policies for Dror et al. (1989)'s single-vehicle MDP is computationally intractable due to the well known "curse of dimensionality" associated with dynamic programs. Thus, much of the literature focuses on heuristic solution methods. Secomandi (2000) investigates an approximate policy iteration procedure that estimates the cost-to-go via a parametric function. Such an approach, which is typical of more traditional approximate dynamic programming, refines parameter estimates via simulation. Secomandi concludes that a one-step rollout policy performs better than using a parametric function to approximate the cost-to-go. Secomandi (2000, 2001) develops a one-step rollout policy by establishing a single fixed-route policy and repeatedly using it to approximate the cost-to-go for a given action by computing the expected cost of cyclically following this fixed route (skipping customers whose demand has been fully served). We note that Secomandi (2001) implements a one-step rollout policy in a fashion that is computationally equivalent to the post-decision rollout approach articulated in this paper. This equivalency depends on the properties of the heuristic utilized by the rollout algorithm (i.e., sequential consistency) and does not hold in general. Goodson et al. (2012) establish conditions where equivalency between one-step and post-decision rollout exists for general stochastic dynamic programs. We note that this equivalency is not an indicator of overall effectiveness as the heuristic used by the rollout policies in this paper does not maintain sequential consistency, yet achieves high quality computational results. Novoa and Storer (2008) take the approach of Secomandi (2001) and show that using a higher quality initial fixed route leads to improved policies. They also demonstrate that computation times can be reduced, without a loss in solution quality, by using simulation to estimate the expected cost of following a fixed route.

As demonstrated by Fan et al. (2006), one way to address a multi-vehicle problem is to de-

compose it into single-vehicle problems by first partitioning customers into groups (one for each vehicle) and then applying Secomandi (2001) to each group; we refer to this method as *static decomposition*. However, extending the method of Secomandi (2001) in this fashion has limitations. In particular, static decomposition does not permit actions that assign vehicles to customers outside of the initial partitions. In this paper, we explore a *dynamic decomposition* scheme that updates the partitions at each decision epoch, thereby allowing customers to be shared across vehicles.

Secomandi and Margot (2009) suggest partial-reoptimization strategies for a constrained version of Dror et al.'s single-vehicle model. The constraint requires each customer demand to be serviced in full before proceeding to another customer, thereby reducing the size of the state space. The state space is further reduced by heuristically selecting a promising set of states over which an optimal policy is obtained. Applying a similar approach to a multi-vehicle problem is non-trivial. Because the state and action spaces of multi-vehicle problems are exponentially larger than those of single-vehicle problems, it is difficult to identify a promising set of states over which it is computationally tractable to obtain an optimal policy.

Erera and Daganzo (2003) propose a partially-dynamic routing scheme for a multi-vehicle VRP with stochastic demand. The procedure begins by serving customers furthest from the depot via a set of fixed-route policies. Then, there is a one-time replanning of vehicle operations and remaining customer demands are serviced via a new set of fixed-route policies. The rollout policies described in this paper are more dynamic, potentially replanning routes whenever vehicle(s) arrive at customer locations.

Hvattum et al. (2006, 2007) consider a multi-vehicle routing problem with stochastic demand where customer orders are placed over a given time horizon. The problem is modeled as a multi-stage stochastic programming problem with an objective of minimizing travel cost. Within this framework, actions are selected at pre-defined stages through a sample-based heuristic. In our rollout policies, we heuristically select actions whenever new information is learned. Additionally, Hvattum et al. (2006, 2007) require customer demand to be served completely by one vehicle on one visit. If this is not possible, then the customer is skipped. In contrast, we require customer demand to be served to the fullest extent possible given available vehicle capacity. Any remaining customer demand may be served during subsequent visits by the same vehicle, by another vehicle, or not at all. The problem instances of Hvattum et al. (2006, 2007) consider as many as 151 customers and six vehicles. Our work considers problem instances with as many as 100 customers and 19 vehicles.

3 Problem Formulation

The VRPSDL is an extension of the classical vehicle routing problem (see Toth and Vigo (2001) for a review) and is characterized by a fleet of vehicles operating from a central depot and by a set of customers with stochastic demand. Let $G = (\mathcal{N}, \mathcal{E})$ be a complete graph where $\mathcal{N} = \{0, 1, \ldots, N\}$ is a set of $N + 1$ nodes and $\mathcal{E} = \{(n, n') : n, n' \in \mathcal{N}\}$ is the set of edges connecting the nodes. Node 0 represents a depot and nodes $1, \ldots, N$ represent customer locations. Travel times $t(n, n')$ associated with each edge (n, n') in $\mathcal{E}$ are known and assumed deterministic. Let $\mathcal{M} = \{1, \ldots, M\}$ be a set of M identical vehicles initially located at the depot. Let Q denote vehicle capacity and L a route duration limit (e.g., end of a working day) by which time all vehicles must return to the depot. Vehicle routes begin and end at the depot. Customer demands are random variables that follow a known joint probability distribution F with a support restricted to be a subset of $[0, \infty)^N$. Prior to arrival at customer locations, customer demands are known only in distribution. Upon arrival, customer demands are observed and served to the maximum extent, subject to available vehicle capacity. When a vehicle's capacity is exhausted (i.e., a route failure occurs), the vehicle must return to the depot to restore capacity up to Q. Demand that remains unserved after an initial vehicle visit may possibly be satisfied on subsequent visits by the same vehicle, or by another.

The objective of the VRPSDL is to obtain a policy that maximizes expected demand served. Our choice of objective function is motivated by van de Klundert and Wormer (2010), who emphasize the need to focus on customer service levels. Further, the VRPSDL objective of maximizing expected demand served aligns with the goals of many humanitarian and disaster relief logistics problems (Campbell et al., 2008), such as the distribution of vaccines or emergency medical supplies. Alternative objective functions might also consider the more traditional goal of cost-minimization by placing appropriate weights on each objective.

We formulate the VRPSDL as a MDP. We present a summary of the MDP model in this section and refer the reader to Appendix A for a detailed formulation. The state of the system is the tuple (l, t, q, d, x). For each vehicle, vectors l, t, and q store vehicle destinations, arrival times at vehicle destinations, and remaining vehicle capacities, respectively. The state of vehicle m in $\mathcal{M}$ is denoted (l_m, t_m, q_m). For each customer, vectors d and x store unserved customer demand and the observed demand, respectively. The state of demand at customer n in $\mathcal{N}$ is denoted (d_n, x_n). An action is an assignment of the vehicles in $\mathcal{M}$ to locations in $\mathcal{N}$. We place the following restrictions on the available actions. Vehicles en route may not be diverted. If a vehicle's capacity will be depleted by serving customer demand at its current location, then the vehicle must return to the depot to replenish. We prohibit actions assigning more than one vehicle to a customer, but allow multiple

vehicles to return to the depot. Actions forcing a vehicle to return to the depot at a time greater than L are prohibited. Vehicles are not permitted to wait at locations, except at the depot when it is not feasible to collect demand from the remaining set of customers. We denote the pre-decision state at decision epoch k by s_k, an action in action space $\mathcal{A}(s_k)$ by a, and the post-decision state corresponding to action a by s_k^a (see Powell (2007) for a discussion of pre- and post-decision states).

The system dynamics proceed as follows. A decision epoch is triggered by the arrival of one or more vehicles at customer locations or at the depot (multiple vehicles may arrive simultaneously). Upon arrival to customer locations, actual demand is observed. In addition, vehicle capacities are replenished for vehicles arriving at the depot. These events are captured in the transition from post-decision state s_{k-1}^a to pre-decision state s_k. For vehicles at customer locations or at the depot (i.e., vehicles not en route), an action is selected indicating each respective location these vehicles will travel to next. Vehicles currently at customer locations then serve demand to the fullest extent given available vehicle capacity (total demand served in period k is recorded as reward $R_k(s_k, a)$) and the selected action, specifying the next destination for each vehicle, is executed. These events are captured in the deterministic transition from pre-decision state s_k to post-decision state s_k^a.

The example in Figure 1 and Table 1 illustrates states and actions. The example consists of two vehicles and four customers with Manhattan distances defined by a unit grid. Pre-decision state s_k is noted at the top of Figure 1. In state s_k, vehicle 1 is at customer 1 and vehicle 2 is at customer 4. These vehicles arrived simultaneously at their respective locations at time 2, both with full capacity of 50 units. Demands at customers 1 and 4 are observed to be 20 and 15 units, respectively, none of which has been served. Demands at customers 2 and 3 are not yet observed (indicated by "?"); the table within Figure 1 displays the possible demands at these customers. The first column of Table 1 displays the feasible action set $\mathcal{A}(s_k)$. Each of the seven actions is a two-dimensional vector indicating a feasible assignment of the two vehicles to locations. For example, action 1 assigns both vehicles to the depot to replenish capacity. The second column of Table 1 displays the post-decision states that result from selecting the actions in the first column. Because the reward collected depends only on the demands at current vehicle locations and available vehicle capacities, the reward $R_k(s_k, a)$ is $20 + 15 = 35$ units of demand for any action a listed in the first column. The post-decision states in the second column reflect that demand at customers 1 and 4 has been fully served by indicating zero unserved demand at these customers. The third column of Table 1 displays all possible pre-decision states at decision epoch $k + 1$. Multiple pre-decision states are listed when customer demand(s) have not yet been observed at the location(s) to which vehicle(s) arrive next. We discuss the remainder of Table 1 in the sections that follow.

Let Π be the set of all Markovian deterministic policies for the VRPSDL. A policy π in Π is a

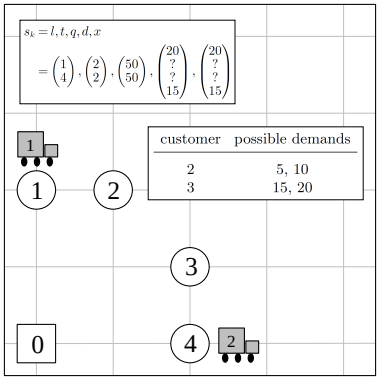

Figure 1: Example of a Pre-Decision State in a Two-Vehicle, Four-Customer VRPSDL

sequence of decision rules: $\pi = (\delta_0^\pi, \delta_1^\pi, \ldots, \delta_K^\pi)$, where each decision rule $\delta_k^\pi(s_k) : s_k \mapsto \mathcal{A}(s_k)$ is a function that specifies the action choice when the process occupies state s_k and follows policy π. We seek a policy π in Π that maximizes the total expected reward, conditional on initial state s_0: $\mathbb{E}[\sum_{k=0}^{K} R_k(s_k, \delta^\pi(s_k))|s_0]$. Denoting by $V(s_k)$ the expected reward-to-go from state s_k in epoch k through final decision epoch K, an optimal policy can be obtained by solving the optimality equation $V(s_k) = \max_{a \in \mathcal{A}(s_k)}\{R_k(s_k, a) + \mathbb{E}[V(s_{k+1})|s_k, a]\}$ for each epoch k and state s_k in state space $\mathcal{S}$, where $V(s_K) = 0$ for each absorbing state s_K.

4 A Fixed-Route Heuristic

In this section, we present a fixed-route heuristic for the VRPSDL. We use this heuristic in §5 to develop rollout policies for the VRPSDL. In §4.1, we define a fixed-route policy class. In §4.2, we discuss a local search procedure for fixed-route policies.

4.1 Fixed-Route Policies

As we discuss in §1, a fixed route specifies a static ordering of customers for a driver to visit. We denote by $\mathcal{V}(s)$ the set of all possible fixed routes when the process occupies a pre- or post-decision state s. We denote by v^m a fixed route for vehicle m in vehicle set $\mathcal{M}$, where $v^m = (v_1^m, v_2^m, \ldots, v_{B^m}^m)$ is a sequence of B^m locations such that v_1^m is the destination of vehicle m in state s and the remaining locations are customers in $\mathcal{N} \setminus \{0\}$; if vehicle m is at a location in state s (versus en route to a destination), then v_1^m is this location. A fixed-route policy is characterized by $v = (v^m)_{m \in \mathcal{M}}$, a set containing a fixed route for each vehicle m in $\mathcal{M}$. In v, each customer with pending or unknown demand appears exactly once on exactly one route. As an example,

Table 1: Example of System Dynamics in a Two-Vehicle, Four-Customer VRPSDL

Action a	**Post-Decision State** s_k^a	**Pre-Decision State(s)** s_{k+1}	**Fixed Routes** v **Yielding Action** a
1: $\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}4\\4\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}4\\4\end{pmatrix}, \begin{pmatrix}50\\50\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	none
2: $\begin{pmatrix}0\\2\end{pmatrix}^{\dagger\ddagger}$	$\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}4\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}4\\5\end{pmatrix}, \begin{pmatrix}50\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$v^1 = (1)$ $v^2 = (4, 2, 3)$
3: $\begin{pmatrix}0\\3\end{pmatrix}^{\dagger\ddagger}$	$\begin{pmatrix}0\\3\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}0\\3\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\15\\0\end{pmatrix}, \begin{pmatrix}20\\?\\15\\15\end{pmatrix}$ $\begin{pmatrix}0\\3\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\20\\0\end{pmatrix}, \begin{pmatrix}20\\?\\20\\15\end{pmatrix}$	$v^1 = (1)$ $v^2 = (4, 3, 2)$
4: $\begin{pmatrix}2\\0\end{pmatrix}^{\dagger\ddagger}$	$\begin{pmatrix}2\\0\end{pmatrix}, \begin{pmatrix}3\\4\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}2\\0\end{pmatrix}, \begin{pmatrix}3\\4\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\5\\?\\0\end{pmatrix}, \begin{pmatrix}20\\5\\?\\15\end{pmatrix}$ $\begin{pmatrix}2\\0\end{pmatrix}, \begin{pmatrix}3\\4\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\10\\?\\0\end{pmatrix}, \begin{pmatrix}20\\10\\?\\15\end{pmatrix}$	$v^1 = (1, 2, 3)$ $v^2 = (4)$
5: $\begin{pmatrix}2\\3\end{pmatrix}^{\dagger}$	$\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\5\\15\\0\end{pmatrix}, \begin{pmatrix}20\\5\\15\\15\end{pmatrix}$ $\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\5\\20\\0\end{pmatrix}, \begin{pmatrix}20\\5\\20\\15\end{pmatrix}$ $\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\10\\15\\0\end{pmatrix}, \begin{pmatrix}20\\10\\15\\15\end{pmatrix}$ $\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\10\\20\\0\end{pmatrix}, \begin{pmatrix}20\\10\\20\\15\end{pmatrix}$	$v^1 = (1, 2)$ $v^2 = (4, 3)$
6: $\begin{pmatrix}3\\0\end{pmatrix}^{\dagger\ddagger}$	$\begin{pmatrix}3\\0\end{pmatrix}, \begin{pmatrix}5\\4\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}3\\0\end{pmatrix}, \begin{pmatrix}5\\4\end{pmatrix}, \begin{pmatrix}30\\50\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$v^1 = (1, 3, 2)$ $v^2 = (4)$
7: $\begin{pmatrix}3\\2\end{pmatrix}^{\dagger}$	$\begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\?\\?\\0\end{pmatrix}, \begin{pmatrix}20\\?\\?\\15\end{pmatrix}$	$\begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\5\\15\\0\end{pmatrix}, \begin{pmatrix}20\\5\\15\\15\end{pmatrix}$ $\begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\5\\20\\0\end{pmatrix}, \begin{pmatrix}20\\5\\20\\15\end{pmatrix}$ $\begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\10\\15\\0\end{pmatrix}, \begin{pmatrix}20\\10\\15\\15\end{pmatrix}$ $\begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}30\\35\end{pmatrix}, \begin{pmatrix}0\\10\\20\\0\end{pmatrix}, \begin{pmatrix}20\\10\\20\\15\end{pmatrix}$	$v^1 = (1, 3)$ $v^2 = (4, 2)$

consider again Figure 1. A fixed route for vehicle 1 must begin at customer 1, vehicle 1's current location. Similarly, a fixed route for vehicle 2 must begin at customer 4. One possible set of fixed routes is obtained by assigning customers 3 and 2 to vehicle 1, in that order, resulting in $v = (v^1, v^2) = ((v_1^1, v_2^1, v_3^1), (v_1^2)) = ((1, 3, 2), (4))$. The set $\mathcal{V}(s_k)$ consists of all possible assignments of customers 2 and 3 to routes for vehicles 1 and 2.

