Partial-outsourcing strategy for the vehicle routing problem with stochastic demands

Lin Zhu[a,*], Yossiri Adulyasak[b], Louis-Martin Rousseau[c]

[a]*Logistics Research Center, Shanghai Maritime University, Shanghai, China, 201306*
[b]*GERAD and HEC Montréal, Quebec, Canada, H3T 2A7*
[c]*CIRRELT and Polytechnique Montréal, Quebec, Canada, H3C 3A7*

Abstract

We present a partial-outsourcing strategy for the vehicle routing problem with stochastic demands (VRPSD). In our strategy, dynamic routing and replenishment decisions are made. Our strategy differs from classical recourse strategies by using outsourcing as recourse for handling unmet demand. We present a Markov decision process (MDP) model for VRPSD, and propose an approach based on an approximate linear programming (ALP) scheme to solve it. To efficiently tackle the curse of dimensionality, we propose several algorithmic enhancements that exploit the structure of the problem, including decomposition-based value function approximations, lower bounding procedures based on affine functions, and constraint sampling. We compare the performance of our approach against state-of-the-art approaches, and the results show that our approach generally yields high-quality solutions.

Keywords: stochastic vehicle routing, reoptimization, Markov decision process, approximate linear programming, outscourcing

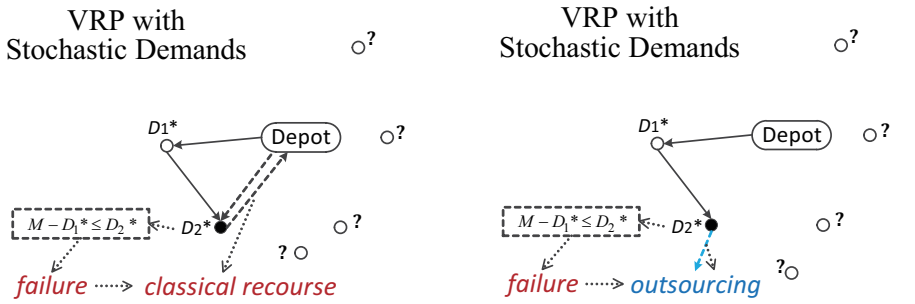
1. Introduction

The vehicle routing problem with stochastic demand (VRPSD) is an important member of the family of vehicle routing problems (Gendreau et al., 2016). The problem reflects the nature of many real-world distribution practices. Research on this problem can facilitate understanding of the nature of routing in a stochastic environment. The VRPSD, with its uncertainty in customer demand, poses a great challenge from a computational standpoint. In the VRPSD, vehicles are dispatched to visit customers in order to satisfy their demand, while also respecting vehicle capacity. Customer demands are random variables with known probability distributions. It is not always possible to avoid routing failure, and so associated recourse actions must sometimes be used. A failure occurs when the residual capacity of the vehicle is insufficient to satisfy customer demand. In these cases, recourse actions are taken to recover the feasibility of the route.

*Corresponding author
Email addresses: zhulin.sife@gmail.com (Lin Zhu), yossiri.adulyasak@hec.ca (Yossiri Adulyasak), louis-martin.rousseau@polymtl.ca (Louis-Martin Rousseau)

The classical recourse action requires the vehicle to make a detour to the depot for replenishment, and then resume its route following the failure. This is shown in Fig. 1(a). A recourse cost is incurred when the vehicle travels additional distance to fulfill unmet demand. The classical recourse action is simple to implement, but the relevant cost is high when it takes substantial effort to conduct the detour trip for a small amount of unmet demand. A better recourse action is to outsource delivery for unmet demand with an economically sound price.

The development of crowdsourcing deliveries also makes it easier for shipment operators to make outsourcing decisions in real-time. In particular, updated shipment information (such as completed orders, pending orders, and shipment costs) is immediately available. Thus, better recourse actions can be taken with the support given by crowdsourcing deliveries. This can save the time it takes to satisfy all customer demand, and also reduce the total cost. We propose a partial-outsourcing strategy for the VRPSD. Specifically, if a vehicle arrives at a customer and observes a failure, it meets the customer demand as much as possible, and leaves the unmet part to be outsourced to other carriers from the crowdsourcing platform. Fig. 1(b) demonstrates the recourse action in the partial-outsourcing strategy, which defers from the classical recourse action shown in Fig. 1(a).



(a) classical recourse action (b) recourse action in partial-outsourcing strategy

Figure 1: Comparison of recourse actions in partial-outsourcing to traditional strategies

A similar practice can be observed in the research by Kafle et al. (2017). They studied a joint delivery problem with self-owned vehicles and carriers. However, our research is different due to the demand in our problem being stochastic. We are the first to introduce an outsourcing strategy for the VRPSD. More specifically, we consider the VRPSD with a single vehicle, in the same sprit as (Toriello et al., 2014; Secomandi, 2000, 2001). In our partial-outsourcing strategy, a reoptimization method is used, in which dynamic routing is enabled along-sides dynamic replenishment decisions. Due to the high complexity when realizing dynamic routing, determining a reoptimization policy remains highly challenging, and research in this area is still very limited (e.g., Psaraftis, 1995; Secomandi and Margot, 2009).

We develop an approximated linear programming (ALP) method to compute our strategy. ALP methods have been utilized in the context of other problems, including inventory routing (Adelman, 2003, 2004), revenue management (Adelman, 2007; Kunnumkal and Topaloglu, 2010; Tong and Topaloglu, 2014), scheduling problems (Adelman and Barz, 2014; Barz and Rajaram, 2015), and the traveling salesman problem (Toriello et al., 2014). To our knowledge,

we are the first to adapt this solution methodology to tackle this variant of VRP with outsourcing. With an ALP method, the Markov decision process (MDP) is first transformed into its linear programming (LP) counterpart, in which the variables are value functions. For tractability, affine functions are then introduced to approximate the variables. The ALP method can yield the best possible bounds on the optimal value functions. The bounds are then used as the approximated value functions within traditional Bellman recursion to derive a price-directed policy (e.g., De Farias and Van Roy, 2003; Toriello et al., 2014).

Our contribution is generally threefold. First, we propose a partial-outsourcing strategy for the VRPSD. This recourse action is different from the classical one, because it leverages the crowdsourcing of deliveries. Second, we present an ALP method and generate a policy for the VRPSD. To enhance the performance of the proposed method, we exploit the structure of the problem and develop several algorithmic enhancements, namely decomposition-based value function approximations, lower bounding procedures based on affine functions, and constraint sampling. Third, we compare our approach with two competitive solution methods, a posteriori bound, and a method of fixing the routing sequence. The experimental results demonstrate that our approach yields high-quality solutions.

The remainder of the paper is organized as follows. Section 2 presents the partial-outsourcing strategy, which is formulated using an MDP formulation. Section 3 introduces our ALP solution framework. By exploring the problem structure, a decomposition-based solution framework is developed. A value function is decomposed into two parts and then Section 4 specifies the approximation for the second part. In Section 5, a price-directed policy is derived based on our approximation of value functions, in which routing and restocking decisions are elaborated given observed routing states. Section 6 discusses the experimental results.

2. Partial-outsourcing strategy

In this section, we formulate the partial-outsourcing strategy using MDP. We first present the notation and assumptions for the routing problem, then we define the value functions and optimal actions under the strategy with outsourcing. Finally, we indicate the difference between our formulation and the ones used in previous research, and generalize the optimality equation in our formulation.

Notation and assumptions

We first present the main notation used in this paper, which is generally in-line with other VRPSD literature (e.g., Bertsimas, 1992; Secomandi, 2000; Novoa and Storer, 2009; Secomandi and Margot, 2009). The strategy can be formulated as a finite-horizon discrete-time Markov decision process. Considering a complete network, customers are denoted by node set $\mathcal{N} = \{1, \ldots, N\}$, and 0 denotes the depot. A vehicle with capacity Q ($Q \in \mathbb{N}^+$) is dispatched from a depot to visit customers, satisfy their demands, and eventually return to the depot. Distance d_{lj} between any two nodes l and j ($l,\ j \in \mathcal{N} \cup 0$) is assumed to satisfy the triangle inequality. Demand quantity for customer l ($l \in \mathcal{N}$), $\tilde{\xi}_l$, is a random variable characterized by a probability distribution $p_l(e) = \Pr(\tilde{\xi}_l = e)$ ($e = 0, 1, \ldots, E \leq Q$) and

$p_l(e) = 0$ $(e = E + 1, \ldots, Q)$, where E is a nonnegative integer. Customer demand $\tilde{\xi}_l$ is independent of the vehicle routing/replenishment policy, and its realization ξ_l can only be observed when the vehicle arrives at the customer. It is assumed that the total depot capacity is sufficient fully service all customers.

In our formulation, split deliveries are allowed. When a failure occurs at customer l, the vehicle delivers its existing load q $(q < \xi_l)$ to the customer, and the remaining unmet demand is outsourced with an expense of $b \cdot (\xi_l - q)$, where b is the unit price for outsourcing.

Value functions

We formulate the strategy as an MDP with stages in set $\Omega = \{N, N-1, \ldots, 0\}$, with stage $k \in \Omega$ corresponding to the number of unvisited customers. Each stage $k \in \Omega\backslash\{N\}$ starts when the vehicle finishes serving the current customer. The corresponding state is denoted by $s_k = (l, q, \mathcal{R}_k(l))$, representing the vehicle departing from current location l $(l \in \mathcal{N})$ with available capacity q $(q \in \mathcal{Q} = \{0, 1, \ldots, Q\})$ and set of remaining unvisited customers $\mathcal{R}_k(l)$ $(\mathcal{R}_k(l) \subseteq \mathcal{N})$. Ψ denotes the state space for the process, and it is composed of

$$\Psi = \{s_N = (0, Q, \mathcal{N})\} \cup \{s_k = (l, q, \mathcal{R}_k(l)) \,|\, k \in \Omega\backslash\{N\},\ l \in \mathcal{N},\ q \in \mathcal{Q},\ \mathcal{R}_k(l) \subset \mathcal{N}\}. \tag{1}$$

For state $s_k = (l, q, \mathcal{R}_k(l))$ at stage $k \in \Omega\backslash\{N, 0\}$, two decisions must be made. First, it must be decided which customer $j \in \mathcal{R}_k(l)$ to visit next. Second, it must be decided whether the vehicle will go directly to that customer (a case labeled $D(j)$), or return to the depot to restock prior to proceeding to that customer (a case labeled $R(j)$). At the beginning stage N, the only available decision is which customer to visit first. For the final stage 0, the only available action is to return to the depot without replenishing.

Let $V_k(l, q, \mathcal{R}_k(l))$ denote the optimal expected cost-to-go from state $s_k = (l, q, \mathcal{R}_k(l)) \in \Psi$. The cost-to-go values at the final stage are

$$V_0(l, q, \phi) = d_{l0}, \qquad \forall l \in \mathcal{N},\ q \in \mathcal{Q}. \tag{2}$$

For state $s_k = (l, q, \mathcal{R}_k(l))$ at stage $k \in \Omega\backslash\{N, 0\}$, the optimal policy satisfies the following Bellman equations,

$$V_k(l, q, \mathcal{R}_k(l)) = \min_{j \in \mathcal{R}_k(l)} \left\{ \min \left\{ V_k^{D(j)}(l, q, \mathcal{R}_k(l)), V_k^{R(j)}(l, q, \mathcal{R}_k(l)) \right\} \right\}, \quad \forall s_k \in \Psi. \tag{3}$$

where $V_k^{D(j)}(l, q, \mathcal{R}_k(l))$ and $V_k^{R(j)}(l, q, \mathcal{R}_k(l))$ are the cost-to-go values associated with stage k and state $(l, q, \mathcal{R}_k(l))$, corresponding to visiting the next customer j directly and by first replenishing at the depot, respectively. $\min\{V_k^{D(j)}(l, q, \mathcal{R}_k(l)), V_k^{R(j)}(l, q, \mathcal{R}_k(l))\}$ ensures customer j $(j \in \mathcal{R}_k(l))$ is reached in the most efficient way. The optimal next customer j^* $(j^* \in \mathcal{R}_k(l))$ corresponds to the one with the minimum cost-to-go value. The cost-to-go values for the two cases can

be written as follows

$$
\begin{aligned}
V_k^{D(j)}(l,q,\mathcal{R}_k(l)) &= d_{lj} + B_j(q) + \sum_{e\le q} p_j(e)\cdot V_{k-1}(j,q-e,\mathcal{R}_{k-1}(j;l)) + V_{k-1}(j,0,\mathcal{R}_{k-1}(j;l))\cdot\sum_{e>q} p_j(e),\\
V_k^{R(j)}(l,q,\mathcal{R}_k(l)) &= d_{l0} + d_{0j} + \sum_{e} p_j(e)\cdot V_{k-1}(j,Q-e,\mathcal{R}_{k-1}(j;l)), \qquad \forall\, j\in\mathcal{R}_k(l),\ s_k\in\Psi,
\end{aligned}
\tag{4}
$$

where $\mathcal{R}_{k-1}(j;l) = \mathcal{R}_k(l)\backslash\{j\}$, and $B_j(q) = b\cdot\sum\limits_{e>q} p_j(e)\cdot(e-q)$ calculates the expected outsourcing cost if residual capacity q is not sufficient to meet customer j's demand (equaling to 0 when $q\ge E$).