The decision rules we associate with a fixed-route policy are dictated by the set of fixed routes. Vehicles are required to visit customers in the order specified by the fixed routes. After a vehicle m serves demand at a customer v_b^m, vehicle m continues directly to the next customer on its fixed route, v_{b+1}^m, or to the depot if v_b^m is the final customer on the fixed route. If continuing on to v_{b+1}^m will result in a violation of the route duration limit, a vehicle is required to return to the depot. A return trip to the depot for the purpose of capacity replenishment is required in the event of a route failure and is not permitted otherwise. Although it is possible to consider fixed-route policies that permit replenishment prior to route failures (e.g., in the spirit of Bertsimas et al. (1995)), preemptive capacity replenishment policies for the VRPSDL are computationally challenging to evaluate (Goodson, 2010). In §5, we discuss how we consider preemptive capacity replenishments via our rollout policies.

To precisely describe a fixed-route decision rule, denote by $\pi(v) = (\pi(v^m))_{m \in \mathcal{M}}$ the fixed-route policy associated with the set of fixed routes v, which consists of a separate policy for each vehicle m in $\mathcal{M}$. At decision epoch k, suppose the process occupies state $s_k = (l, t, q, d, x)$. Denote by T_k the time at which decision epoch k occurs and by $\mathcal{M}'$ the set of vehicles that reach their destinations at time T_k. Let v_b^m be the first customer on the fixed route for vehicle m that will have pending or unknown demand after demand at the current location is served to the fullest extent possible. Customer v_b^m may be the current location if available capacity is insufficient to meet demand at the vehicle's current location. If demand at all customers on the fixed route will be served during the current period, then let v_b^m be the depot. Then, the fixed-route decision rule for vehicle m is

$$\delta_k^{\pi(v^m)}(s_k) = \begin{cases} l_m, & m \notin \mathcal{M}', \\ 0, & m \in \mathcal{M}' \text{ and } q_m \leq d_{l_m}, \\ 0, & m \in \mathcal{M}' \text{ and } T_k + t(l_m, v_b^m) + t(v_b^m, 0) > L, \\ v_b^m, & \text{otherwise.} \end{cases} \tag{1}$$

The first case in equation (1) requires a vehicle en route to continue to its destination. The second case assigns a vehicle to the depot if capacity will be depleted after serving demand at its current location. The third case assigns a vehicle to the depot if traveling to customer v_b^m will result in a

violation of the route duration limit. The fourth case assigns a vehicle to location v_b^m. The decision rule for policy $\pi(v)$ consists of the decision rules for each fixed route composing v: $\delta_k^{\pi(v)} = (\delta_k^{\pi(v^m)})_{m \in \mathcal{M}}$.

We denote by $V^{\pi(v)}(s)$ the expected value of following policy $\pi(v)$ onward from a pre- or post-decision state s, where policy $\pi(v)$ is obtained by applying the fixed-route heuristic of §4.2. We denote by $\widehat{V}^{\pi(v)}(s)$ an unbiased and consistent estimator of $V^{\pi(v)}(s)$. In Appendix B, we provide a pseudo-polynomial-time procedure to calculate $V^{\pi(v)}(s)$ and a Monte Carlo simulation procedure to calculate $\widehat{V}^{\pi(v)}(s)$. The local search procedure in §4.2 employs both evaluation methods.

4.2 Fixed-Route Local Search

From a given pre- or post-decision state s, we describe a local search procedure to search $\mathcal{V}(s)$, the space of all fixed routes when the process occupies state s. The local search utilizes a *relocation* neighborhood (Kindervater and Savelsbergh, 2003) and a *first-improving* search criterion.

The relocation neighborhood of a set of fixed routes v consists of all sets of fixed routes that can be obtained by relocating a customer v_i^m after another customer $v_j^{m'}$ such that $i \neq 1$. The condition $i \neq 1$ ensures that vehicles' current destinations, given by state s, are fixed for all solutions in the neighborhood. For example, consider again the pre-decision state s_k depicted in Figure 1 where the current location of vehicle 1 is customer 1 and the current location of vehicle 2 is customer 4. Suppose the current fixed-route policy $\pi(v)$ is characterized by the set of fixed routes $v = (v^1, v^2) = ((1, 2, 3), (4))$. Then, the relocation neighborhood of v consists of three sets of fixed routes: $((1, 3, 2), (4))$, $((1, 3), (4, 2))$, and $((1, 2), (4, 3))$.

Each iteration of the local search proceeds as follows. Given a current set of fixed routes v, a set of fixed routes $\bar{v}$ is randomly selected from the relocation neighborhood of v. If the estimated value of policy $\pi(\bar{v})$ is greater than the estimated value of policy $\pi(v)$ (i.e., $\widehat{V}^{\pi(\bar{v})}(s) > \widehat{V}^{\pi(v)}(s)$), then the current set of fixed routes is updated to be $\bar{v}$ and the process repeats. Otherwise, the search continues by randomly selecting another set of fixed routes from the relocation neighborhood of v. The procedure terminates and returns fixed-route policy $\pi(v)$ if there does not exist a set of fixed routes $\bar{v}$ in the relocation neighborhood of v such that $\widehat{V}^{\pi(\bar{v})}(s) > \widehat{V}^{\pi(v)}(s)$. After the local search terminates, we calculate $V^{\pi(v)}(s)$, the exact value of policy $\pi(v)$.

We note that our first-improving local search strikes a balance between computational effort and solution quality. In preliminary experiments, we employed simple greedy construction procedures as well as more sophisticated simulated-annealing based methods. The former executed very quickly, but delivered poor performance in the rollout policies of §5. The latter performed quite well, but required significant computation time, thereby making it difficult to carry out the full

range of computational experiments we present in §6. The first-improving local search we present in this section yields acceptable fixed-route policies with reasonable computational effort.

5 Rollout Policies for the VRPSDL

We use the fixed-route heuristic of §4, in conjunction with rollout algorithms, to approximate the reward-to-go at each decision epoch. Rollout algorithms are heuristic versions of policy iteration for dynamic programs and employ the concept of forward dynamic programming. From a current state, the reward-to-go is approximated by heuristic policies, which are then used to guide action selection in the current state. For an in-depth discussion of rollout applied to deterministic dynamic programs, we refer the reader to Bertsekas et al. (1997) and Bertsekas (2000). Goodson et al. (2012) describe rollout policies for general stochastic dynamic programs. A general introduction to approximate dynamic programming is provided by Powell (2007).

In §5.1 and §5.2, we present *one-step* and *post-decision* rollout policies for the VRPSDL. In §5.3, we introduce a hybrid rollout policy for the VRPSDL. In §5.4, we discuss a *fortified* rollout procedure that provides a performance guarantee for our rollout policies. In §5.5, we discuss *static* and *dynamic* decomposition methods for the VRPSDL and describe how to implement rollout policies for both schemes.

5.1 One-Step Rollout

In the literature, rollout policies are typically implemented as an approximate dynamic programming procedure that looks ahead one step, using a heuristic to approximate the reward-to-go (e.g., see Secomandi (2003), Bertsimas and Popescu (2003), Bertsimas and Demir (2002), or Guerriero and Mancini (2003)). From a current state s_k and for a given action a, our one-step rollout policy transitions to all possible states s_{k+1} that may be observed at the next decision epoch. We denote these states by the set $\mathcal{S}(s_k, a) = \{s_{k+1} : \mathbb{P}\{s_{k+1}|s_k, a\} > 0\}$. Augmenting our notation in §4, we denote by $V^{\pi(v,s')}(s)$ the expected value of following a fixed-route policy $\pi(v)$ onward from a pre- or post-decision state s, where policy $\pi(v)$ is obtained by applying the fixed-route heuristic of §4.2 in a pre- or post-decision state s'. From each pre-decision state s_{k+1} in $\mathcal{S}(s_k, a)$, we execute the fixed-route heuristic to obtain a fixed-route policy $\pi(v)$ with value $V^{\pi(v,s_{k+1})}(s_{k+1})$. In a one-step rollout algorithm, the estimate of the reward-to-go when selecting action a in state s_k is given by the expected value of the fixed-route policies obtained in all possible states s_{k+1}: $\mathbb{E}[V^{\pi(v,s_{k+1})}(s_{k+1})|s_k, a] = \sum_{s_{k+1}\in\mathcal{S}(s_k,a)} V^{\pi(v,s_{k+1})}(s_{k+1}) \times \mathbb{P}\{s_{k+1}|s_k, a\}$. One-step rollout selects an action a in feasible action set $\mathcal{A}(s_k)$ that maximizes the value of

$R_k(s_k, a) + \mathbb{E}[V^{\pi(v,s_{k+1})}(s_{k+1})|s_k, a]$.

For each feasible action a, one-step rollout executes the fixed-route heuristic $|\mathcal{S}(s_k, a)|$ times, resulting in a total of $\sum_{a\in\mathcal{A}(s_k)} |\mathcal{S}(s_k, a)|$ executions of the fixed-route heuristic. For example, in Table 1, the estimated reward-to-go for selecting action 5 is obtained by calculating the expected value of the four fixed-route policies that result from executing the fixed-route heuristic from each of the four states listed in the third column. Over all actions in $\mathcal{A}(s_k)$, one-step rollout executes the fixed-route heuristic 15 times, once from each state in the third column.

5.2 Post-Decision Rollout

Our computational work in §6 shows that one-step rollout policies yield good results. However, the computation required to look ahead one full step at each decision epoch is possible only for problem instances with 50 or fewer customers. We seek to reduce this computational burden by considering a *post-decision* rollout policy (see Goodson et al. (2012) for a more in-depth discussion of post-decision rollout). From a current state s_k and for a given action a, our post-decision rollout policy transitions to the post-decision state s_k^a. From state s_k^a, we execute the fixed-route heuristic resulting in a fixed-route policy $\pi(v)$, which is then used to estimate the reward-to-go. Post-decision rollout selects an action a in $\mathcal{A}(s_k)$ that maximizes the value of $R_k(s_k, a) + \mathbb{E}[V^{\pi(v,s_k^a)}(s_{k+1})|s_k, a]$.

Post-decision rollout fixes vehicle destinations, capacities, and arrival times at the next decision epoch, but does not observe future demand as in one-step rollout. Thus, in contrast to one-step rollout, which executes the fixed-route heuristic $\sum_{a\in\mathcal{A}(s_k)} |\mathcal{S}(s_k, a)|$ times, post-decision rollout executes the fixed-route heuristic only $|\mathcal{A}(s_k)|$ times, or once from each post-decision state. This contrast is highlighted in Table 1. Whereas one-step rollout executes the fixed-route heuristic from each of the 15 states in the third column, post-decision rollout executes the fixed-route heuristic only seven times, once from each post-decision state listed in the second column. Noting that the worst-case number of feasible actions is $O(M^N)$, the worst-case number of fixed-route executions per epoch is $O(M^N|\mathcal{S}|)$ in one-step rollout and $O(M^N)$ in post-decision rollout. Although the worst-case complexity of both procedures is exponential, the complexity of one-step rollout depends on the size of the state space, which can be quite large (see the detailed problem formulation in Appendix A).

Our computational experiments suggest that post-decision rollout is an attractive alternative to one-step rollout. Although post-decision rollout reduces computation relative to one-step rollout, the computational requirement of post-decision rollout can still be demanding, particularly when decision epochs occur as a result of the simultaneous arrival of vehicles to locations. In these

situations, the number of feasible actions grows combinatorially with the number of vehicles arriving simultaneously and with the number of customers. For example, the number of feasible actions at a decision epoch may range from several hundred to several million. In the latter case, it is not difficult to execute a "simple" heuristic with short run times (e.g., a greedy construction procedure) from the post-decision states corresponding to each feasible action. However, as we discuss in §4.2, our experience with the VRPSDL indicates that overly-simplified heuristics perform poorly because they frequently return low-quality fixed-route policies. Although our local search procedure for fixed-route policies yields higher-quality policies, it requires more time to execute. Consequently, executing this heuristic once for each action in a large action space is computationally prohibitive.

5.3 Hybrid Rollout

We introduce hybrid rollout to circumvent the computational issues encountered in post-decision rollout. Hybrid rollout limits the number of post-decision states from which the fixed-route heuristic is executed. We refer to our method as *hybrid* because it executes the fixed-route heuristic from the pre-decision state and from select post-decision states. The basic premise of hybrid rollout is that a single execution of the fixed-route heuristic from the current pre-decision state s_k (in the spirit of a rolling horizon procedure) can be used to implicitly evaluate a subset of the action space, $\bar{\mathcal{A}}(s_k) \subseteq \mathcal{A}(s_k)$. Then, specific actions are identified and evaluated via the fixed-route heuristic from their respective post-decision states. This is in contrast to one-step and post-decision rollout, which explicitly evaluate each feasible action by executing the fixed-route heuristic from the resulting pre- or post-decision states, respectively.

Algorithm 1 details the logic used by hybrid rollout to select an action at decision epoch k. Line 1 defines the set $\bar{\mathcal{A}}(s_k) = \{\delta_k^{\pi(v)}(s_k) : v \in \mathcal{V}(s_k)\}$ as the set of actions obtainable from the application of the fixed-route decision rule in equation (1) to each set of fixed routes v in $\mathcal{V}(s_k)$, the space of all fixed routes in the current pre-decision state. To identify the "best" action in $\bar{\mathcal{A}}(s_k)$, line 2 executes the fixed-route heuristic from the current pre-decision state s_k. Line 3 uses the resulting policy, $\pi(\bar{v})$, to extract action $\bar{a}_k = \delta_k^{\pi(\bar{v})}(s_k)$ via the fixed-route decision rule in equation (1). Action $\bar{a}_k$ is one of a set of select actions to be evaluated from the post-decision state in subsequent steps of Algorithm 1.

Table 1 illustrates the construction of $\bar{\mathcal{A}}(s_k)$. The fourth column of Table 1 lists the fixed routes composing $\mathcal{V}(s_k)$ (note that in larger problem instances, there may be many fixed-route policies that yield the same action). The actions that result from applying the decision rule in equation (1) to each fixed route are displayed in the first column. For example, action 5 is obtained by applying the

Algorithm 1 Hybrid Rollout

$\bar{\mathcal{A}}(s_k) \leftarrow \{\delta_k^{\pi(v)}(s_k) : v \in \mathcal{V}(s_k)\}$
Execute fixed-route heuristic from pre-decision state s_k yielding fixed-route policy $\pi(\bar{v})$
$\bar{a}_k \leftarrow \delta_k^{\pi(\bar{v})}(s_k)$
$\hat{\mathcal{A}}(s_k) \leftarrow \{a \in \bar{\mathcal{A}}(s_k) : \exists\ m \in \mathcal{M}'$ such that $a_m = 0$ and $q_m > d_{l_m}\}$
$\lambda \leftarrow -\infty$
for $a \in \{\mathcal{A}(s_k) \setminus \bar{\mathcal{A}}(s_k)\} \cup \hat{\mathcal{A}}(s_k) \cup \bar{a}_k$ **do**
    Execute fixed-route heuristic from post-decision state s_k^a yielding fixed-route policy $\pi(v)$
    $\eta \leftarrow R_k(s_k, a) + \mathbb{E}[V^{\pi(v, s_k^a)}(s_{k+1})|s_k, a]$
    if $\eta > \lambda$ **then**
        $\lambda \leftarrow \eta$
        $a^\star \leftarrow a$
Select action $a^\star$

decision rule in equation (1) to each fixed route in the set of fixed routes $((1,2),(4,3))$, resulting in an assignment of vehicle 1 to customer 2 and vehicle 2 to customer 3. Actions 2-7 in Table 1, superscripted by †, compose the set $\bar{\mathcal{A}}(s_k)$.