At beginning stage N, for unique starting state $s_N = (0,Q,\mathcal{N})$, the optimal value function is

$$
V_N(0,Q,\mathcal{N}) = \min_{j\in\mathcal{N}}\left\{d_{0j} + \sum_{e} p_j(e)\cdot V_{N-1}(j,Q-e,\mathcal{N}\backslash\{j\})\right\}. \tag{5}
$$

Optimal actions

The optimal action at final stage 0 is simply to return to the depot from the final customer, whereas beginning stage N includes a choice of which customer j to visit in such a way that

$$
j_N(0,Q,\mathcal{N}) = \arg\min_{j\in\mathcal{N}}\left\{d_{0j} + \sum_{e} p_j(e)\cdot V_{N-1}(j,Q-e,\mathcal{N}\backslash\{j\})\right\}. \tag{6}
$$

We introduce variable $u_{j,l,\mathcal{R}_k(l)}(q)$ to denote the replenishment decision at state $(l,q,\mathcal{R}_k(l))$ with respect to routing customer j next. The optimal replenish decision $u_{j,l,\mathcal{R}_k(l)}(q)$ is determined by

$$
u_{j,l,\mathcal{R}_k(l)}(q) = \begin{cases} 1, & \text{if } V_k^{R(j)}(l,q,\mathcal{R}_k(l)) \le V_k^{D(j)}(l,q,\mathcal{R}_k(l)) \\ 0, & \text{if } V_k^{R(j)}(l,q,\mathcal{R}_k(l)) > V_k^{D(j)}(l,q,\mathcal{R}_k(l)). \end{cases} \qquad \forall\, j\in\mathcal{R}_k(l),\ \text{for } s_k\in\Psi. \tag{7}
$$

By defining $u_{j,l,\mathcal{R}_k(l)}(q)$, we can rewrite equation (4) as

$$
\begin{aligned}
V_k^{u_{j,l,\mathcal{R}_k(l)}(q)}(l,q,\mathcal{R}_k(l)) &= d_{lj} + B_j(q) + (\Delta_{lj} - B_j(q))\cdot u_{j,l,\mathcal{R}_k(l)}(q)\\
&\quad + \mathbb{E}\left[V_{k-1}(j,q',\mathcal{R}_{k-1}(j;l))\,\middle|\, q,\ u_{j,l,\mathcal{R}_k(l)}(q)\right]. \quad \forall\, j\in\mathcal{R}_k(l),\ s_k\in\Psi.
\end{aligned}
\tag{8}
$$

where $\Delta_{lj} = d_{l0} + d_{0j} - d_{lj}$ represents the extra cost for a preventive return to the depot, when l and j are consecutive customers in the delivery route. $\mathbb{E}\left[V_{k-1}(j,q',\mathcal{R}_{k-1}(j;l))\,\middle|\, q,\ u_{j,l,\mathcal{R}_k(l)}(q)\right]$ is the expected future cost, given initial residual capacity q and taking the replenish decision given by equation (7)

$$\mathbb{E}\left[V_{k-1}(j,q',\mathcal{R}_{k-1}(j;l))\,\middle|\,q,\ u_{j,l,\mathcal{R}_k(l)}(q)\right] =$$
$$\begin{cases} \sum\limits_{e\le q} p_j(e)\cdot V_{k-1}(j,q-e,\mathcal{R}_{k-1}(j;l)) + V_{k-1}(j,0,\mathcal{R}_{k-1}(j;l))\cdot \sum\limits_{e>q} p_j(e), & \text{if } u_{j,l,\mathcal{R}_k(l)}(q)=0, \\ \sum\limits_{e} p_j(e)\cdot V_{k-1}(j,Q-e,\mathcal{R}_{k-1}(j;l)), & \text{if } u_{j,l,\mathcal{R}_k(l)}(q)=1. \end{cases} \tag{9}$$

Based on the definition of $V_k^{u_{j,l,\mathcal{R}_k(l)}(q)}(l,q,\mathcal{R}_k(l))$, for state $s_k=(l,q,\mathcal{R}_k(l))$ at stage $k\in\Omega\backslash\{N,0\}$, the best next customer location is determined by

$$j_k(l,q,\mathcal{R}_k(l)) = \arg\min_{j\in\mathcal{R}_k(l)}\left\{V_k^{u_{j,l,\mathcal{R}_k(l)}(q)}(l,q,\mathcal{R}_k(l))\right\}. \tag{10}$$

Comparison of dynamic programming equations for VRPSD with outsourcing, and traditional VRPSD

In traditional VRPSD (see, e.g., Yang et al., 2000; Secomandi, 2001; Secomandi and Margot, 2009), the cost-to-go value for each state s_k can be expressed as

$$V_k(l,q,\mathcal{R}_k(l)) = \min\left\{V_k^{D(j_k^D(s_k))}(l,q,\mathcal{R}_k(l)),\ V_k^{R(j_k^R(s_k))}(l,q,\mathcal{R}_k(l))\right\},\quad \forall\, s_k\in\Psi, \tag{11}$$

where $j_k^D(s_k)$ and $j_k^R(s_k)$ are the best next customer locations for the case of proceeding to the next customer directly and the case of first replenishing at the depot, respectively. In their formulations, the best routing options are considered first, then replenishment decisions are made. This stands in contrast to equation (3), where replenishment decisions are made first, after which the best routing option is decided. Note that we change the decision sequence (i.e., $u\to j$, instead of $j\to u$) for ease of formulation for our approximation scheme.

Our partial-outsourcing strategy is formulated as in Section 2. Correspondingly, optimality equations (3) and (4) can be jointly expressed as

$$V_k(l,q,\mathcal{R}_k(l)) = \min_{j\in\mathcal{R}_k(l)}\left\{d_{lj} + B_j(q) + (\Delta_{lj}-B_j(q))\cdot u_{j,l,\mathcal{R}_k(l)}(q) + \mathbb{E}\left[V_{k-1}(j,q',\mathcal{R}_{k-1}(j;l))\,\middle|\,q,\ u_{j,l,\mathcal{R}_k(l)}(q)\right]\right\} \tag{12}$$
$$\forall\, j\in\mathcal{R}_k(l),\ s_k\in\Psi.$$

We describe our ALP solution framework to solve (12) in the subsequent section.