While the single execution of the fixed-route heuristic from the pre-decision state s_k may be effective at implicitly evaluating the large number of actions in $\bar{\mathcal{A}}(s_k)$, it may suffer from two shortcomings. First, it may not consider some actions, i.e., $\{\mathcal{A}(s_k) \setminus \bar{\mathcal{A}}(s_k)\} \neq \emptyset$ if there are actions that cannot be represented by a set of fixed routes in the pre-decision state. In Table 1, action 1, which preemptively replenishes the capacity of both vehicles, composes $\{\mathcal{A}(s_k) \setminus \bar{\mathcal{A}}(s_k)\}$.

Second, the single execution of the fixed-route heuristic from the pre-decision state s_k may inaccurately evaluate a subset of actions $\hat{\mathcal{A}}(s_k) \subseteq \bar{\mathcal{A}}(s_k)$. Line 4 of Algorithm 1 defines $\hat{\mathcal{A}}(s_k) = \{a \in \bar{\mathcal{A}}(s_k) : \exists\ m \in \mathcal{M}'$ such that $a_m = 0$ and $q_m > d_{l_m}\}$ as the set of preemptive capacity replenishment actions belonging to $\bar{\mathcal{A}}(s_k)$; as defined in §4.1, $\mathcal{M}'$ is the set of vehicles that have arrived at a customer location at decision epoch k. In Table 1, $\hat{\mathcal{A}}(s_k)$ consists of actions 2, 3, 4, and 6, each superscripted by ‡. The evaluation of these preemptive capacity replenishment actions by the fixed-route heuristic from the pre-decision state is often inaccurate. For example, consider action 2 in Table 1, which assigns vehicle 2 to customer 2 and preemptively replenishes vehicle 1's capacity by assigning vehicle 1 to the depot. The fourth column of Table 1 indicates that action 2 corresponds to the set of fixed routes $((1),(4,2,3))$. Thus, a fixed-route heuristic evaluating action 2 from the pre-decision state does so by assuming that vehicle 1 will serve demand at its current location (customer 1), then return permanently to the depot, and that all other customers will be visited by vehicle 2. This is potentially problematic because assuming that vehicle 1 is unable to

serve additional demand is often inaccurate, particularly near the beginning of the time horizon.

Using local variables λ and η, lines 5-11 execute post-decision rollout over the actions excluded or potentially misevaluated by the previous application of the fixed-route heuristic from the pre-decision state. In addition, $\bar{a}_k$ is also evaluated from the post-decision state to facilitate a fair comparison. Line 12 concludes Algorithm 1 by identify the action to select at decision epoch k.

To summarize the example in Table 1, executing the fixed-route heuristic from pre-decision state s_k returns action $\bar{a}_k = 2$, 3, 4, 5, 6, or 7. Action 1 composes the set $\mathcal{A}(s_k) \setminus \bar{\mathcal{A}}(s_k)$ and preemptive capacity replenishment actions 2, 3, 4, and 6 belong to $\hat{\mathcal{A}}(s_k)$. In the worst case, if $\bar{a}_k$ is action 5 or 7, then the fixed-route heuristic is executed from the pre-decision state and from six post-decision states, a total of seven executions. In this small example, the number of fixed-route heuristic executions may be the same for post-decision and hybrid rollout, i.e., the worst-case complexities are equivalent. However, depending on the sizes of the sets $\mathcal{A}(s_k)$, $\bar{\mathcal{A}}(s_k)$, and $\hat{\mathcal{A}}(s_k)$, a hybrid rollout policy may execute the fixed-route heuristic many fewer times than the number of executions required by a post-decision rollout policy. Our experience with the VRPSDL indicates that this difference grows combinatorially with the number of customers and the number of vehicles.

5.4 Fortified Rollout

Our one-step, post-decision, and hybrid rollout procedures for the VRPSDL are each implemented in a fashion that guarantees their performance is at least as good as the performance of the benchmark static fixed routes described in §6.2. We accomplish this by using the benchmark policy to select an action at the first decision epoch, then at each subsequent decision epoch we compare the value of the in-hand fixed-route policy against the value of the policy returned by the fixed-route heuristic and use the policy with the larger value to select an action. In the rollout literature, the performance guarantee we achieve is known as the *rollout improvement property* and the procedure is commonly referred to as a *fortified rollout algorithm*. Bertsekas et al. (1997) introduce the rollout improvement property and the concept of a fortified algorithm for deterministic dynamic programs. The results we apply in this work are due to Goodson et al. (2012), which extends the work of Bertsekas et al. (1997) to general stochastic dynamic programs. We apply the same techniques in the decomposition schemes of §5.5.

5.5 Decomposition-Based Rollout Policies

The computational experiments of §6 find that, relative to one-step and post-decision rollout, hybrid rollout increases the problem sizes we are able to consider up to 75-customer problems. Beyond this, hybrid rollout faces the same obstacle that prevents post-decision rollout from scaling to larger problem instances: the number of post-decision states from which the fixed-route heuristic is executed becomes prohibitive. The *static* and *dynamic decomposition* schemes we explore in this section seek to reduce the required computation by applying rollout policies to single-vehicle problems and using the resulting policies to select actions for a multi-vehicle fleet. The effect of both decomposition schemes is to restrict the action space over which one-step, post-decision, and hybrid rollout are applied. The restriction is achieved by disallowing actions that assign vehicles to customers outside of the single-vehicle problems. The decomposition schemes lead to computational tractability for larger VRPSDL problem instances.

As discussed in §2, the *static decomposition* method of Fan et al. (2006) is achieved by partitioning customers into permanent groups at the beginning of the time horizon, i.e., at initial state s_0. At a given decision epoch, the feasible action space for a vehicle is restricted by the customers in its partition. Referring to the two-vehicle example problem in Figure 1, one possible partitioning assigns customers 1 and 2 to vehicle 1 and customers 3 and 4 to vehicle 2. Then, from the state s_k given in Figure 1, the feasible action set for vehicle 1 consists of an assignment to customer 2 or to the depot. Similarly, the feasible action set for vehicle 2 consists of an assignment to customer 3 or to the depot. In their work, Fan et al. (2006) partition customers geographically in a radial fashion. Our computational experience suggests that performance can be improved by partitioning customers via a high-quality set of fixed routes. We obtain this set of fixed routes via a simulated annealing heuristic executed from initial state s_0 (see Appendix C for details).

The static decomposition method of Fan et al. (2006) leads to two computational advantages. The first advantage is a reduction in the number of feasible actions – all actions assigning a vehicle to a customer outside of its partition are ignored. This in turn reduces the number of times the fixed-route heuristic must be executed in one-step, post-decision, and hybrid rollout policies. The worst-case complexities of the rollout algorithms reflect this reduction. In the worst-case, the number of fixed-route heuristic executions per epoch is $O(MN|\mathcal{S}|)$ for one-step rollout and $O(MN)$ for post-decision and hybrid rollout. The second advantage is a reduction in the size of the neighborhood used in the fixed-route local search. Because attention is restricted to single-vehicle routes within a particular partition, the number of fixed routes in the neighborhood of a current solution is much smaller than in the multi-vehicle case.

Although the static decomposition method of Fan et al. (2006) is computationally attractive,

our computational experience suggests that rollout policies achieve better results without decomposition. While disallowing customers to be shared across vehicles significantly decreases the size of the action space, it also prohibits the selection of actions that lead to increases in expected demand served.

In an effort to overcome this limitation, we propose a *dynamic decomposition* scheme that aims to retain the computational advantage of static decomposition while improving its performance. In dynamic decomposition, customers are re-partitioned at each decision epoch. At decision epoch k when the process occupies state s_k, we re-partition customers by executing the fixed-route heuristic of §4.2 from the current pre-decision state s_k. The fixed routes returned by the heuristic define the partitions for decision epoch k. Due to its time-consuming nature, we employ the fixed-route simulated annealing heuristic (see Appendix C) only to determine the initial partitions. As with static decomposition, one-step, post-decision, and hybrid rollout can be applied to each partition to obtain a decision rule for the corresponding vehicle. At the next decision epoch, we again execute the fixed-route heuristic to obtain a potentially different partitioning.

Like static decomposition, dynamic decomposition limits the number of feasible actions by only allowing a vehicle to be routed to customers in its partition. However, because dynamic decomposition permits changes in the customer partitions, customers assigned to a certain partition at one decision epoch may not necessarily by assigned to the same partition at a future decision epoch. Our computational experiments indicate that dynamically adjusting the decomposition in this fashion yields much better results than static decomposition.

6 Computational Results

In this section, we report on computational experiments that implement the VRPSDL rollout policies described in §5. In §6.1, we detail the generation of problem instances used in our experiments. In §6.2, we describe upper bounds and benchmarks for our rollout policies. A discussion of our computational results is provided in §6.3.

We implement our procedures in C++ and execute all computational experiments on 2.8GHz Intel Xeon processors with 12-48GB of RAM and the CentOS 5.3 operating system (we do not utilize parallel processing). The total computing time required to carry out the experiments is 6.2 CPU years, which breaks down as follows: 76 percent of the computing time is due to experiments with no decomposition, 2 percent is due to experiments with static decomposition, and 22 percent is due to experiments with dynamic decomposition.

Table 2: Problem Parameters

Problem	Vehicles	Duration Limits (short, medium, long)	Capacities (small, medium, large)
R101 (25)	4	(85.575, 142.625, 199.675)	(25, 50, 75)
C101 (25)	5	(67.875, 113.125, 158.375)	(25, 50, 75)
R101 (50)	8	(89.475, 149.125, 208.775)	(25, 50, 75)
C101 (50)	9	(67.875, 113.125, 158.375)	(25, 50, 75)
R101 (75)	11	(103.05, 171.75, 240.45)	(25, 50, 75)
C101 (75)	14	(99.45, 165.75, 232.05)	(25, 50, 75)
R101 (100)	15	(94.875, 158.125, 221.375)	(25, 50, 75)
C101 (100)	19	(95.325, 158.875, 222.425)	(25, 50, 75)

6.1 Problem Instances

To facilitate our experiments, we modify eight problems derived from the instances of Solomon (1987), ignoring the time windows, with the aim of considering a broad range of problem types. The problems include R101 (randomly dispersed customers) and C101 (clustered customers), each with 25, 50, 75, and 100 customers. We vary vehicle capacity, impose route duration limits, and vary customer demand variability to yield a total of 216 problem instances. We set problem parameters as follows. For each of the eight Solomon instances, we consider vehicle capacities of 25, 50, and 75 units, hereafter referred to as *small*, *medium*, and *large*. To determine fleet size and route duration limits, we first solve each of the eight Solomon instances as a classical VRP (i.e., time windows are ignored) with a vehicle capacity of 100. We employ the heuristic of Goodson et al. (2012) for this purpose. The fleet size M for each problem instance is set to the number of routes in the VRP solution. We then multiply the length of the longest route in each solution by 0.75, 1.25, and 1.75. The resulting route duration limits are hereafter referred to as *short*, *medium*, and *long*. Fleet size, route duration limits, and capacities are displayed in Table 2.

We assume customer demands are independent random variables, and we construct discrete, symmetric probability distributions for customer demand with *low*, *moderate*, and *high* variability. The expected demand for each customer is set to the deterministic demand given in the original Solomon instances; denote this quantity by $\bar{x}$. To construct a high variability probability distribution for a given customer, we assign uniform probabilities of 0.20 to $\{0, \bar{x}/2, \bar{x}, 3\bar{x}/2, 2\bar{x}\}$. To construct a moderate variability probability distribution for a given customer, we assign probabilities of $\{0.05, 0.15, 0.60, 0.15, 0.05\}$ to $\{0, \bar{x}/2, \bar{x}, 3\bar{x}/2, 2\bar{x}\}$. To construct a low variability probability distribution for a given customer, we assign probabilities of $\{0.05, 0.9, 0.05\}$ to $\{\bar{x}/2, \bar{x}, 3\bar{x}/2\}$. For each of the 216 problem instances, we randomly generate 500 realizations according to the specified probability distributions for customer demand (a total of 108,000 realizations).

A common measure of the difficulty of a problem instance is the *filling rate*, defined as total

expected demand divided by total vehicle capacity (e.g., see Laporte et al. (2002)). The filling rates for our problem instances range from 0.85 to 3.92. In the stochastic VRP literature, problem instances with a filling rate greater than 1.0 are considered to be difficult to solve (Laporte et al., 2002). As the inclusion of a route duration limit adds a dimension not captured by the filling rate, we report these values mainly as a method to connect our problem instances to other problem instances treated in the literature. As we show in our computational experiments, our rollout policies perform close to an upper bound for "difficult" problem instances with a high filling rate provided that the duration limit is long.

6.2 Bounds and Benchmarks

As an upper bound on the value of an optimal policy, we sum the expected demand at all customers. We label this bound "Expected Demand." This is a valid upper bound for any policy because it is impossible to serve more demand than the total available demand. For several problem instances with short duration limits, it is impossible to reach some customers within the route duration limit (i.e., via a singleton route from the depot, to the customer, and back to the depot). When calculating the upper bound for these problem instances, we do not include the expected demand of impossible-to-reach customers. For problems with longer duration limits and larger vehicle capacities, we observe the upper bound to be tight. However, as the duration limit decreases and vehicle capacities become smaller, we observe that the upper bound becomes weaker.

We consider two benchmarks for our rollout policies. First, we use the value of high-quality fixed-route policies within the class of policies described in §4.1. We label this benchmark "Static Fixed Routes." Because fixed-route policies are commonly implemented in industry, our benchmark policies serve as a practice-based standard for our rollout policies. In Appendix C, we describe the simulated annealing procedure we employ to obtain the benchmark fixed-route policies. Our experience suggests that the simulated annealing procedure yields high-quality fixed-route policies for the problem instances we consider.

The second benchmark we employ is a rolling horizon procedure. At each decision epoch, the rolling horizon procedure executes the fixed-route heuristic from the current, pre-decision state. An action is selected by following the resulting fixed-route policy. We label this benchmark "Rolling Horizon." We also implement the rolling horizon procedure within the static decomposition scheme of §5.5. In this case, the rolling horizon procedure executes the fixed-route heuristic separately for each partition of customers. We label this benchmark "Rolling Horizon (D)." Although rolling horizon methods are well-known in the literature, the implementation via our local search-based fixed-route heuristic appears to be unique in the vehicle routing literature. The only

similar method of which we are aware is due to Novoa and Storer (2008), who consider recomputing fixed-route policies at each decision epoch via a greedy construction heuristic. Further, our local search scheme is an improvement over Fan et al. (2006), who use a simple cyclic heuristic in their static decomposition.

6.3 Results and Discussion

Table 3 provides aggregate results of our computational experiments on the problem instances described in §6.1. For each method listed in the first column, Table 3 displays the average demand served and the average number of CPU seconds required to select an action at each decision epoch. These results are aggregated over all problem parameters for 25-, 50-, 75-, and 100-customer problem instances. Thus, each entry in Table 3 represents results for 54 of the 216 problem instances. Aggregating results in this fashion provides a concise overview of our computational experiments. In Appendix D, we provide disaggregated results for each method.

The first portion of Table 3 displays the benchmarks and bounds described in §6.2. The remaining portions of Table 3 display results for our rollout policies without decomposition, with static decomposition, and with dynamic decomposition. The average demand served values reported for "Static Fixed Routes" are averages of the expected demand served (calculated using Algorithm 2 in Appendix B) across 54 problem instances. The average demand served values reported for "Expected Demand" are the average of the expected demands across 54 problem instances. Average demand served values for the remaining methods are averages of estimates of the expected demand served for each of 500 realizations in each of 54 problem instances.

Tables 4, 5, and 6 compare the performance of one-step, post-decision, and hybrid rollout, with dynamic decomposition and without decomposition, to methods from the literature. Specifically, Table 4 compares our rollout policies to "Static Fixed Routes," Table 5 to "Rolling Horizon," and Table 6 to one-step rollout with static decomposition. Each entry in these tables is the percent improvement over the corresponding method from the literature. As in Table 3, the values in Tables 4, 5, and 6 are aggregated over all problem parameters for 25-, 50-, 75-, and 100-customer problems.