3. Decomposition-based approximate linear program framework

Solving the MDP formulation directly will inevitably lead to the curse of dimensionality. In our formulation, the cardinality of the state space is $1+N(Q+1)2^{N-1}$, which is intractable as the the number of customers grows. One efficient approach to solve this problem is the approximate linear programming framework. This approach is built upon the linear programming (LP) formulation of the MDP and replaces the value functions with approximate

(and possibly affine) functions, defining the lower bounds of the true value functions (for our case, a minimization problem). We formally rewrite our MDP model (2)-(5) as an LP as follows.

$$\max \quad V_N(0, Q, \mathcal{N}) \tag{13.1}$$

$$\text{s.t.} \quad V_N(0, Q, \mathcal{N}) \leq d_{0j} + \sum_e p_j(e) \cdot V_{N-1}(j, Q-e, \mathcal{N}\backslash\{j\}), \quad j \in \mathcal{N}, \tag{13.2}$$

$$V_k(l, q, \mathcal{R}_k(l)) \leq d_{lj} + B_j(q) + \sum_{e \leq q} p_j(e) \cdot V_{k-1}(j, q-e, \mathcal{R}_{k-1}(j; l)) + V_{k-1}(j, 0, \mathcal{R}_{k-1}(j; l)) \cdot \sum_{e>q} p_j(e),$$

$$V_k(l, q, \mathcal{R}_k(l)) \leq d_{l0} + d_{0j} + \sum_e p_j(e) \cdot V_{k-1}(j, Q-e, \mathcal{R}_{k-1}(j; l)),$$

$$\forall\, k \in \Omega\backslash\{N, 0\},\ \forall\, l \in \mathcal{N},\ q \in \mathcal{Q},\ \mathcal{R}_{k-1}(j; l) \subseteq \mathcal{N}\backslash\{l, j\},\ j \in \mathcal{R}_k(l), \tag{13.3}$$

$$V_0(l, q, \phi) \leq d_{l0}, \quad \forall\, l \in \mathcal{N},\ q \in \mathcal{Q}. \tag{13.4}$$

One potential approach to reduce the complexity of the model is to leverage affine functions to approximate value functions. Toriello et al. (2014) studied a traveling salesman problem with stochastic arc costs and identified the challenge of applying ALP in a routing problem. They shared the insight that the approximation of value functions can be poor if directly replaced by simple affine functions, and pointed out that strong affine functions can be derived based on exploring the structure of the routing problem first (Toriello et al., 2014).

In the subsequent section, we analyze the structure of the value functions in our formulation and propose an approximation reformulation scheme to efficiently solve it.

Decomposition-based value function approximation

Reoptimization and a priori optimization are two major approaches for VRPSD (Gendreau et al., 2016; Zhu et al., 2014). A priori optimization is often modeled by stochastic programming with recourse (SPR). In fact, a basic structural characteristic is revealed in the objective function of the SPR formulations, as the objective function is normally composed of two parts (e.g., Laporte and Louveaux, 1993; Louveaux and Salazar-González, 2018; Salavati-Khoshghalb et al., 2019)

Similarly, value functions within our problem can also be decomposed into two parts

$$V_k(l, q, \mathcal{R}_k(l)) = \min_{[J, (j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)]} \left\{ \left[d_{lJ} + v_{k-1}(J, \mathcal{R}_{k-1}(J; l)) \Big|_{(j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)} \right] + f^J_k(l, q, \mathcal{R}_k(l)) \Big|_{(j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)} \right\}, \quad J \in \mathcal{R}_k(l), \quad \forall\, s_k \in \Psi. \tag{14}$$

The first component $[d_{lJ} + v_{k-1}(J, \mathcal{R}_{k-1}(J; l)) \Big|_{(j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)}]$ is relevant to the travel cost between customers, given routing sequence $[l, J, (j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)]$. The realized route starts from location l and includes remaining customers in $\mathcal{R}_k(l)$, by visiting customer $J \in \mathcal{R}_k(l)$ first and customers $j^J_{k'}$ ($j^J_{k'} \in \mathcal{R}_{k-1}(J; l)\backslash\{j^J_{k-1}, \ldots, j^J_{k'+1}\}$) at subsequent stages $k' = \{k-1, k-2, \ldots, 1\}$. $v_{k-1}(J, \mathcal{R}_{k-1}(J; l)) \Big|_{(j^J_{k-1}, j^J_{k-2}, \ldots, j^J_1)}$ refers to the travel cost related to the partial route starting from customer

J and including the remaining customers in $\mathcal{R}_{k-1}(j;l)$. The second component $f_k^J(l,q,\mathcal{R}_k(l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}$ is related to the penalty cost, including any additional distance traveled for replenishments, as well as potential outsourcing cost. The second component also indicates the expected future penalty cost given current state $(l,q,\mathcal{R}_k(l))$ and the decision for routing customer J next. Both components rely on the future routing policy, represented by $[J,(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)]$, where superscript J in $j^J_{k'}$ ($k'=\{k-1,k-2,\ldots,1\}$) indicates that future routing policy should be made by fixing J as the next customer to be visited. Equation (14) aims at determining the optimal routing policy $[J,(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)]$ for state s_k, to minimize expected future costs.

We derive the decomposition-based value function approximation from equation (14). An original value function is approximated by decomposing it into two components and then estimating each component separately. In fact, the following relation holds for the second component in equation (14)

$$f_k^J(l,q,\mathcal{R}_k(l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)} \geq \min_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}\left\{f_k^J(l,q,\mathcal{R}_k(l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}\right\} = \min\left\{f_k^J(l,q,\mathcal{R}_k(l))\right\}, \quad \forall\, s_k\in\Psi,\ J\in\mathcal{R}_k(l), \tag{15}$$

where the right-hand side of the equation is obtained as term $\min\left\{f_k^J(l,q,\mathcal{R}_k(l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}\right\}$ indicates the minimal penalty cost among all realized routing sequences. Let $L^J_{s_k}$ denote the lower bound of the future expected penalty cost for state s_k along with routing customer J next. Assuming that better lower bounds $L^J_{s_k}$ ($\forall\, s_k\in\Psi$, $J\in\mathcal{R}_k(l)$) are given, $L^J_{s_k}$ are then used to approximate the second component $f_k^J(l,q,\mathcal{R}_k(l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}$. Let $\tilde{V}_k(l,q,\mathcal{R}_k(l))$ ($s_k\in\Psi$) denote the approximated value functions. The value functions are approximated as

$$\tilde{V}_k(l,q,\mathcal{R}_k(l)) = \min_{[J,(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)]}\left\{d_{lJ}+v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}+L^J_{s_k}\right\} \tag{16.1}$$

$$= \min_{J\in\mathcal{R}_k(l)}\left\{d_{lJ}+\min_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}\left\{v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\Big|_{(j^J_{k-1},j^J_{k-2},\ldots,j^J_1)}\right\}+L^J_{s_k}\right\} \tag{16.2}$$

$$= \min_{J\in\mathcal{R}_k(l)}\left\{d_{lJ}+\min\left\{v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\right\}+L^J_{s_k}\right\}, \quad \forall\, s_k\in\Psi, \tag{16.3}$$

where equation (16.3) holds for the same reason as in equation (15). In fact, $\min\{v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\}$ implies a deterministic TSP process. Following the definition of $v_{k-1}(J,\mathcal{R}_{k-1}(J;l))$, $\min\{v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\}$ determines a route that includes remaining customers in $\mathcal{R}_{k-1}(J;l)$ and ends at the depot with the minimal traveling cost. For a deterministic TSP problem, tight lower bounds can be fast generated by extant methods (Grötschel and Holland, 1991; Crowder and Padberg, 1980). Let $l_{tsp}^{J,\mathcal{R}_{k-1}(J;l)}$ ($l\in\mathcal{N}$, $J\in\mathcal{R}_k(l)$, $|\mathcal{R}_k(l)|=k$) represent the lower bounds for approximating $\min\{v_{k-1}(J,\mathcal{R}_{k-1}(J;l))\}$. Value functions can be further estimated by

$$\tilde{V}_k(l,q,\mathcal{R}_k(l)) = \min_{J\in\mathcal{R}_k(l)}\left\{d_{lJ}+l_{tsp}^{J,\mathcal{R}_{k-1}(J;l)}+L^J_{s_k}\right\}, \quad \forall\, s_k\in\Psi. \tag{17}$$

In our method, value functions are approximated using two steps. In the first step, expected penalty costs $L^J_{s_k}$ are approximated for all possible states $s_k \in \Psi$ and all associated routing decisions $J \in \mathcal{R}_k(l)$. Essentially, a lookup table for penalty costs is available before any routing decisions are made. In the second step, the approximate travel costs for potential future TSP route $\left\{l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}\right\}$ are calculated along with the decision process ($k = N, N-1, \ldots, 1$), given realized state s_k and each potential routing decision $J \in \mathcal{R}_k(l)$. Based on the two steps, the optimal routing decision at state s_k is made according to

$$j_k(l, q, \mathcal{R}_k(l)) = \arg\min_{J \in \mathcal{R}_k(l)} \left\{ d_{lJ} + l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp} + L^J_{s_k} \right\}, \quad \text{given a realized state } s_k. \tag{18}$$

In other words, $L^J_{s_k}$ are calculated a priori, whereas $l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}$ are generated as needed. More specifically, at state $s_k = (l, q, \mathcal{R}_k(l))$, values $L^J_{s_k}$ ($J \in \mathcal{R}_k(l)$) are already available, so only $l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}$ ($J \in \mathcal{R}_k(l)$, $\mathcal{R}_{k-1}(J;l) = \mathcal{R}_k(l)\backslash\{J\}$) need to be computed. The following LP formulation can be utilized to generate $l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}$ for each routing decision $J \in \mathcal{R}_k(l)$, and this LP is solvable in polynomial time (Crowder and Padberg, 1980; Padberg and Rinaldi, 1990; Hao and Orlin, 1994).

$$l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp} = \min_x \left\{ \sum_{i' \in \mathcal{R}_{k-1}(J;l)} d_{Ji'} x_{Ji'} + \sum_{i' \in \mathcal{R}_{k-1}(J;l)} \sum_{j' \in \mathcal{R}_{k-1}(J;l) \cup \{0\} \backslash i'} d_{i'j'} x_{i'j'} \right\} \tag{19.1}$$

$$\text{s.t.} \sum_{i' \in \mathcal{R}_{k-1}(J;l)} x_{Ji'} = 1, \tag{19.2}$$

$$\sum_{i' \in \mathcal{R}_{k-1}(J;l)} x_{i'0} = 1, \tag{19.3}$$

$$\sum_{j' \in \mathcal{R}_{k-1}(J;l) \backslash i'} x_{i'j'} = 1, \quad i' \in \mathcal{R}_{k-1}(J;l), \tag{19.4}$$

$$\sum_{j' \in \mathcal{R}_{k-1}(J;l) \backslash i'} x_{j'i'} = 1, \quad i' \in \mathcal{R}_{k-1}(J;l), \tag{19.5}$$

$$\sum_{i' \in U} \sum_{j' \in \mathcal{R}_{k-1}(J;l) \cup \{0\} \backslash U} x_{i'j'} \geq 1, \quad \forall\, \phi \neq U \subseteq \mathcal{R}_{k-1}(J;l) \cup \{0\}, \tag{19.6}$$

$$x \geq 0, \quad x \in \mathbb{R}. \tag{19.7}$$

The LP formulation (19) can be used to determine lower bounds $l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}$ ($J \in \mathcal{R}_k(l)$). In the next section, we focus on generating the lookup table for expected penalty costs, i.e., $\left\{L^J_{s_k} \mid \forall\, s_k \in \Psi,\ J \in \mathcal{R}_k(l)\right\}$.

4. Lower bound of the expected penalty

We approximate the expected cost-to-go value for each state s_k ($s_k \in \Psi$) according to (17). d_{lJ} is the distance between current location l and the potential next destination $J \in \mathcal{R}_k(l)$. $l^{J,\mathcal{R}_{k-1}(J;l)}_{tsp}$ is the lower bound of the corresponding TSP problem, which can be obtained from formulation (19). Therefore, to approximate the cost-to-go value, only the lower bound of expected penalty cost $L^J_{s_k}$ is required.

In Section 4.1, the optimality equation for expected penalty costs is determined, then the MDP formulation is re-formulated using the linear program counterpart. In Section 4.2, by defining the affine functions, the ALP formulation for approximating penalty costs is obtained. Finally, we propose a method to solve the ALP formulation in Section 4.3.