The best performance relative to methods in the literature is achieved by one-step and post-decision rollout policies without decomposition. On average, these two methods improve upon the performance of static fixed routes by 3.26 percent, rolling horizon without decomposition by 0.83 percent, and one-step rollout with static decomposition by 2.05 percent. However, as problem size increases beyond 50 customers, executing one-step and post-decision rollout becomes computationally prohibitive given our computing resources. For example, applying one-step roll-

Table 3: Aggregate Results of Computational Experiments

Method	25 Customers Avg. Demand Served	25 Customers Avg. Per Epoch CPU	50 Customers Avg. Demand Served	50 Customers Avg. Per Epoch CPU	75 Customers Avg. Demand Served	75 Customers Avg. Per Epoch CPU	100 Customers Avg. Demand Served	100 Customers Avg. Per Epoch CPU
Benchmarks and Bounds								
Static Fixed Routes	308.87	N/A	609.19	N/A	1038.89	N/A	1362.01	N/A
Rolling Horizon (D)	311.49	0.02	612.58	0.03	1043.09	0.04	1368.64	0.07
Rolling Horizon	316.99	0.02	622.48	0.09	1059.74	0.53	1393.84	3.48
Expected Demand	372.67	N/A	752.17	N/A	1209.50	N/A	1610.67	N/A
No Decomposition								
One-Step Rollout	319.89	0.57	627.67	42.37	–	–	–	–
Post-Decision Rollout	319.47	0.16	627.50	9.22	–	–	–	–
Hybrid Rollout	318.11	0.04	623.38	0.63	1060.31	17.55	–	–
Static Decomposition								
One-Step Rollout	313.39	0.05	614.80	0.08	1045.45	0.10	1372.94	0.22
Post-Decision Rollout	313.38	0.02	614.69	0.03	1045.29	0.05	1372.81	0.16
Hybrid Rollout	313.25	0.02	614.42	0.04	1045.01	0.06	1372.36	0.10
Dynamic Decomposition								
One-Step Rollout	318.27	0.06	623.54	0.13	1060.83	0.59	1395.34	3.31
Post-Decision Rollout	318.32	0.03	623.43	0.12	1060.83	0.57	1395.30	3.38
Hybrid Rollout	318.13	0.03	623.11	0.11	1060.45	0.54	1394.94	3.52

Table 4: Percent Improvement Over Static Fixed Routes

Method	25 Customers	50 Customers	75 Customers	100 Customers	Average
		No Decomposition			
One-Step Rollout	3.57	3.03	–	–	3.30
Post-Decision Rollout	3.43	3.01	–	–	3.22
Hybrid Rollout	2.99	2.33	2.06	–	2.46
		Dynamic Decomposition			
One-Step Rollout	3.05	2.36	2.11	2.45	2.49
Post-Decision Rollout	3.06	2.34	2.11	2.44	2.49
Hybrid Rollout	3.00	2.28	2.08	2.42	2.44

Table 5: Percent Improvement Over Rolling Horizon

Method	25 Customers	50 Customers	75 Customers	100 Customers	Average
		No Decomposition			
One-Step Rollout	0.91	0.83	–	–	0.87
Post-Decision Rollout	0.78	0.81	–	–	0.79
Hybrid Rollout	0.35	0.15	0.05	–	0.18
		Dynamic Decomposition			
One-Step Rollout	0.40	0.17	0.10	0.11	0.20
Post-Decision Rollout	0.42	0.15	0.10	0.11	0.19
Hybrid Rollout	0.36	0.10	0.07	0.08	0.15

Table 6: Percent Improvement Over One-Step Rollout with Static Decomposition

Method	25 Customers	50 Customers	75 Customers	100 Customers	Average
		No Decomposition			
One-Step Rollout	2.07	2.09	–	–	2.08
Post-Decision Rollout	1.94	2.08	–	–	2.01
Hybrid Rollout	1.55	1.46	1.46	–	1.49
		Dynamic Decomposition			
One-Step Rollout	1.56	1.42	1.47	1.63	1.52
Post-Decision Rollout	1.57	1.42	1.49	1.64	1.53
Hybrid Rollout	1.56	1.41	1.48	1.65	1.52

out to some realizations of 50-customer problem instances required as much as 16 CPU hours to complete. Although such computing times may be acceptable for individual problem instances (i.e., many per epoch CPU times may be reasonable), given our current computing resources, it is difficult to carry out the full range of experiments required to evaluate one-step and post-decision rollout on problem instances with more than 50 customers.

Although hybrid rollout without decomposition does not perform as well as one-step and post-decision rollout, it is a computationally attractive alternative for 75-customer problem instances. On average, hybrid rollout without decomposition improves upon the performance of static fixed routes by 2.46 percent, rolling horizon without decomposition by 0.18 percent, and one-step rollout with static decomposition by 1.49 percent. Despite these improvements over benchmark methods, for 100-customer problem instances, hybrid rollout without decomposition faces the same computational obstacles encountered by one-step and post-decision rollout without decomposition.

Rollout policies combined with dynamic decomposition strike a balance between computational effort and solution quality. On average, rollout policies with dynamic decomposition improve upon the performance of static fixed routes by 2.47 percent, rolling horizon by 0.18 percent, and one-step rollout with static decomposition by 1.52 percent. These results suggest the performance of rollout policies combined with dynamic decomposition is comparable to hybrid rollout without decomposition. Moreover, on our hardware, per epoch CPU times are almost always less than five seconds for 100-customer problem instances. This suggests that rollout policies with dynamic decomposition are strong candidates for making real-time dynamic routing decisions in large-scale operations.

As can be seen in the disaggregated results in Appendix D, there are circumstances in which rollout performance is significantly greater than in the aggregated averages reported above. For example, when the duration limit is short and vehicle capacity is small, the average performance of one-step rollout without decomposition improves upon the average performance of static fixed routes by 10.08 percent, rolling horizon by 3.88 percent, and one-step rollout with static decomposition by 6.64 percent. For the same problem instances, the average performance of one-step rollout with dynamic decomposition improves upon the average performance of static fixed routes by 6.94 percent, rolling horizon by 1.09 percent, and one-step rollout with static decomposition by 4.27 percent.

7 Conclusion

We develop fixed-route based rollout policies to make dynamic routing decisions in the vehicle routing problem with stochastic demand and duration limits (VRPSDL). Our rollout policies perform favorably in comparison to practice-based fixed-route policies and a rolling horizon procedure. To counter the computational demands of larger problem instances, we execute our rollout policies using a dynamic decomposition scheme. Our computational results indicate that rollout combined with dynamic decomposition is able to achieve high quality solutions for large problem instances with reasonable run times, thereby suggesting that dynamic decomposition is a strong candidate for making real-time decisions in large-scale operations.

We suggest several avenues for future research. First, although we achieve good results with a local search-based fixed-route heuristic, further improvement may be obtained by considering a more intricate search procedure. For example, preliminary experience with a simulated annealing-based fixed route heuristic indicates that significant improvements can be realized on certain problem instances. However, due to the increased computation required to execute a simulated annealing-based fixed-route heuristic, our current computing resources prohibit an extensive test of this method. Efforts to speed the convergence of complex search procedures to high-quality policies would facilitate more exhaustive tests.

Second, consideration of less restrictive fixed-route policies may lead to improved results. Currently, we utilize the fixed-route policy class described in §4.1. Other potential policy classes include the paired-vehicle strategy of Ak and Erera (2007) and the recourse strategy of Novoa et al. (2006), both of which consider a wider range of recourse actions than the policy class in §4.1. However, policy classes with more complex recourse actions may lead to increased computing times, possibly limiting the ability to implement such methods in real time for large problem instances.

Third, the upper bound of expected demand is weak in many problem instances. Ideally, we would bound the objective with the value of an optimal policy. However, a primary motivation for the development of our rollout policies is the difficulty of obtaining optimal VRPSDL policies. To emphasize this, consider the work of Secomandi and Margot (2009), which addresses a single-vehicle VRP with stochastic demand. Provably optimal split-delivery policies are obtained only for problems with 15 or fewer customers. Moreover, the authors are only able to obtain perfect-information bounds on the value of optimal split-delivery policies for problem instances with 15 or fewer customers (although pseudo-bounds are computed for larger problem instances by obtaining valid bounds on unsplit-delivery policies). Due to the additional complexity introduced by multiple vehicles, obtaining optimal VRPSDL policies, or non-trivial bounds on the value of optimal VRPSDL policies, is a subject of future research. One option is to investigate a dual bound based

on an information relaxation (Brown et al., 2010). Yet, even a perfect-information relaxation is problematic to obtain. Despite the absence of uncertainty, a perfect-information relaxation of the VRPSDL remains a difficult combinatorial problem. Another alternative, motivated by Bertsimas et al. (1995), is to study the value of heuristic policies under a perfect information relaxation (e.g., execute rollout algorithms with known customer demands). Although such policies would not yield valid bounds, they may provide insight on the value of having more information.

Acknowledgments

The authors gratefully acknowledge the comments and suggestions of Nicola Secomandi, Dimitri Bertsekas, the associate editor, and three anonymous referees. The authors wish to thank the Information Technology Services at the University of Iowa and at Saint Louis University for access to high performance computing clusters. The authors are also appreciative for a grant from the Executive Council for Graduate and Professional Students at the University of Iowa. The grant allowed for the purchase of a computer on which the authors conducted preliminary experiments.

References

Ak, A. and A. Erera (2007). A paired-vehicle routing recourse strategy for the vehicle routing problem with stochastic demands. *Transportation Science 41*(2), 222–237.

American Trucking Association (2009). Allowance allocation policies in climate legislation: assisting consumers, investing in a clean energy future, and adapting to climate change. Statement made before the House Committee on Energy and Commerce Subcommittee on Energy and Environment. `http://www.truckline.com/AdvIssues/Energy/Cap%20and%20Trade%20Documents/Testimony%20on%20Allowance%20Allocation%20Policies%20in%20Climate%20Legislation.pdf`, Accessed on August 15, 2011.

Bent, R. and P. Van Hentenryck (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science 38*(4), 515–530.

Bertsekas, D. (2000). *Dynamic programming and optimal control* (2nd ed.), Volume I. Belmont, MA: Athena Scientific.

Bertsekas, D., J. Tsitsiklis, and C. Wu (1997). Rollout algorithms for combinatorial optimization. *Journal of Heuristics 3*(3), 245–262.

Bertsimas, D., P. Chervi, and M. Peterson (1995). Computational approaches to stochastic vehicle routing problems. *Transportation Science 29*(4), 342–352.

Bertsimas, D. and R. Demir (2002). An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science 48*(4), 550–565.

Bertsimas, D. and I. Popescu (2003). Revenue management in a dynamic network environment. *Transportation Science 37*(3), 257–277.

Bertsimas, D. and D. Simchi-Levi (1996). A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research 44*(2), 286–304.

Birattari, M., P. Balaprakash, T. Stützle, and M. Dorigo (2008). 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.

Brown, D., J. Smith, and P. Sun (2010). Information relaxations and duality in stochastic dynamic programs. *Operations Research 58*(4), 785–801.

Campbell, A. and B. Thomas (2008). Challenges and advances in a priori routing. In B. Golden, S. Raghavan, and E. Wasil (Eds.), *The vehicle routing problem: latest advances and new challenges*. New York, NY: Springer.

Campbell, A., D. Vandenbussche, and W. Hermann (2008). Routing for relief efforts. *Transportation Science 42*(2), 127–145.

Dror, M. (1993). Modeling vehicle routing with uncertain demands as a stochastic program: properties of the corresponding solution. *European Journal of Operational Research 64*, 432–441.

Dror, M., G. Laporte, and P. Trudeau (1989). Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science 23*(3), 166–176.

Erera, A. and C. Daganzo (2003). A dynamic scheme for stochastic vehicle routing. Working paper, Georgia Institute of Technology, `http://www2.isye.gatech.edu/people/faculty/Alan_Erera/pubs/ereradaganzo2003.pdf`, Accessed on April 25, 2009.

Erera, A., M. Savelsbergh, and E. Uyar (2009). Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks 54*(4), 270–283.

Fan, J., X. Wang, and H. Ning (2006). A multiple vehicles routing problem algorithm with stochastic demand. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, Volume 1, pp. 1688–1692.

Goodson, J. (2010). *Solution methodologies for vehicle routing problems with stochastic demand*. Ph. D. thesis, University of Iowa.

Goodson, J., J. Ohlmann, and B. Thomas (2012). Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research 217*(2), 312–323.

Goodson, J., B. Thomas, and J. Ohlmann (2012). A generalized rollout policy framework for stochastic dynamic programming. Working paper, Saint Louis University, `http://www.slu.edu/~goodson/papers/GoodsonRolloutFramework.pdf`, Accessed on April 30, 2012.

Guerriero, F. and M. Mancini (2003). A cooperative parallel rollout algorithm for the sequential ordering problem. *Parallel Computing 29*(5), 663–677.

Hvattum, L., A. Løkketangen, and G. Laporte (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science 40*(4), 421–438.

Hvattum, L., A. Løkketangen, and G. Laporte (2007). A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks 49*(4), 330–340.

Johnson, D., C. Aragon, L. McGeoch, and C. Schevon (1989). Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research 37*(6), 865–892.

Johnson, D., C. Aragon, L. McGeoch, and C. Schevon (1991). Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations Research 39*(3), 378–406.

Kindervater, G. and M. Savelsbergh (2003). Vehicle routing: handling edge exchanges. In E. Aarts and L. J. (Eds.), *Local search in combinatorial optimization*. Princeton, NJ: Princeton University Press.

Kirkpatrick, S., J. Gelatt, C., and M. Vecchi (1983). Optimization by simulated annealing. *Science 220*(4598), 671–680.

Laporte, G., F. Louveaux, and L. Van Hamme (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research 50*(3), 415–423.

Novoa, C., R. Berger, J. Linderoth, and R. Storer (2006). A set-partitioning-based model for the stochastic vehicle routing problem. Technical Report 06T-008, Lehigh University.

Novoa, C. and R. Storer (2008). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research 196*(2), 509–515.

Powell, W. (2007). *Approximate Dynamic Programming*. New York, NY: John Wiley and Sons.

Secomandi, N. (2000). Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers and Operations Research 27*(11-12), 1201–1225.

Secomandi, N. (2001). A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research 49*(5), 796–802.

Secomandi, N. (2003). Analysis of a rollout approach to sequencing problems with stochastic routing applications. *Journal of Heuristics 9*(4), 321–352.

Secomandi, N. and F. Margot (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research 57*(1), 214–230.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research 35*(2), 254–265.

Toth, P. and D. Vigo (2001). *The vehicle routing problem*. United States: SIAM.

van de Klundert, J. and L. Wormer (2010). ASAP: the after-salesman problem. *Manufacturing and Service Operations Management 12*(4), 627–641.

Wilson, R. (2011). 22nd annual state of logistics report. Council of Supply Chain Management Professionals.

Yang, W., K. Mathur, and R. Ballou (2000). Stochastic vehicle routing problem with restocking. *Transportation Science 34*(1), 99–112.

A Detailed Problem Formulation

In this appendix, we provide the details of the problem description given in §3. The state of the system captures all relevant information to make routing decisions and includes vehicle destinations, arrival times at vehicle destinations, remaining vehicle capacities, unserved customer demand, and the observed demand. We represent attributes of vehicle m in $\mathcal{M}$ by the tuple (l_m, t_m, q_m), where l_m in $\mathcal{N}$ is the destination of vehicle m (or its current location if the vehicle has arrived), t_m in $[0, L]$ is the time at which vehicle m will arrive at l_m, and q_m in $[0, Q]$ is the available capacity of vehicle m. Let $(l, t, q) = (l_m, t_m, q_m)_{m\in\mathcal{M}}$ denote the vector of vehicle attributes.

We characterize demand at customer n in $\mathcal{N} \setminus \{0\}$ by the pair (d_n, x_n), where d_n in $\{\{?\} \cup [0, \infty)\}$ is the unserved (pending) demand at customer n and x_n in $\{\{?\} \cup [0, \infty)\}$ is the observed demand at customer n. We include x_n in the state because demand not yet observed at unvisited customers may be dependent upon observed demands. Note that $(d_n, x_n) = (?, ?)$ indicates the demand at customer n is currently unknown. Furthermore, $d_n = ?$ if $x_n = ?$ and the demand served at customer n is $x_n - d_n$. Let $(d, x) = (d_n, x_n)_{n\in\mathcal{N}}$ denote the vector of unserved and observed demand.