4.1. Expected penalty cost

$L^J_{s_k}$ ($\forall\, s_k = (l, q, \mathcal{R}_k(l)) \in \Psi,\ J \in \mathcal{R}_k(l)$) are the lower bounds of expected penalty costs $f^J_k(l, q, \mathcal{R}_k(l))$. In fact, the expected penalty cost of visiting customer J next for each state s_k satisfies the following optimality equation

$$
\begin{aligned}
&f^J_k(l, q, \mathcal{R}_k(l)) = \min \\
&\begin{cases}
\Delta_{lJ} + \sum\limits_{e} p_J(e) \cdot f_{k-1}(J, Q - e, \mathcal{R}_{k-1}(J; l)), & u_{J,l,\mathcal{R}_k(l)}(q) = 1, \\
\sum\limits_{e \le q} p_J(e) \cdot f_{k-1}(J, q - e, \mathcal{R}_{k-1}(J; l)) + b \cdot \sum\limits_{e > q} p_J(e) \cdot (e - q) + \sum\limits_{e > q} p_J(e) \cdot f_{k-1}(J, 0, \mathcal{R}_{k-1}(J; l)), & u_{J,l,\mathcal{R}_k(l)}(q) = 0,
\end{cases} \\
&\qquad\qquad \forall\, s_k = (l, q, \mathcal{R}_k(l)) \in \Psi,\ J \in \mathcal{N} \backslash l,\ k \in \Omega \backslash \{N, 0\}, \qquad (20)
\end{aligned}
$$

where replenishment decision $u_{J,l,\mathcal{R}_k(l)}(q)$ for each state s_k is made to minimize the future expected penalty costs, either with ($u_{J,l,\mathcal{R}_k(l)}(q) = 1$) or without ($u_{J,l,\mathcal{R}_k(l)}(q) = 0$) replenishing before arriving at customer J. If decision $u_{J,l,\mathcal{R}_k(l)}(q) = 1$ is made, the vehicle arrives at customer J with full capacity Q and with the expense of additional travel cost Δ_{lJ} ($\Delta_{lJ} = d_{l0} + d_{0J} - d_{lJ}$). Otherwise, decision $u_{J,l,\mathcal{R}_k(l)}(q) = 0$ dictates that the vehicle proceeds to customer J, with probability $\sum\limits_{e > q} p_J(e)$ of failing the service of customer J and leaving empty vehicle capacity.

Note that $f_{k-1}(J, q', \mathcal{R}_{k-1}(J; l))$ equals to $\min\limits_{j' \in \mathcal{R}_{k-1}(J;l)} \left\{ f^{j'}_{k-1}(J, q', \mathcal{R}_{k-1}(J; l)) \right\}$, indicating that $f_{k-1}(J, q', \mathcal{R}_{k-1}(J; l))$ is the minimal expected penalty cost among all next routing decisions j' in set $\mathcal{R}_{k-1}(j'; l)$. In the following, we substitute $f_k(l, q, \mathcal{R}_k(l))$ ($f^j_k(l, q, \mathcal{R}_k(l))$) with $f_{l,\mathcal{R}}(q)$ ($f^j_{l,\mathcal{R}}(q)$) for ease of notation.

Proposition 1. (Monotonicity of penalty cost on residual capacity q) *for given customer l, customer j, and unvisited set $\mathcal{R}$, penalty cost $f^j_{l,\mathcal{R}}(q)$ is non-increasing in residual capacity q.*

Proposition 2. (Possible threshold-type replenishment) *for particular customer l^*, customer j^* and unvisited set $\mathcal{R}^*$ (not all), the optimal choice between replenishing and moving directly to the next customer is of threshold type in residual capacity q.*

The proofs are shown in Appendix B.□

Equation (20) is expressed by the following LP formulation (21)

$$
\begin{aligned}
&\max \quad f_{0,\mathcal{N}}(Q) && (21.1) \\
&\text{s.t.} \quad f^j_{0,\mathcal{N}}(Q) \le \sum_{e} p_j(e) \cdot f_{j,\mathcal{N}\backslash j}(Q - e), \quad \forall\, j \in \mathcal{N}, && (21.2)
\end{aligned}
$$

$$f^j_{l,\mathcal{R}}(q) \le \Delta_{lj} + \sum_e p_j(e) \cdot f_{j,\mathcal{R}\setminus l}(Q-e), \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 2\}), \tag{21.3}$$

$$f^j_{l,\mathcal{R}}(q) \le \sum_{e\le q} p_j(e) \cdot f_{j,\mathcal{R}\setminus l}(q-e) + b \cdot \sum_{e>q} p_j(e) \cdot (e-q) + \sum_{e>q} p_j(e) \cdot f_{j,\mathcal{R}\setminus l}(0),$$
$$\forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 2\}), \tag{21.4}$$

$$f^j_{l,\{j\}}(q) \le \Delta_{lj}, \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{N}\setminus l, \mathcal{R} = \{j\},\ q \in Q^{fe}_1, \tag{21.5}$$

$$f^j_{l,\{j\}}(q) \le b \cdot \sum_{e>q} p_j(e) \cdot (e-q), \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{N}\setminus l,\ \mathcal{R} = \{j\},\ q \in Q^{fe}_1, \tag{21.6}$$

$$f^j_{l,\mathcal{R}}(q),\ f_{l,\mathcal{R}}(q),\ f^j_{0,\mathcal{N}}(Q),\ f_{0,\mathcal{N}}(Q) \in \mathbb{R}, \quad \forall\ l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 1\}). \tag{21.7}$$

Constraints (21.2) indicate that the vehicle departs from the depot at initial stage N, with full capacity Q, and only the option to proceed to the first customer directly. Constraints (21.3) and (21.4) translate equation (20) correspondingly, and capture the transition of expected penalty costs from stage $N-1$ to 2. Constraints (21.5) and (21.6) state the situation at stage 1 when any expected penalty cost at final stage 0 ($f_{l,\phi}(q)$, $\forall\, l \in \mathcal{N}$, $q \in Q^{fe}_0$) is zero. In addition, $q \in Q^{fe}_{|\mathcal{R}|}$ ($|\mathcal{R}| = N-1, \ldots, 1$) in constraints (21.3)-(21.6) indicates the feasible range of residual capacity q at each stage k ($k = |\mathcal{R}| = N-1, \ldots, 1$), and Q^{fe}_k ($Q^{fe}_{|\mathcal{R}|}$) is $[(Q-(N-k)\cdot E)^+,\ Q - e_{\min}]$ for each stage k ($e_{\min}$ is the minimal amount that demand ξ can take).