At decision epoch k, a state s_k in state space $\mathcal{S} = \mathcal{N}^M \times [0, L]^M \times [0, Q]^M \times \{\{?\}\cup[0, \infty)\}^N \times \{\{?\} \cup [0, \infty)\}^N$ takes the form $((l, t, q), (d, x))$. The initial state is $s_0 = ((0, 0, Q)^M, (?, ?)^N)$. States included in the set of absorbing states $\mathcal{S}_K$ must meet several conditions. First, vehicle locations must be the depot. Second, we require all vehicles to return to the depot by the duration limit. Third, travel to customers with pending or unknown demand must be infeasible. The set of absorbing states is $\mathcal{S}_K = \{((0, t, Q)^M, (d, x)) : t_m \leq L, t_m = t_{m'} \ \forall \ m, m' \in \mathcal{M}, \text{if } d_n = ? \text{ or } d_n > 0 \text{ then } T_K + t(0, n) + t(n, 0) > L, d_n = ? \text{ if } x_n = ?, d_n \leq x_n \text{ if } x_n \neq ?\}$.

An action, taken at each decision epoch, is an assignment of the vehicles in $\mathcal{M}$ to locations in $\mathcal{N}$. Given pre-decision state s_k, which contains the vector of arrival times $(t_m)_{m\in\mathcal{M}}$, decision epoch k occurs at time $T_k = \min_{m\in\mathcal{M}} t_m$. Let $\mathcal{M}' = \arg\min_{m\in\mathcal{M}} t_m$ be the set of vehicles at decision epoch k currently at a customer location or at the depot (i.e., they are no longer en route). A vehicle in $\mathcal{M}'$ may be assigned to the depot to replenish capacity or may be assigned to a customer whose demand is either unknown or will be pending after customer demands are served in the current period. Vehicles in $\{\mathcal{M} \setminus \mathcal{M}'\}$ remain en route to their destinations, i.e., we do not consider diverting them.

Action a is an M-dimensional vector where the m^{th} element, a_m, is the action directing vehicle m in $\mathcal{M}$. The set of actions available in state s_k is

$$\mathcal{A}(s_k) = \Big\{ a \in \mathcal{N}^M :$$

$$a_m = l_m \,\forall\, m \in \{\mathcal{M} \setminus \mathcal{M}'\}, \tag{2}$$

$$a_m = 0 \,\forall\, \{m \in \mathcal{M}' : q_m \leq d_{l_m}, l_m \neq 0\}, \tag{3}$$

$$a_m \neq a_{m'} \,\forall\, \{m, m' \in \mathcal{M} : m \neq m', a_m \neq 0, a_{m'} \neq 0\}, \tag{4}$$

$$a_m \notin \{n \in \{\cup_{m' \in \mathcal{M}'} l_{m'} \setminus \{0\}\} : d_n \leq q_{\text{veh}(n)}\}, \tag{5}$$

$$a_m \notin \{n \in \{\mathcal{N} \setminus \{0\}\} : d_n = 0\}, \tag{6}$$

$$a_m \notin \{n \in \mathcal{N} : T_k + t(l_m, n) + t(n, 0) > L\}, \tag{7}$$

$$a_m \neq l_m \,\forall\, \{m \in \mathcal{M}' : l_m \neq 0\}, \tag{8}$$

$$|\{m \in \mathcal{M}' : l_m = 0, a_m = 0\}| = \max\{|\bar{\mathcal{M}}| - |\bar{\mathcal{N}}|, 0\}\Big\}. \tag{9}$$

Condition (2) requires that vehicles en route continue to their current destination. Condition (3) requires vehicles in $\mathcal{M}'$ to return to the depot if capacity will be depleted by serving customer demands at current locations. Condition (4) prevents assignment of multiple vehicles to customer locations, except in the case of assignments to the depot. Condition (5) disallows the assignment of vehicles to locations that will have zero demand after customer demands are served in the current period (veh(n) denotes the vehicle at location n, i.e., the vehicle m such that $l_m = n$). Similarly, condition (6) disallows the assignment of vehicles to locations where demand has already been fully served. Condition (7) prohibits the assignment of vehicles to locations that will result in violations of the route duration limit. For vehicles in $\mathcal{M}'$ not at the depot, condition (8) disallows assignment to the current location, i.e., these vehicles are not permitted to wait. In condition (9), the set $\{m \in \mathcal{M}' : l_m = 0, a_m = 0\}$ is the set of vehicles in $\mathcal{M}'$ assigned by action a to wait at the depot. The set $\bar{\mathcal{M}} = \{m \in \mathcal{M}' : l_m = 0 \vee (l_m \neq 0 \wedge a_m \neq 0)\}$ is the set of vehicles in $\mathcal{M}'$ that are either at the depot or are not at the depot and are not assigned to the depot by action a. The set $\bar{\mathcal{N}} = \{n \in \mathcal{N} \setminus \{\{\cup_{m \in \mathcal{M} \setminus \mathcal{M}'} l_m\} \cup \{0\}\} : T_k + t(0, n) + t(n, 0) \leq L \wedge [((d_n = ? \vee d_n > 0) \wedge n \notin \{\cup_{m \in \mathcal{M}'} l_m\}) \vee (d_n > q_{\text{veh}(n)} \wedge n \in \{\cup_{m \in \mathcal{M}'} l_m\})]\}$ is the set of customers not assigned to en route vehicles, that can be visited by a vehicle currently at the depot without violating the duration limit, and that will have pending demand after demand is served in the current period. Condition (9) only allows waiting at the depot if vehicles are unable to feasibly collect demand from the remaining set of customers, a situation that typically occurs toward the end of the time horizon when serving additional customer demand may not be possible.

Given that the state is currently s_k and that action a in $\mathcal{A}(s_k)$ is selected, a deterministic transition is made to post-decision state $s_k^a = ((l^a, t^a, q^a), (d^a, x^a))$ by updating vehicle destinations, vehicle arrival times, remaining vehicle capacities, and served demand. Vehicle destinations in

state s_k^a are $l_m^a = a_m$ for all m in $\mathcal{M}$. To accommodate vehicles waiting at the depot (see condition (9)), arrival times are calculated in two stages. First, for vehicles not assigned to wait at their current location, we set

$$t_m^a = \begin{cases} t_m + t(l_m, l_m^a), & \forall\, m \in \{m' \in \mathcal{M}' : l_{m'} \neq l_{m'}^a\}, \\ t_m, & \forall\, m \in \{\mathcal{M} \setminus \mathcal{M}'\}. \end{cases} \tag{10}$$

Second, for all m in $\{m' \in \mathcal{M}' : l_{m'} = l_{m'}^a\}$, we set arrival times to the minimum of the arrival times calculated in the first stage:

$$t_m^a = \min_{\substack{m' \in \{\{\mathcal{M} \setminus \mathcal{M}'\} \cup \\ \{m'' \in \mathcal{M}' : l_{m''} \neq l_{m''}^a\}\}}} t_{m'}^a. \tag{11}$$

Thus, vehicles assigned by action a to wait at the depot will "arrive" at the depot at the next decision epoch. We update remaining vehicle capacities to reflect the demand served at each location visited at the current epoch:

$$q_m^a = \begin{cases} \max\{q_m - d_{l_m}, 0\}, & \forall\, m \in \{m' \in \mathcal{M}' : l_{m'} \neq 0\}, \\ q_m, & \text{otherwise.} \end{cases} \tag{12}$$

In addition, we update unserved customer demands:

$$d_n^a = \begin{cases} \max\{d_n - q_{\text{veh}(n)}, 0\}, & \forall\, n \in \{\cup_{m \in \mathcal{M}'} l_m \setminus \{0\}\}, \\ d_n, & \text{otherwise.} \end{cases} \tag{13}$$

Observed demands remain unchanged, thus $x^a = x$.

A transition from pre-decision state s_k to post-decision state s_k^a results in a reward $R_k(s_k, a) = \sum_{m \in \mathcal{M}} R_{k,m}(s_k, a)$ equal to the served demand, where

$$R_{k,m}(s_k, a) = \begin{cases} \min\{d_{l_m}, q_m\}, & \forall\, m \in \{m' \in \mathcal{M}' : l_{m'} \neq 0\}, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

To be consistent with standard MDP notation, we notate the reward as a function of pre-decision state s_k and selected action a. Note, however, that the VRPSDL reward function depends only on demand served at current vehicle locations and available vehicle capacity, which are both known in s_k. Therefore, the VRPSDL reward function is independent of action a, which specifies vehicle destinations at the next epoch.

At decision epoch $k+1$, a transition is made from post-decision state s_k^a to pre-decision state $s_{k+1} = ((l, t, q), (d, x))$ by observing customer demand and replenishing the capacity of vehicles

at the depot. The joint distribution function F governs state transitions conditional on observed demand x. In this transition, vehicle destinations and arrival times remain unchanged, thus $l = l^a$ and $t = t^a$. We update capacities by

$$q_m = \begin{cases} Q, & \forall\, m \in \{m' \in \mathcal{M}' : l_{m'} = 0\}, \\ q_m^a, & \text{otherwise.} \end{cases} \tag{15}$$

Let $\mathcal{N}' = \{l_m^a : m \in \mathcal{M}', l_m^a \neq 0, x_{l_m}^a = ?\}$ be the set of customers with unknown demand at which a vehicle has arrived at the current decision epoch. If $\hat{x}$ is the observed demand at decision epoch $k+1$, then set

$$x_n = \begin{cases} \hat{x}_n, & n \in \mathcal{N}', \\ x_n^a, & \text{otherwise,} \end{cases} \tag{16}$$

and

$$d_n = \begin{cases} \hat{x}_n, & n \in \mathcal{N}', \\ d_n^a, & \text{otherwise.} \end{cases} \tag{17}$$

B Evaluation of Fixed-Route Policies

For notational convenience, when evaluating a fixed-route policy $\pi(v)$ from state s onward, we assume the first customer on each fixed route in v is given by the vehicles' locations/destinations in state s. Any set of fixed routes can easily be modified to satisfy this condition. For example, suppose the fixed route for vehicle m is $v^m = (v_1^m, v_2^m, \ldots, v_b^m, \ldots, v_{B^m}^m)$ and that in state s vehicle m is en route to customer v_b^m. The methods discussed in this section evaluate the fixed-route policy beginning at customer v_b^m, i.e., they address the expected demand served by the fixed-route policy characterized by route $(v_b^m, v_{b+1}^m, \ldots, v_{B^m}^m)$.

B.1 Exact Evaluation

Evaluating a fixed-route policy $\pi(v)$ is accomplished by calculating the expected demand served at each customer on each fixed route in v. In accordance with the literature on fixed-route policies, we assume customer demands are independent. This assumption results in separability by route when calculating the value of a policy.

Below, we develop a method to calculate the expected demand served at a single customer. Applying this method to each customer in a collection of fixed routes and summing the results

allows for the exact evaluation of a fixed-route policy. Denote by $Z_{v_b^m}$ the random amount of demand served at customer v_b^m and by $\mathbb{E}[Z_{v_b^m}]$ the expected demand served at customer v_b^m when fixed-route policy $\pi(v^m)$ is followed from state s onward. The total value of following a fixed-route policy $\pi(v)$ from state s onward is the sum of the expected demand served at each customer: $V^{\pi(v)}(s) = \sum_{m=1}^{M} \sum_{b=1}^{B^m} \mathbb{E}[Z_{v_b^m}]$.

The quantity $Z_{v_b^m}$ depends on vehicle capacity upon arrival to customer v_b^m (denoted $Q_{v_b^m}$) and arrival time at v_b^m (denoted $A_{v_b^m}$). Denote the random amount of demand at customer v_b^m by $x_{v_b^m}$. The quantities $Q_{v_b^m}$ and $A_{v_b^m}$ depend on vehicle capacity upon arrival to customer v_{b-1}^m and the demand at customer v_{b-1}^m; the quantity $A_{v_b^m}$ also depends on the arrival time of the vehicle to customer v_{b-1}^m. We separate the calculation of $Q_{v_b^m}$ and $A_{v_b^m}$ into three cases:

$$Q_{v_b^m} = \begin{cases} Q_{v_{b-1}^m} - x_{v_{b-1}^m}, & x_{v_{b-1}^m} < Q_{v_{b-1}^m}, \\ Q, & \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} = \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil, \\ \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil Q - x_{v_{b-1}^m} + Q_{v_{b-1}^m}, & \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} < \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil \end{cases} \tag{18}$$

and

$$A_{v_b^m} = \begin{cases} A_{v_{b-1}^m} + t(v_{b-1}^m, v_b^m), & x_{v_{b-1}^m} < Q_{v_{b-1}^m}, \\ A_{v_{b-1}^m} + \left(\frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} + 1 \right) t(v_{b-1}^m, 0) & \\ \quad + \left(\frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right) t(0, v_{b-1}^m) + t(0, v_b^m), & \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} = \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil, \\ A_{v_{b-1}^m} + t(v_{b-1}^m, v_b^m) + \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil & \\ \quad \times \left(t(v_{b-1}^m, 0) + t(0, v_{b-1}^m) \right), & \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} < \left\lceil \frac{x_{v_{b-1}^m} - Q_{v_{b-1}^m}}{Q} \right\rceil, \end{cases} \tag{19}$$

where the boundary conditions $v_1^m = l_m$, $Q_{v_1^m} = q_m$, and $A_{v_1^m} = t_m$ are given by the current state of vehicle m. In the first case, demand at customer v_{b-1}^m is less than vehicle capacity upon arrival to v_{b-1}^m, thus vehicle capacity is simply decremented by the amount of demand and the vehicle travels directly from v_{b-1}^m to v_b^m. The second and third cases account for situations where demand at customer v_{b-1}^m is greater than or equal to vehicle capacity upon arrival to v_{b-1}^m, thereby requiring return trips to the depot to replenish capacity. The required number of return trips is $\lceil (x_{v_{b-1}^m} - Q_{v_{b-1}^m})/Q \rceil$. In the second case, satisfying demand at v_{b-1}^m exactly depletes vehicle capacity, thus

requiring the vehicle to replenish at the depot one additional time and travel directly to customer v_b^m with full capacity. In the third case, there is some capacity remaining after serving demand at customer v_{b-1}^m. After making the necessary return trips to the depot, the vehicle travels directly from v_{b-1}^m to v_b^m with the remaining capacity.

To compute $Z_{v_b^m}$, we consider the route duration limit, noting that violations of the route duration limit result in zero demand served. Four cases are considered in the calculation:

$$Z_{v_b^m} = \begin{cases} 0, & A_{v_b^m} > L - t(v_b^m, 0), \\ \left\lfloor \frac{L - t(v_b^m,0) - A_{v_b^m}}{t(v_b^m,0) + t(0,v_b^m)} \right\rfloor Q + Q_{v_b^m}, & \left\lfloor \frac{L - t(v_b^m,0) - A_{v_b^m}}{t(v_b^m,0) + t(0,v_b^m)} \right\rfloor < \left\lceil \frac{x_{v_b^m} - Q_{v_b^m}}{Q} \right\rceil, \\ x_{v_b^m}, & x_{v_b^m} \leq Q_{v_b^m} \text{ and } A_{v_b^m} \leq L - t(v_b^m, 0), \\ x_{v_b^m}, & \left\lfloor \frac{L - t(v_b^m,0) - A_{v_b^m}}{t(v_b^m,0) + t(0,v_b^m)} \right\rfloor \geq \left\lceil \frac{x_{v_b^m} - Q_{v_b^m}}{Q} \right\rceil. \end{cases} \tag{20}$$

In the first case, zero demand is served because the route duration limit is violated. In the second case, only a portion of demand is served because the vehicle does not have enough time to make the return trips to the depot necessary to serve demand in full. In the third case, demand is served in full because the vehicle arrives prior to the route duration limit and with sufficient capacity. In the fourth case, demand is served in full because the vehicle has enough time to make any necessary replenishments.

For clarity in what follows, we sometimes write $Q_{v_b^m}$, $A_{v_b^m}$, and $Z_{v_b^m}$ as functions of the terms they depend on. Specifically, we denote by $Q_{v_b^m}(x, q)$ the capacity upon initial arrival to customer v_b^m when $x_{v_{b-1}^m} = x$ and $Q_{v_{b-1}^m} = q$; by $A_{v_b^m}(x, q, t)$ the time of initial arrival to customer v_b^m when $x_{v_{b-1}^m} = x$, $Q_{v_{b-1}^m} = q$, and $A_{v_{b-1}^m} = t$; and by $Z_{v_b^m}(x, q, t)$ the demand served at customer v_b^m when $x_{v_b^m} = x$, $Q_{v_b^m} = q$, and $A_{v_b^m} = t$.

Proposition 1 states three properties we use to assist in calculating the expected demand served by a fixed-route policy. Property (i) establishes a recursion to calculate the joint probability of initial arrival time at customer v_b^m and capacity upon initial arrival to customer v_b^m. Property (ii) provides an expression to calculate the expected demand served at customer v_b^m. We discuss property (iii) below. In each result, we assume customer demands follow a discrete probability distribution with a finite support. We denote by $f_{x_n}(\cdot)$ the probability mass function of demand at customer n.

Proposition 1 (Fixed-Route Policy Evaluation)**.** *The following properties hold when evaluating a fixed-route policy.*

(i) *The joint probability of arriving to customer v_b^m with available capacity q and at time t is*

$$\mathbb{P}\left\{Q_{v_b^m} = q, A_{v_b^m} = t\right\} = \sum_{q'}\sum_{t'}\sum_{x'} \mathbf{1}\left\{q = Q_{v_b^m}(x', q'), t = A_{v_b^m}(x', q', t')\right\} \times \mathbb{P}\left\{Q_{v_{b-1}^m} = q', A_{v_{b-1}^m} = t'\right\} \times f_{x_{v_{b-1}^m}}(x'), \quad (21)$$

where $\mathbf{1}\{\cdot\}$ is an indicator function that returns 1 if the statement $\{\cdot\}$ is true and 0 otherwise.

(ii) *The expected demand served at customer v_b^m is*

$$\mathbb{E}\left[Z_{v_b^m}\right] = \sum_{q}\sum_{t}\sum_{x} Z_{v_b^m}(x, q, t) \times \mathbb{P}\left\{Q_{v_b^m} = q, A_{v_b^m} = t\right\} \times f_{x_{v_b^m}}(x). \quad (22)$$

(iii) *Assume travel times $t(\cdot, \cdot)$ satisfy the triangle inequality. If $Z_{v_b^m} < x_{v_b^m}$, then $Z_{v_j^m} = 0$ for $j = b+1, b+2, \ldots, B^m$.*

Proof. We begin by proving property (i):

$$\begin{aligned}
&\mathbb{P}\{Q_{v_b^m} = q, A_{v_b^m} = t\} \\
&\quad = \sum_{q'}\sum_{t'}\sum_{x'} \mathbb{P}\left\{Q_{v_b^m} = q, A_{v_b^m} = t \middle| Q_{v_{b-1}^m} = q', A_{v_{b-1}^m} = t', x_{v_{b-1}^m} = x'\right\} && (23)\\
&\qquad \times \mathbb{P}\left\{Q_{v_{b-1}^m} = q', A_{v_{b-1}^m} = t', x_{v_{b-1}^m} = x'\right\} && (24)\\
&\quad = \sum_{q'}\sum_{t'}\sum_{x'} \mathbf{1}\left\{q = Q_{v_b^m}(x', q'), t = A_{v_b^m}(x', q', t')\right\} && (25)\\
&\qquad \times \mathbb{P}\left\{A_{v_{b-1}^m} = t', A_{v_{b-1}^m} = t'\right\} \times f_{x_{v_{b-1}^m}}(x'). && (26)
\end{aligned}$$

Equations (23) and (24) hold by the law of total probability. The conditional probability term in (23) is one if $q = Q_{v_b^m}(x', q')$ and $t = A_{v_b^m}(x', q', t')$ and is zero otherwise. Thus, the conditional probability term is equivalent to the indicator function in (25). Equation (26) follows from the assumed independence of customer demands.

We now prove property (ii):

$$\begin{aligned}
\mathbb{E}\left[Z_{v_b^m}\right] = &\sum_{q}\sum_{t}\sum_{x} \mathbb{E}\left[Z_{v_b^m} \middle| Q_{v_b^m} = q, A_{v_b^m} = t, x_{v_b^m} = x\right] && (27)\\
&\times \mathbb{P}\left\{Q_{v_b^m} = q, A_{v_b^m} = t, x_{v_b^m} = x\right\} && (28)
\end{aligned}$$

$$= \sum_{q} \sum_{t} \sum_{x} Z_{v_b^m}(x, q, t) \times \mathbb{P}\left\{Q_{v_b^m} = q, A_{v_b^m} = t\right\} \times f_{x_{v_b^m}}(x). \tag{29}$$

Equations (27) and (28) hold by the law of iterated expectations. Because the expectation in (27) is conditioned on q, t, and x, it is given directly by (20), which we denote by $Z_{v_b^m}(x, q, t)$. The remainder of (29) follows from the assumed independence of customer demands.

We now prove property (iii). Since $Z_{v_b^m} < x_{v_b^m}$, we have one of two cases. In the first case, $Z_{v_b^m} = 0$, implying by (20) that $A_{v_b^m} > L - t(v_b^m, 0)$. By the triangle inequality, this implies that for any $A_{v_{b+1}^m}$ that may result from (19), $A_{v_{b+1}^m} > L - t(v_{b+1}^m, 0)$. Thus, $Z_{v_{b+1}^m} = 0$. The same reasoning applies to the demand served at customers $v_{b+2}^m, \ldots, v_{B^m}^m$. In the second case, $0 < Z_{v_b^m} < x_{v_b^m}$, implying by (20) that $\lfloor (L - t(v_b^m, 0) - A_{v_b^m})/(t(v_b^m, 0) + t(0, v_b^m)) \rfloor < \lceil (x_{v_b^m} - Q_{v_b^m})/Q \rceil$, meaning that the number of replenishments required to satisfy demand in full is greater than the number of replenishments possible before violating the route duration limit L. Let t' be the time at which demand at customer v_b^m is served in full. It must be that $t' > L - t(v_b^m, 0)$. Then, by the triangle inequality, $t' > L - t(v_b^m, v_{b+1}^m) - t(v_{b+1}^m, 0)$, meaning that $A_{v_{b+1}^m} > L - t(v_{b+1}^m, 0)$. Thus, $Z_{v_{b+1}^m} = 0$. Then, using the same arguments as in the first case, demand served at customers $v_{b+2}^m, \ldots, v_{B^m}^m$ is also zero. □

Algorithm 2 employs properties (i) and (ii) of Proposition 1 to calculate the expected demand served at customer v_b^m. To ease the notation, we define $g(v_b^m, q, t) = \mathbb{P}\{Q_{v_b^m} = q, A_{v_b^m} = t\}$. The support of $x_{v_b^m}$ is denoted $\text{supp}(x_{v_b^m})$. Lines 1-4 of Algorithm 2 initialize the procedure. Line 1 assigns a probability of one to the current state of vehicle m. Line 2 assigns a probability of zero to all other possible states. Line 3 initializes joint capacity and arrival probabilities for all other customers to zero and line 4 initializes the expected demand served at customer v_b^m to zero; these quantities are updated as Algorithm 2 is executed. Lines 5-8 utilize property (i) of Proposition 1 to build the joint probabilities of capacities and arrival times at customers $v_1^m, v_2^m, \ldots, v_{b-1}^m$. Finally, lines 9-11 apply property (ii) of Proposition 1 to calculate the expected demand served at customer v_b^m.

Assuming integer demand values and travel times, the worst-case complexity of Algorithm 2 is $O(b \times Q \times L \times \max_{j=1,\ldots,b}\{|\text{supp}(x_{v_j^m})|\})$. Although the complexity is pseudo-polynomial, executing Algorithm 2 can still be computationally demanding. When travel times satisfy the triangle inequality, property (iii) of Proposition 1 provides a mechanism to reduce the computational burden. For a given capacity q', arrival time t', and demand x' at customer v_{j-1}^m, property (iii) of Proposition 1 shows that if $Z_{v_{j-1}^m}(x', q', t') < x'$, then demand served will be zero at all subsequent customers on any sample path, thereby making no contribution to the expected demand served. Thus, in line 7 of Algorithm 2, instead of iterating over the entire support, we can restrict the loop

Algorithm 2 Calculation of Expected Demand Served at Customer v_b^m

$g(v_1^m, q_m, t_m) \leftarrow 1$
$g(v_1^m, q, t) \leftarrow 0$ for all $(q, t) \neq (q_m, t_m)$
$g(v_j^m, q, t) \leftarrow 0$ for $j = 2, 3, \ldots, b$ and for all (q, t)
$\mathbb{E}[Z_{v_b^m}] \leftarrow 0$
for $j = 2, 3, \ldots, b$ **do**
    for $g(v_{j-1}^m, q', t') \neq 0$ **do**
        for $x' \in \text{supp}(x_{v_{j-1}^m})$ **do**
            $g(v_j^m, Q_{v_j^m}(x', q'), A_{v_j^m}(x', q', t')) \leftarrow$
                $g(v_j^m, Q_{v_j^m}(x', q'), A_{v_j^m}(x', q', t')) + g(v_{j-1}^m, q', t') \times f_{x_{v_{j-1}^m}}(x')$
for $g(v_b^m, q', t') \neq 0$ **do**
    for $x' \in \text{supp}(x_{v_b^m})$ **do**
        $\mathbb{E}[Z_{v_b^m}] \leftarrow \mathbb{E}[Z_{v_b^m}] + Z_{v_b^m}(x', q', t') \times g(v_b^m, q', t') \times f_{x_{v_b^m}}(x')$

to iterate over $\{x' \in \text{supp}(x_{v_{j-1}^m}) : Z(x', q', t') = x'\}$. When the duration limit is even moderately restrictive, modifying Algorithm 2 in this fashion significantly reduces the number of joint probability terms that need to be considered.

B.2 Evaluation via Simulation

Given P randomly generated demand realizations from F, $\hat{x}^1, \ldots, \hat{x}^P$, the expected demand served at customer v_b^m is estimated to be $\widehat{\mathbb{E}}[Z_{v_b^m}] = P^{-1} \sum_{p=1}^{P} Z_{v_b^m}(\hat{x}^p)$, where $Z_{v_b^m}(\hat{x}^p)$ denotes the demand served at customer v_b^m given the p^{th} demand realization $\hat{x}^p$. To calculate $Z_{v_b^m}(\hat{x}^p)$, replace x_b^m in (20) with $\hat{x}_{v_b^m}^p$, the p^{th} generated demand realization at customer v_b^m. The same is done in equations (18) and (19). Because $\widehat{\mathbb{E}}[Z_{v_b^m}]$ is the sample mean of demand served, it is an unbiased and consistent estimator of $\mathbb{E}[Z_{v_b^m}]$. The total value of following a fixed-route policy $\pi(v)$ from state s onward is estimated to be the sum of the estimated demand served at each customer:

$$\widehat{V}^{\pi(v)}(s) = \sum_{m=1}^{M} \sum_{b=1}^{B^m} \widehat{\mathbb{E}}\left[Z_{v_b^m}\right]. \tag{30}$$

When calculating $\widehat{V}^{\pi(\bar{v})}(s)$, where $\bar{v}$ is in the relocation neighborhood of v (see §4.2), it is not necessary to estimate the expected demand served at each customer in $\bar{v}$. Because the expected demand served at a customer depends only on the preceding sequence of customers, it is only necessary to estimate the expected demand served at customers where the preceding sequence has changed; demand served at other customers remains the same as in v. More generally, let v be the

current set of fixed routes and $\bar{v}$ a set of fixed routes in the relocation neighborhood of v obtained by relocating a customer v_i^m after customer $v_j^{m'}$. If $m \neq m'$, then it is only necessary to estimate $\mathbb{E}[Z_{\bar{v}_b^m}]$ for $b = i, \ldots, |v^m| - 1$ and $\mathbb{E}[Z_{\bar{v}_b^{m'}}]$ for $b = j+1, \ldots, |v^{m'}| + 1$. If $m = m'$ and $i < j$, then it is only necessary to estimate $\mathbb{E}[Z_{\bar{v}_b^m}]$ for $b = i, \ldots, |v^m|$. If $m = m'$ and $i > j$, then it is only necessary to estimate $\mathbb{E}[Z_{\bar{v}_b^m}]$ for $b = j+1, \ldots, |v^m|$. Expected demand served at other customers in $\bar{v}$ remains the same as in v.

In the computational experiments of §6, when estimating the expected demand served by a fixed-route policy in the local search procedure, we use $P = 30$ demand samples. We observe that 30 demand samples provide an acceptable tradeoff between solution quality and computing time. This decision is based on experiments using values of P ranging from five to several hundred. To further reduce computation, we utilize common random numbers by maintaining the same set of P demand samples during each execution of the local search procedure. As noted by Birattari et al. (2008), this practice reduces the variance of the difference between two solution values and also decreases computing time.

C Simulated Annealing for Benchmark Fixed-Route Policies

The benchmark fixed-route policies are obtained via a simulated annealing procedure. Simulated annealing is a local search algorithm in which non-improving moves are probabilistically accepted in an attempt to avoid becoming trapped in a low-quality, locally-optimal solution (Kirkpatrick et al., 1983; Johnson et al., 1989, 1991). Given a current set of fixed routes, the procedure generates a neighboring set of fixed routes via one of the following five neighborhood structures employed by Bent and Van Hentenryck (2004) in a simulated annealing algorithm for the VRP with time windows: two-exchange, Or-exchange, relocation, exchange, and crossover. A detailed explanation of these neighborhoods is given in Kindervater and Savelsbergh (2003). At each iteration of our simulated annealing implementation, we randomly generate a candidate set of fixed routes from a randomly selected neighborhood of the current set of fixed routes. Denote the policy induced by the current set of fixed routes by $\pi(v)$ and the policy induced by a neighboring set of fixed routes by $\pi(\bar{v})$. As in (30), we denote estimates of the expected demand served by following these policies from state s on as $\widehat{V}^{\pi(v)}(s)$ and $\widehat{V}^{\pi(\bar{v})}(s)$, respectively. We accept the neighboring fixed-route policy with probability $\exp(-(\widehat{V}^{\pi(v)}(s) - \widehat{V}^{\pi(\bar{v})}(s))^+/\tau)$, where $(y)^+ = y$ if $y > 0$ and 0 otherwise and τ is the current temperature, a control parameter in simulated annealing (Kirkpatrick et al., 1983).

The simulated annealing procedure begins with an initial set of M fixed routes obtained by

assigning customers to routes in a radial fashion with respect to the depot. We generate 7,000 neighbor solutions at each temperature, employ a temperature multiplier of 0.95, and begin with an initial temperature of 100. The procedure terminates after at least 100 temperature decrements and not updating the best-found solution for 75 successive temperature decrements. Our experience suggests that these parameters yield high-quality solutions for the problem instances we consider. We run the procedure 10 times and select as the benchmark policy the fixed routes with the largest expected demand served. When estimating the expected demand served by a fixed-route policy in the simulated annealing procedure, we use $P = 100$ demand samples. As in the local search procedure of §4.2, we use the same set of P demand samples throughout the simulated annealing procedure.

D Disaggregated Results of Computational Experiments

In this appendix, we report the disaggregated results of our computational experiments. Table 7 displays the expected demand served by the benchmark fixed-route policies. Tables 8 and 9 respectively display the expected demand served by the rolling horizon procedure with and without decomposition. Table 10 shows the expected demand for each problem instance; expected demand does not change with variability in customer demand. Tables 11, 12, and 13 depict disaggregated results for our rollout policies without decomposition. Tables 14, 15, and 16 present disaggregated results for our rollout policies with static decomposition. Tables 17, 18, and 19 display disaggregated results for our rollout policies with dynamic decomposition. In Tables 8-9 and 11-19, each entry is the average demand served by the respective method across 500 realizations of the problem instance. The values in Tables 7 and 10 are exact calculations. Table 20 is similar to Table 3, except that Table 20 displays the average number of CPU seconds required to execute each method for an entire instance; Table 3 only presents the average number of CPU seconds per decision epoch.