In LP formulation (21), the variables are $f^j_{l,\mathcal{R}}(q)$ and $f_{l,\mathcal{R}}(q)$ ($l \in \mathcal{N} \cup 0$, $j \in \{\mathcal{N} \cup 0\}\setminus l$, $\mathcal{R} \subseteq \{\mathcal{N} \cup 0\}\setminus l$, $q \in Q^{fe}_{|\mathcal{R}|}$). Under objective (21.1), the optimal solution $f^j_{l,\mathcal{R}}(q)^*$ ($f_{l,\mathcal{R}}(q)^*$) is the largest lower bound of each expected penalty cost $f^j_{l,\mathcal{R}}(q)$ ($f_{l,\mathcal{R}}(q)$).

4.2. Affine approximation for lower bound

Solving formulation (21) can be inefficient due to its large number of variables. Variables $f^j_{l,\mathcal{R}}(q)$ and $f_{l,\mathcal{R}}(q)$ amount to the scale of $O(N^2 \cdot 2^{N-1} \cdot Q)$ (calculated by $N^2 \cdot (\mathrm{C}^1_{N-1} + \mathrm{C}^2_{N-1} + , \ldots, + \mathrm{C}^{N-1}_{N-1}) \cdot (Q+1) + 1$). Thus, we apply a variable reduction method to reduce the dimensionality of the model and improve its scalability.

Variable reduction is firstly achieved by noting that $f_{l,\mathcal{R}}(q) = \min\limits_{j\in\mathcal{R}} \left\{ f^j_{l,\mathcal{R}}(q) \right\}$, meaning that constraints (22) are added in formulation (21).

$$f_{l,\mathcal{R}}(q) \le f^j_{l,\mathcal{R}}(q), \quad \forall\, j \in \mathcal{R} \subseteq N\setminus l \quad (l \in 0 \cup \mathcal{N},\ q \in Q^{fe}_k,\ k \in \{N,\ N-1,\ \ldots,\ 1\}) \tag{22}$$

We obtain formulation (23)

$$\max \quad f_{0,\mathcal{N}}(Q) \tag{23.1}$$

$$\text{s.t.} \quad f_{0,\mathcal{N}}(q) \le \sum_e p_j(e) \cdot f_{j,\mathcal{N}\setminus j}(Q-e), \quad \forall\, j \in \mathcal{N}, \tag{23.2}$$

$$f_{l,\mathcal{R}}(q) \le \Delta_{lj} + \sum_e p_j(e) \cdot f_{j,\mathcal{R}\setminus l}(Q-e), \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,$$

$$q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 2\}), \tag{23.3}$$

$$f_{l,\mathcal{R}}(q) \le \sum_{e \le q} p_j(e) \cdot f_{j,\mathcal{R}\setminus j}(q-e) + b \cdot \sum_{e>q} p_j(e) \cdot (e-q) + \sum_{e>q} p_j(e) \cdot f_{j,\mathcal{R}\setminus j}(0),$$

$$\forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 2\}), \tag{23.4}$$

$$f_{l,\{j\}}(q) \le \Delta_{lj}, \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{N}\setminus l,\ \mathcal{R} = \{j\},\ q \in Q^{fe}_{1}, \tag{23.5}$$

$$f_{l,\{j\}}(q) \le b \cdot \sum_{e>q} p_j(e) \cdot (e-q), \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{N}\setminus l, \mathcal{R} = \{j\},\ q \in Q^{fe}_{1}, \tag{23.6}$$

$$f_{l,\mathcal{R}}(q),\ f_{0,\mathcal{N}}(Q) \in \mathbb{R}, \quad \forall\ l \in \mathcal{N},\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 1\}). \tag{23.7}$$

The number of variables is decreased, with only variables $f_{l,\mathcal{R}}(q)$ included in the formulation. Variable size can be further reduced by identifying a set of bases and substituting their affine form for the original variables in the formulation. We derive the affine function forms as in (24.1)-(24.3) to approximate the expected penalty costs. The affine functions are tailored to our problem structure, which are adapted from the forms in extant research (e.g., in Toriello et al., 2014; Tong and Topaloglu, 2014).

$$f_{0,\mathcal{N}}(Q) \approx \theta_{0,Q,0}, \tag{24.1}$$

$$f_{l,\mathcal{R}}(q) \approx \theta_{l,q,0} + \sum_{j \in \mathcal{R}} (\alpha_{l,q,j} \cdot \Delta_{l,j} + \omega_{l,j} \cdot q), \quad l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ |\mathcal{R}| \ge 2,\ q \in Q^{fe}_{|\mathcal{R}|}, \tag{24.2}$$

$$f_{l,\{j\}}(q) \approx \theta_{l,q,0} + \alpha_{l,q,j} \cdot \Delta_{lj} + \omega_{l,j} \cdot q, \quad l \in \mathcal{N},\ j \in \mathcal{N}\setminus l,\ \mathcal{R} = \{j\}. \tag{24.3}$$

where $\theta \in \mathbb{R}^{N \cdot Q+N+1}$, $\alpha \in \mathbb{R}^{(N^2-N)\cdot(Q+1)}$ and $\omega \in \mathbb{R}^{N^2-N}$. Affine functions (24.1)-(24.3) approximate expected penalty cost $f_{l,\mathcal{R}}(q)$. Firstly, penalty costs occur because restocking actions are taken or outsourcing occurs, so terms $\alpha_{l,q,j} \cdot \Delta_{l,j}$ and $\omega_{l,j} \cdot q$ are introduced, respectively. $\alpha_{l,q,j} \cdot \Delta_{lj}$ implies that restocking contributes to additional travel cost Δ_{lj} ($j \in \mathcal{R} \subseteq \mathcal{N}\setminus l$), and $w_{l,j} \cdot q$ indicates that the outsourcing cost is relevant to available residual capacity q, whereas constant θ adjusts the approximation of the penalty costs. Secondly, expected penalty costs are determined based on the states ($\forall\ s_k = (l, q, \mathcal{R})$), so parameters θ, α and ω are defined in relation to the states. Note that ω is only relevant to current location l and remaining unvisited customer j ($j \in \mathcal{R}$), considering that residual capacity q is already reflected as a multiplier factor in term $\omega_{l,j} \cdot q$. In fact, term $\omega \cdot q$ reflects the monotonicity of penalty costs given residual capacity q (the proof can be found in Appendix A). Lastly, (24.2) reflects the penalty cost attributed to the different possibilities for being the next visited customer $j \in \mathcal{R}$, when more than one customer is included in the remaining unvisited set ($|\mathcal{R}| \ge 2$). With this approximation (24.1)-(24.3), formulation (23) becomes