Table 7: Average Demand Served by Static Fixed Routes

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
Low Variability									
R101 (25)	131.63	214.84	276.42	226.58	315.96	331.41	294.40	331.99	332.00
C101 (25)	192.68	296.33	319.58	306.46	436.77	459.98	387.85	459.31	460.00
R101 (50)	337.23	481.95	623.42	500.62	704.77	720.72	637.13	720.92	720.99
C101 (50)	315.69	526.47	589.30	525.37	767.80	857.62	662.85	858.62	859.95
R101 (75)	568.79	779.50	970.15	808.54	1074.73	1078.83	1002.32	1078.79	1078.86
C101 (75)	733.15	1076.52	1286.69	1048.17	1357.75	1359.66	1268.60	1359.57	1359.96
R101 (100)	771.23	1083.56	1346.60	1090.28	1453.40	1457.25	1345.34	1457.59	1457.72
C101 (100)	858.89	1319.87	1627.62	1278.69	1776.58	1809.30	1599.42	1809.29	1809.83
Moderate Variability									
R101 (25)	128.69	205.72	270.71	213.26	296.70	326.31	272.12	331.24	332.00
C101 (25)	184.31	280.84	313.56	285.96	396.76	450.36	364.68	454.22	459.74
R101 (50)	303.84	459.44	593.97	471.01	663.38	718.58	599.46	718.80	720.87
C101 (50)	300.66	487.55	576.23	484.06	693.95	833.09	628.25	840.62	858.95
R101 (75)	522.43	738.35	928.74	762.79	1042.20	1078.05	952.69	1077.91	1078.72
C101 (75)	683.79	1008.79	1236.12	992.37	1315.48	1358.71	1201.19	1356.39	1359.52
R101 (100)	711.22	1011.36	1281.26	1028.11	1406.08	1456.74	1278.16	1456.09	1457.52
C101 (100)	808.02	1257.14	1561.16	1207.82	1671.49	1799.09	1510.13	1804.79	1808.89
High Variability									
R101 (25)	131.14	194.20	256.01	198.74	278.98	319.03	259.45	328.35	331.98
C101 (25)	173.81	257.66	302.04	264.96	368.17	434.54	341.96	437.41	458.94
R101 (50)	293.27	439.67	562.54	446.32	627.70	709.53	571.36	714.48	720.80
C101 (50)	283.10	455.82	552.80	455.12	652.22	790.08	593.24	805.53	858.63
R101 (75)	485.48	706.25	893.09	730.22	996.87	1076.39	905.63	1076.02	1078.26
C101 (75)	644.45	964.15	1165.14	939.60	1255.78	1357.45	1128.58	1352.25	1359.61
R101 (100)	665.41	971.54	1226.79	979.67	1356.60	1453.37	1218.94	1453.61	1456.93
C101 (100)	764.04	1186.03	1469.21	1139.66	1566.30	1765.16	1415.73	1783.52	1808.56

Table 8: Average Demand Served by Rolling Horizon with Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
Low Variability									
R101 (25)	137.27	219.65	276.98	229.22	317.62	331.54	298.33	332.04	332.04
C101 (25)	193.17	296.86	319.69	308.61	445.77	459.64	388.77	459.04	459.64
R101 (50)	341.97	489.16	625.18	507.16	709.16	720.82	645.94	721.12	721.15
C101 (50)	317.44	530.69	590.17	527.55	786.20	857.94	665.96	858.24	859.89
R101 (75)	580.62	790.87	974.40	818.52	1075.89	1078.36	1011.37	1078.41	1078.42
C101 (75)	735.12	1084.63	1288.37	1053.50	1358.37	1359.93	1275.02	1359.95	1360.03
R101 (100)	779.34	1101.82	1357.90	1106.99	1455.62	1458.41	1357.89	1458.43	1458.53
C101 (100)	862.48	1331.90	1634.40	1287.92	1790.56	1810.64	1613.97	1810.56	1810.81
Moderate Variability									
R101 (25)	130.81	207.93	270.52	218.77	304.51	328.58	281.70	331.35	331.61
C101 (25)	190.64	284.23	315.42	288.50	405.29	455.67	366.19	454.55	459.07
R101 (50)	306.78	463.29	599.07	479.60	674.48	719.05	610.88	719.85	720.43
C101 (50)	302.49	492.65	577.03	492.69	708.24	838.97	631.90	845.26	858.09
R101 (75)	532.65	752.04	935.02	774.85	1049.17	1077.20	960.59	1077.28	1077.55
C101 (75)	685.46	1014.33	1240.19	998.85	1325.93	1359.70	1212.07	1358.63	1360.37
R101 (100)	727.13	1020.29	1288.34	1042.55	1418.09	1457.02	1294.14	1456.60	1457.23
C101 (100)	813.09	1266.22	1565.47	1218.57	1690.17	1805.70	1520.58	1807.35	1809.57
High Variability									
R101 (25)	132.55	197.06	257.73	205.11	289.03	319.97	266.63	329.44	331.71
C101 (25)	173.73	260.71	303.59	266.89	371.79	435.69	346.24	442.17	459.08
R101 (50)	293.09	442.18	562.18	455.23	632.74	708.14	581.68	711.06	716.85
C101 (50)	285.14	456.00	552.72	461.10	653.70	789.17	599.19	808.10	854.30
R101 (75)	490.59	710.10	899.77	736.22	1003.19	1073.66	914.74	1073.51	1075.44
C101 (75)	648.07	965.41	1170.11	944.13	1260.48	1357.97	1133.24	1356.15	1360.29
R101 (100)	672.88	975.18	1226.37	998.23	1359.12	1448.77	1231.22	1450.71	1452.55
C101 (100)	766.18	1188.06	1470.60	1144.68	1580.18	1776.49	1425.86	1787.21	1805.89

Table 9: Average Demand Served by Rolling Horizon without Decomposition

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	140.52	219.88	277.03	229.68	318.17	331.71	299.64	332.04	332.04
C101 (25)	209.28	296.98	320.03	310.74	452.05	459.64	391.39	459.24	459.64
R101 (50)	350.81	492.65	625.31	509.06	709.83	721.13	648.59	721.15	721.15
C101 (50)	319.58	531.85	593.54	532.46	792.23	858.97	671.07	859.83	859.92
R101 (75)	582.42	794.76	976.47	824.36	1077.45	1078.40	1018.85	1078.42	1078.42
C101 (75)	749.88	1089.16	1291.66	1060.88	1359.11	1359.99	1285.85	1360.04	1360.05
R101 (100)	789.77	1103.90	1362.27	1113.41	1456.67	1458.59	1376.21	1458.60	1458.59
C101 (100)	878.75	1339.14	1636.33	1301.72	1795.10	1810.78	1636.86	1810.83	1810.84
				Moderate Variability					
R101 (25)	133.50	208.98	270.52	221.37	309.99	330.05	287.27	331.59	331.61
C101 (25)	206.80	288.78	318.54	297.63	419.18	456.68	380.73	458.34	459.28
R101 (50)	316.18	466.69	602.90	490.59	680.11	719.75	626.67	720.37	720.53
C101 (50)	319.59	499.83	581.52	516.03	733.05	847.34	655.15	854.22	858.62
R101 (75)	555.44	762.86	946.09	796.72	1059.98	1077.56	993.32	1077.58	1077.58
C101 (75)	723.91	1039.45	1268.25	1029.13	1341.16	1360.45	1257.05	1360.49	1360.55
R101 (100)	757.97	1058.78	1303.08	1079.08	1433.99	1457.26	1340.49	1457.36	1457.37
C101 (100)	857.42	1291.26	1594.82	1268.75	1743.40	1809.16	1588.52	1809.92	1809.97
				High Variability					
R101 (25)	136.55	201.34	257.97	211.86	296.36	324.61	275.62	331.26	331.73
C101 (25)	193.52	276.17	311.39	288.42	396.53	451.21	368.53	453.93	460.01
R101 (50)	303.71	451.62	565.27	473.95	652.44	711.11	609.13	716.22	716.93
C101 (50)	300.02	478.45	564.60	494.88	701.92	814.45	638.61	836.66	855.57
R101 (75)	520.90	736.35	914.94	768.96	1034.07	1075.55	964.08	1075.58	1075.64
C101 (75)	697.37	1006.75	1217.73	1009.46	1314.32	1359.97	1219.56	1360.54	1360.58
R101 (100)	723.59	1018.95	1257.22	1053.17	1403.36	1452.71	1304.16	1452.86	1452.86
C101 (100)	828.75	1227.63	1520.97	1233.83	1675.34	1800.65	1534.04	1803.53	1806.58

Table 10: Expected Demand

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
R101 (25)	332.00	332.00	332.00	332.00	332.00	332.00	332.00	332.00	332.00
C101 (25)	320.00	320.00	320.00	460.00	460.00	460.00	460.00	460.00	460.00
R101 (50)	721.00	721.00	721.00	721.00	721.00	721.00	721.00	721.00	721.00
C101 (50)	630.00	630.00	630.00	860.00	860.00	860.00	860.00	860.00	860.00
R101 (75)	1079.00	1079.00	1079.00	1079.00	1079.00	1079.00	1079.00	1079.00	1079.00
C101 (75)	1300.00	1300.00	1300.00	1360.00	1360.00	1360.00	1360.00	1360.00	1360.00
R101 (100)	1438.00	1438.00	1438.00	1458.00	1458.00	1458.00	1458.00	1458.00	1458.00
C101 (100)	1690.00	1690.00	1690.00	1810.00	1810.00	1810.00	1810.00	1810.00	1810.00

Table 11: Average Demand Served by One-Step Rollout without Decomposition

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	155.05	220.54	277.09	232.76	319.62	332.00	303.63	332.04	332.04
C101 (25)	213.72	297.27	320.02	313.65	452.66	459.64	392.37	459.64	459.64
R101 (50)	355.23	494.19	625.80	517.13	713.42	721.15	656.91	721.15	721.15
C101 (50)	331.73	534.18	594.89	541.84	799.66	858.76	678.20	859.86	859.92
				Moderate Variability					
R101 (25)	143.15	209.81	273.21	226.84	313.36	330.57	293.90	331.58	331.61
C101 (25)	210.58	291.01	319.07	305.50	431.14	458.96	382.88	458.50	459.28
R101 (50)	330.45	474.34	603.76	501.38	691.49	720.11	637.17	720.48	720.53
C101 (50)	320.02	504.01	582.10	527.23	749.10	850.57	662.09	857.16	858.75
				High Variability					
R101 (25)	144.00	203.41	260.00	218.17	301.71	326.68	282.02	331.42	331.73
C101 (25)	199.35	279.76	312.29	296.49	403.18	454.88	372.22	452.35	460.01
R101 (50)	315.49	461.48	571.41	484.27	661.52	712.57	616.46	716.45	716.93
C101 (50)	308.86	483.24	565.32	501.75	714.57	827.35	645.01	840.19	855.58

Table 12: Average Demand Served by Post-Decision Rollout without Decomposition

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	155.14	220.00	277.09	232.76	319.28	332.00	303.90	332.04	332.04
C101 (25)	205.95	297.21	320.03	313.73	453.54	459.64	391.76	459.64	459.64
R101 (50)	355.67	494.67	626.04	516.38	715.33	721.15	657.39	721.15	721.15
C101 (50)	330.45	534.06	593.03	542.14	799.69	858.71	679.88	859.88	859.92
				Moderate Variability					
R101 (25)	142.63	209.59	273.21	226.49	312.42	330.23	293.39	331.52	331.61
C101 (25)	208.08	290.13	319.22	304.49	429.44	458.54	383.20	458.71	459.28
R101 (50)	330.33	475.66	604.21	500.46	690.61	720.09	638.46	720.45	720.53
C101 (50)	319.05	502.50	582.49	524.25	748.47	850.48	662.07	856.95	858.75
				High Variability					
R101 (25)	143.50	202.53	260.02	216.35	302.30	326.96	282.07	331.44	331.73
C101 (25)	200.30	278.42	311.32	295.42	403.95	454.54	370.29	452.63	460.01
R101 (50)	316.20	462.03	570.23	484.55	660.90	713.41	617.06	716.36	716.93
C101 (50)	309.49	483.51	565.53	501.34	711.78	823.75	645.00	839.05	855.58

Table 13: Average Demand Served by Hybrid Rollout without Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	149.24	220.10	277.09	232.21	318.97	331.81	302.46	332.04	332.04
C101 (25)	213.80	296.97	320.03	311.65	451.94	459.64	391.03	459.39	459.64
R101 (50)	350.86	494.06	625.58	510.60	710.62	721.13	650.25	721.15	721.15
C101 (50)	322.27	532.59	596.77	531.98	797.26	858.94	671.33	859.79	859.91
R101 (75)	586.28	795.70	977.28	826.46	1077.67	1078.41	1023.13	1078.42	1078.42
C101 (75)	753.45	1092.23	1291.37	1060.48	1359.28	1359.99	1285.90	1360.05	1360.05
				Moderate Variability					
R101 (25)	141.38	209.20	270.74	223.66	311.59	329.99	290.16	331.54	331.61
C101 (25)	209.77	289.18	318.81	301.90	421.44	457.39	380.67	458.35	459.28
R101 (50)	323.27	467.44	603.95	493.92	681.69	719.79	629.40	720.30	720.53
C101 (50)	317.48	500.35	581.65	514.28	736.46	846.98	654.61	855.19	858.62
R101 (75)	557.47	764.88	947.67	801.21	1060.22	1077.41	994.77	1077.56	1077.58
C101 (75)	724.66	1038.14	1266.35	1030.25	1340.35	1360.48	1254.94	1360.45	1360.55
				High Variability					
R101 (25)	141.35	202.95	259.81	215.02	299.73	325.89	279.73	331.36	331.73
C101 (25)	195.57	274.97	311.05	288.48	393.22	450.54	367.84	451.89	460.01
R101 (50)	308.06	454.74	571.49	477.59	654.35	712.14	608.70	715.90	716.93
C101 (50)	301.57	477.69	564.18	491.80	703.16	814.09	635.66	836.88	855.54
R101 (75)	525.73	738.71	915.64	772.24	1037.18	1075.59	966.28	1075.58	1075.64
C101 (75)	697.05	1007.06	1216.34	1005.40	1310.54	1360.04	1217.06	1360.45	1360.58

Table 14: Average Demand Served by One-Step Rollout with Static Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	140.63	219.52	277.00	231.05	319.12	331.54	301.88	332.04	332.04
C101 (25)	197.48	296.86	319.69	310.31	447.91	459.64	389.80	459.04	459.64
R101 (50)	344.06	490.30	625.24	508.92	709.72	720.93	649.84	721.12	721.15
C101 (50)	317.46	533.34	590.17	530.07	791.82	857.92	666.92	859.17	859.91
R101 (75)	584.01	791.94	974.73	827.77	1076.26	1078.35	1016.50	1078.42	1078.42
C101 (75)	736.81	1086.75	1288.37	1054.55	1358.36	1359.93	1276.63	1359.99	1360.05
R101 (100)	782.76	1104.59	1360.02	1113.61	1455.69	1458.47	1363.90	1458.43	1458.55
C101 (100)	863.40	1335.71	1634.51	1293.76	1793.78	1810.69	1625.16	1810.73	1810.81
				Moderate Variability					
R101 (25)	139.81	208.01	273.09	222.09	307.80	328.48	286.75	331.37	331.61
C101 (25)	191.46	284.23	315.42	292.30	410.09	455.90	369.23	454.72	459.07
R101 (50)	312.03	463.42	599.40	485.28	678.88	719.05	616.95	719.90	720.43
C101 (50)	303.57	492.40	577.03	497.40	715.53	839.21	636.47	846.37	858.13
R101 (75)	546.82	754.55	935.41	782.85	1049.72	1077.36	966.30	1077.37	1077.56
C101 (75)	690.66	1015.22	1240.33	1000.75	1325.95	1359.73	1217.85	1358.67	1360.37
R101 (100)	738.37	1027.34	1290.48	1052.75	1420.65	1457.02	1302.57	1456.71	1457.36
C101 (100)	814.45	1275.72	1566.98	1225.48	1705.62	1805.67	1531.68	1807.03	1809.62
				High Variability					
R101 (25)	137.28	198.85	260.05	211.30	292.10	320.56	271.23	329.94	331.71
C101 (25)	180.06	260.71	303.59	272.10	373.93	436.11	350.11	447.84	459.08
R101 (50)	295.67	444.18	562.40	460.08	638.37	708.38	586.57	711.93	716.84
C101 (50)	287.41	461.85	552.72	468.04	660.16	789.14	604.16	817.66	854.37
R101 (75)	498.09	712.06	903.31	744.61	1008.21	1073.71	924.09	1074.06	1075.48
C101 (75)	650.53	966.30	1169.94	948.10	1265.66	1358.07	1140.14	1355.99	1360.47
R101 (100)	683.94	980.09	1227.50	1011.39	1364.37	1449.43	1241.87	1450.76	1452.80
C101 (100)	771.32	1196.41	1471.10	1154.53	1599.86	1776.68	1432.99	1787.65	1806.11

Table 15: Average Demand Served by Post-Decision Rollout with Static Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
Low Variability									
R101 (25)	140.63	220.19	277.00	230.98	319.08	331.54	301.87	332.04	332.04
C101 (25)	197.46	296.86	319.69	310.31	447.91	459.64	389.96	459.04	459.64
R101 (50)	344.06	490.30	625.21	509.26	709.74	720.88	649.79	721.12	721.15
C101 (50)	317.45	533.34	590.17	530.10	791.82	857.95	666.92	859.17	859.91
R101 (75)	584.19	791.93	974.69	826.70	1076.31	1078.36	1016.78	1078.42	1078.42
C101 (75)	736.82	1086.70	1288.37	1054.22	1358.31	1359.93	1276.56	1360.02	1360.05
R101 (100)	783.44	1104.64	1360.33	1113.35	1455.70	1458.47	1363.79	1458.43	1458.54
C101 (100)	863.40	1335.71	1634.51	1294.05	1793.74	1810.72	1624.83	1810.73	1810.83
Moderate Variability									
R101 (25)	139.81	208.01	273.09	221.83	307.77	328.50	286.90	331.40	331.61
C101 (25)	191.45	284.23	315.42	292.32	410.07	455.90	369.29	454.71	459.07
R101 (50)	312.81	463.49	599.40	484.62	677.90	719.06	616.62	719.93	720.43
C101 (50)	303.63	492.31	577.03	497.19	715.33	839.20	636.63	845.88	858.09
R101 (75)	546.91	754.69	935.31	782.99	1049.39	1077.33	965.96	1077.37	1077.56
C101 (75)	690.69	1013.60	1240.32	999.94	1325.95	1359.76	1217.79	1358.73	1360.46
R101 (100)	738.96	1026.94	1290.30	1051.53	1420.64	1457.03	1301.57	1456.82	1457.30
C101 (100)	814.48	1273.15	1567.05	1224.92	1705.82	1805.68	1531.64	1807.05	1809.69
High Variability									
R101 (25)	137.28	199.77	260.05	210.27	292.01	320.50	271.28	329.98	331.71
C101 (25)	180.06	260.71	303.59	271.11	374.29	436.11	349.70	447.88	459.08
R101 (50)	295.66	444.15	562.40	458.83	638.31	708.39	584.77	711.74	716.84
C101 (50)	287.33	461.85	552.72	468.08	660.09	789.18	603.56	817.11	854.40
R101 (75)	497.40	712.01	903.69	743.40	1008.06	1073.75	922.06	1073.92	1075.49
C101 (75)	650.05	966.53	1169.58	948.29	1265.75	1357.97	1139.76	1356.03	1360.42
R101 (100)	683.45	979.26	1227.30	1010.29	1366.30	1449.89	1240.39	1450.85	1452.76
C101 (100)	770.91	1195.83	1471.10	1154.08	1600.14	1776.89	1432.50	1787.64	1806.17

Table 16: Average Demand Served by Hybrid Rollout with Static Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
Low Variability									
R101 (25)	140.63	220.19	277.00	231.88	319.08	331.53	301.76	332.04	332.04
C101 (25)	197.48	296.86	319.69	310.31	447.89	459.64	389.98	459.04	459.64
R101 (50)	343.77	490.22	625.21	509.87	709.34	720.88	648.47	721.12	721.15
C101 (50)	317.34	533.34	590.17	529.83	791.64	857.96	666.93	858.80	859.91
R101 (75)	585.20	791.71	974.56	824.13	1076.26	1078.36	1015.89	1078.42	1078.42
C101 (75)	736.69	1086.67	1288.37	1054.43	1358.30	1359.93	1277.32	1359.99	1360.05
R101 (100)	783.26	1099.78	1358.91	1111.95	1455.63	1458.48	1363.49	1458.45	1458.53
C101 (100)	863.40	1335.71	1634.42	1293.61	1793.63	1810.71	1624.43	1810.73	1810.81
Moderate Variability									
R101 (25)	139.75	208.01	273.03	221.57	307.70	328.43	285.65	331.37	331.61
C101 (25)	191.39	284.23	315.42	292.22	409.93	455.75	369.34	454.69	459.07
R101 (50)	312.74	463.20	599.43	483.55	677.55	718.87	616.63	719.89	720.43
C101 (50)	303.57	492.85	577.03	497.10	714.37	838.94	636.34	846.14	858.10
R101 (75)	541.46	753.80	935.20	782.53	1049.42	1077.30	963.94	1077.33	1077.56
C101 (75)	690.83	1015.18	1240.24	999.53	1325.85	1359.71	1216.77	1358.73	1360.46
R101 (100)	738.28	1025.45	1289.43	1050.29	1419.73	1457.03	1301.13	1456.64	1457.26
C101 (100)	814.37	1273.22	1566.94	1224.66	1705.42	1805.65	1530.99	1807.30	1809.66
High Variability									
R101 (25)	136.35	198.82	260.05	210.12	291.59	320.67	271.01	330.01	331.71
C101 (25)	179.96	260.71	303.59	271.11	374.19	435.89	349.33	445.67	459.08
R101 (50)	295.68	442.77	562.41	458.76	637.53	708.52	584.03	711.70	716.88
C101 (50)	287.28	456.91	552.72	467.26	659.54	789.17	603.02	817.54	854.33
R101 (75)	497.70	711.79	901.24	744.38	1007.50	1073.76	920.98	1073.69	1075.49
C101 (75)	649.88	966.48	1169.93	948.03	1265.29	1357.97	1139.26	1356.38	1360.44
R101 (100)	682.79	978.20	1227.18	1010.26	1364.28	1449.00	1240.33	1450.90	1452.73
C101 (100)	770.91	1195.49	1470.72	1154.12	1599.29	1776.59	1432.13	1787.20	1805.98

Table 17: Average Demand Served by One-Step Rollout with Dynamic Decomposition

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	152.11	219.74	277.09	232.63	319.14	331.70	302.73	332.04	332.04
C101 (25)	209.83	296.97	320.02	312.27	451.39	459.64	391.08	459.24	459.64
R101 (50)	351.67	493.06	625.39	510.52	710.83	721.12	650.84	721.12	721.15
C101 (50)	319.45	533.70	593.32	532.63	795.87	858.79	669.99	859.75	859.92
R101 (75)	584.68	794.93	977.95	826.95	1077.39	1078.42	1021.12	1078.42	1078.42
C101 (75)	751.00	1092.96	1291.51	1061.38	1359.14	1359.99	1286.45	1360.04	1360.05
R101 (100)	792.43	1106.24	1362.62	1117.00	1456.68	1458.60	1376.81	1458.59	1458.60
C101 (100)	878.64	1341.20	1636.07	1301.80	1798.22	1810.78	1636.46	1810.80	1810.84
				Moderate Variability					
R101 (25)	141.56	209.13	273.18	224.39	311.90	330.35	291.36	331.57	331.61
C101 (25)	208.37	288.69	318.66	298.25	420.88	457.27	381.17	458.25	459.28
R101 (50)	320.19	467.32	603.27	494.67	681.22	719.84	631.75	720.22	720.53
C101 (50)	321.19	499.96	581.76	516.89	737.07	847.32	656.22	854.80	858.69
R101 (75)	558.81	764.84	946.31	800.42	1059.83	1077.55	996.08	1077.58	1077.58
C101 (75)	723.07	1039.51	1267.74	1034.08	1340.92	1360.46	1257.67	1360.51	1360.55
R101 (100)	761.28	1063.79	1303.76	1084.71	1436.19	1457.26	1346.12	1457.36	1457.37
C101 (100)	857.15	1296.50	1595.76	1269.63	1747.43	1809.22	1592.28	1809.90	1809.97
				High Variability					
R101 (25)	142.87	201.72	260.50	216.09	299.21	325.57	279.42	331.22	331.73
C101 (25)	194.71	274.96	310.96	288.85	398.85	451.04	370.33	453.53	460.01
R101 (50)	305.79	454.13	572.17	477.24	655.32	711.08	611.74	716.01	716.93
C101 (50)	300.39	479.43	564.66	494.95	704.28	813.91	639.09	836.32	855.58
R101 (75)	524.07	739.55	916.86	774.19	1036.06	1075.54	968.85	1075.57	1075.64
C101 (75)	699.44	1005.64	1218.84	1010.57	1315.31	1360.03	1223.25	1360.56	1360.58
R101 (100)	726.02	1022.61	1256.95	1059.16	1404.77	1452.69	1311.49	1452.86	1452.86
C101 (100)	828.12	1236.29	1522.65	1232.56	1673.39	1800.49	1537.48	1803.57	1806.58

Table 18: Average Demand Served by Post-Decision Rollout with Dynamic Decomposition

Duration		short			medium			long	
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	152.30	220.42	277.09	232.23	318.95	331.71	301.43	332.04	332.04
C101 (25)	210.30	296.98	319.99	311.50	451.93	459.64	390.78	459.35	459.64
R101 (50)	351.03	493.15	625.38	510.71	711.02	721.10	651.40	721.13	721.15
C101 (50)	319.49	533.49	593.68	532.50	795.86	858.96	670.47	859.79	859.92
R101 (75)	584.63	795.47	977.87	827.94	1077.50	1078.42	1022.19	1078.42	1078.42
C101 (75)	751.69	1092.90	1291.44	1061.46	1359.18	1359.99	1286.67	1360.04	1360.05
R101 (100)	792.17	1106.24	1361.62	1115.98	1456.68	1458.60	1376.29	1458.60	1458.60
C101 (100)	878.65	1340.69	1636.05	1301.51	1798.83	1810.78	1638.35	1810.80	1810.84
				Moderate Variability					
R101 (25)	142.18	209.20	273.03	223.50	312.21	330.23	290.31	331.58	331.61
C101 (25)	209.97	289.22	318.37	298.50	419.99	457.12	380.86	458.34	459.28
R101 (50)	320.02	467.30	602.20	494.25	681.12	719.80	631.70	720.27	720.53
C101 (50)	320.74	499.90	581.61	516.37	737.76	848.08	655.64	855.26	858.58
R101 (75)	556.63	764.63	946.44	799.89	1059.96	1077.54	995.86	1077.58	1077.58
C101 (75)	724.69	1040.92	1265.05	1031.45	1340.71	1360.47	1258.40	1360.46	1360.55
R101 (100)	761.61	1063.70	1304.01	1084.20	1435.75	1457.26	1342.67	1457.37	1457.37
C101 (100)	857.04	1297.74	1596.51	1269.90	1749.72	1809.38	1590.03	1809.83	1809.97
				High Variability					
R101 (25)	143.20	202.24	260.56	214.95	299.49	325.66	278.85	331.41	331.73
C101 (25)	196.03	276.49	310.91	288.84	399.60	451.03	370.62	453.58	460.01
R101 (50)	304.67	454.50	569.24	475.51	654.32	711.51	610.36	716.09	716.93
C101 (50)	300.79	479.47	564.57	494.82	707.34	812.79	638.48	836.77	855.57
R101 (75)	524.81	739.20	917.73	773.73	1037.81	1075.61	967.80	1075.58	1075.64
C101 (75)	698.91	1005.14	1217.85	1010.86	1314.33	1360.03	1225.84	1360.56	1360.58
R101 (100)	725.95	1021.28	1258.29	1057.99	1404.80	1452.70	1309.15	1452.86	1452.86
C101 (100)	827.88	1235.94	1524.59	1234.40	1673.82	1800.85	1537.64	1803.56	1806.58

Table 19: Average Demand Served by Hybrid Rollout with Dynamic Decomposition

Duration	short			medium			long		
Capacity	small	medium	large	small	medium	large	small	medium	large
				Low Variability					
R101 (25)	150.71	220.42	277.09	231.82	318.90	331.69	300.88	332.04	332.04
C101 (25)	209.97	296.98	320.02	311.41	451.17	459.64	391.27	459.23	459.64
R101 (50)	351.08	492.56	625.36	509.52	710.39	721.15	650.54	721.14	721.15
C101 (50)	319.42	533.60	593.58	532.33	794.57	858.95	671.22	859.83	859.92
R101 (75)	584.39	795.13	978.01	826.11	1077.48	1078.42	1021.42	1078.42	1078.42
C101 (75)	750.58	1090.68	1291.73	1060.97	1359.14	1359.99	1286.41	1360.04	1360.05
R101 (100)	792.44	1105.70	1361.89	1115.07	1456.71	1458.60	1376.41	1458.60	1458.59
C101 (100)	878.25	1340.65	1636.46	1301.74	1798.40	1810.80	1638.26	1810.79	1810.84
				Moderate Variability					
R101 (25)	142.07	209.13	273.03	223.28	311.69	330.18	289.98	331.56	331.61
C101 (25)	208.62	288.86	318.56	297.86	420.43	456.56	381.28	458.25	459.28
R101 (50)	319.64	467.14	602.19	492.34	680.73	719.72	629.51	720.38	720.52
C101 (50)	319.72	499.99	581.72	516.41	738.50	846.99	655.35	854.57	858.61
R101 (75)	557.34	763.12	946.40	798.86	1060.11	1077.58	994.08	1077.58	1077.58
C101 (75)	724.24	1040.06	1267.57	1029.88	1340.91	1360.47	1257.51	1360.50	1360.55
R101 (100)	759.86	1064.81	1303.49	1082.38	1435.79	1457.27	1341.77	1457.37	1457.37
C101 (100)	856.57	1295.34	1595.89	1269.43	1748.48	1809.19	1588.95	1809.92	1809.97
				High Variability					
R101 (25)	141.80	202.23	260.24	215.45	298.42	325.77	278.77	331.35	331.73
C101 (25)	196.63	276.65	311.59	289.41	397.67	451.18	369.69	453.21	460.01
R101 (50)	304.38	454.02	566.80	475.23	654.16	711.49	609.31	716.18	716.93
C101 (50)	300.57	479.26	564.68	493.98	704.98	815.71	637.36	837.06	855.58
R101 (75)	523.82	738.55	915.75	772.87	1035.51	1075.57	967.67	1075.58	1075.64
C101 (75)	697.74	1006.92	1217.59	1010.41	1315.36	1360.00	1222.61	1360.55	1360.58
R101 (100)	724.68	1021.31	1258.87	1054.84	1404.25	1452.72	1308.61	1452.86	1452.86
C101 (100)	828.91	1234.82	1521.58	1234.33	1675.04	1800.94	1535.79	1803.91	1806.58

Table 20: Average CPU Seconds Per Instance

Method	25 Customers Avg. CPU	50 Customers Avg. CPU	75 Customers Avg. CPU	100 Customers Avg. CPU
		No Decomposition		
One-Step Rollout	19.01	2647.93	–	–
Post-Decision Rollout	5.48	579.78	–	–
Pre-Decision Rollout	0.63	5.54	56.93	480.30
Hybrid Rollout	1.32	36.81	1664.97	–
		Static Decomposition		
One-Step Rollout	1.52	5.15	10.79	33.06
Post-Decision Rollout	0.55	2.28	5.05	24.67
Pre-Decision Rollout	0.76	1.99	4.48	9.27
Hybrid Rollout	0.81	2.65	6.34	14.71
		Dynamic Decomposition		
One-Step Rollout	1.95	8.16	64.00	457.47
Post-Decision Rollout	0.89	7.42	61.72	466.23
Hybrid Rollout	0.87	7.12	58.38	485.14