$$\max \quad \theta_{0,0,Q} \tag{25.1}$$

$$\text{s.t.} \quad \theta_{0,0,Q} \le \sum_{e} p_j(e) \cdot \theta_{j,0,Q-e} + \sum_{t \in \mathcal{N}\setminus j} \Delta_{jt} \cdot (\sum_{e} p_j(e) \cdot \alpha_{j,t,Q-e}) + \sum_{t \in \mathcal{N}\setminus j} \omega_{jt} \cdot (\sum_{e} p_j(e) \cdot (Q-e)),$$

$$\forall\, j \in \mathcal{N}, \tag{25.2}$$

$$\theta_{l,0,q} + \sum_{t\in\mathcal{R}}(\Delta_{lt}\cdot\alpha_{l,t,q} + \omega_{lt}\cdot q) \le \Delta_{lj} + \sum_{e} p_j(e)\cdot\theta_{j,0,Q-e} + \sum_{t\in\mathcal{R}\setminus j}\Delta_{jt}\cdot(\sum_{e} p_j(e)\cdot\alpha_{j,t,Q-e}) + \sum_{t\in\mathcal{R}\setminus j}\omega_{jt}\cdot(\sum_{e} p_j(e)\cdot(Q-e)),$$

$$\forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2,\ \ldots,\ 2\}), \tag{25.3}$$

$$\theta_{l,0,q} + \sum_{t\in\mathcal{R}}(\Delta_{lt}\cdot\alpha_{l,t,q} + \omega_{lt}\cdot q) \le (\sum_{e\le q} p_j(e)\cdot\theta_{j,0,q-e} + \sum_{e>q} p_j(e)\cdot\theta_{j,0,0}) + b\cdot\sum_{e>q} p_j(e)\cdot(e-q) + \sum_{t\in\mathcal{R}\setminus j}\Delta_{jt}\cdot(\sum_{e\le q} p_j(e)\cdot\alpha_{j,t,Q-e} + \sum_{e>q} p_j(e)\cdot\alpha_{j,t,0}) + \sum_{t\in\mathcal{R}\setminus j}\omega_{jt}\cdot(\sum_{e\le q} p_j(e)\cdot(q-e))$$

$$\forall\, l \in \mathcal{N},\ j \in \mathcal{R} \subseteq \mathcal{N}\setminus l,\ q \in Q^{fe}_{|\mathcal{R}|}\ (|\mathcal{R}| \in \{N-1,\ N-2\ \ldots, 2\}), \tag{25.4}$$

$$\theta_{l,0,q} + \Delta_{lj}\cdot\alpha_{l,j,q} + \omega_{lj}\cdot q \le \Delta_{lj}, \quad \forall\, l \in \mathcal{N},\ j \in \mathcal{N}\setminus l,\ q \in Q^{fe}_{1}, \tag{25.5}$$

$$\theta_{l,0,q} + \Delta_{lj}\cdot\alpha_{l,j,q} + \omega_{lj}\cdot q \le b\cdot\sum_{e>q} p_j(e)\cdot(e-q), \quad \forall l\,\mathcal{N},\ j \in \mathcal{N}\setminus l,\ q \in Q^{fe}_{1}, \tag{25.6}$$

$$\theta,\ \alpha,\ \omega \in \mathbb{R}. \tag{25.7}$$

In formulation (25), variables include θ, α and ω, reaching the scale of $O(N^2\cdot Q)$ (calculated by $N^2\cdot(Q+1)+N^2-N+1$). Clearly, the variable size is dramatically reduced with respect to its original size in formulation (23.1)-(23.7). In the following subsection, we introduce a method to solve ALP formulation (25).

4.3. Constraint sampling

Formulation (25) reduces variables to a manageable number. However, the number of constraints is still potentially too large to solve. We employ a constraint sampling approach (De Farias and Van Roy, 2004) to tackle this problem. Constraint sampling is a general method used to tackle LP formulations with few variables and an intractable number of constraints. It approximates the solution to the ALP. The method selects promising constraints and obtains a solution based on the reduced formulation. Specifically, a promising constraint set is formed based on selected state-action pairs, and each pair is obtained by sample learning from an optimal policy. Only a subset of constraints are included in the formulation; this is possible because some constraints are inactive or have a minor impact on the feasible region (De Farias and Van Roy, 2004).

Our method is developed based on the general framework for constraint sampling. The general method relies on the existence of an optimal policy, which is normally unknown (as in routing problems with stochastic demand). We propose a multi-policy sampling framework to mimic the optimal policy. Based on the constraints sampled by a set of heuristic policies, the constraint space relevant to the ideal policy is mimicked. The local optimum obtained by a single heuristic policy can be escaped by exploring a larger solution space discovered via policy diversification.

In our multi-policy sampling framework, we begin by preparing a set of heuristic policies (we list the candidate policies that we select in Appendix A). Each policy (or heuristic algorithm) is applied to learn each sample, whereby

state-action pairs are generated. Specifically, for sample $\{\xi\}^{sam}$, if applying policy pl, a sequence of states and actions is obtained in the form

$$\left(s_N,\, a_{s_N|pl};\; s_{N-1},\, a_{s_{N-1}|pl};\ldots;\; s_0,\, a_{s_0|pl} \right)^{\{\xi\}^{sam}},$$

where $\{\xi\}^{sam}$ denotes a sample of realized customer demand $\{\xi_l \,|\, l \in \mathcal{N}\}$. $a_{s_{k|pl}} = (j, u_{j,l,\mathcal{R}}(q))$ specifies the outcome of applying distinct heuristic policy pl, potentially indicating a different routing decision j and restocking decision $u_{j,l,\mathcal{R}}(q))$ to be taken given realized state s_k. State $s_k = (l, q, \mathcal{R})$ transitions to state $s_{k-1} = (j, [q + (Q - q) \cdot u_{j,l,\mathcal{R}}(q) - \xi_j]^+, \mathcal{R}\backslash j)$ depending on action $a_{s_{k|pl}} = (j, u_{j,l,\mathcal{R}}(q))$ and realized demand ξ_j, where $[\cdot]^+$ indicates non-negative residual capacity. Therefore, as each heuristic policy learns each sample, a set of states s and associated actions $a_{s|pl}$ are obtained. The sequence can be written as a set of pairs of states and actions $(s, a_{s|pl})$ $(s = \{s_N, s_{N-1}, \ldots, s_0\})$. The pool of state-action pairs is finally formed by combining all sets of state-action pairs obtained during the learning, denoted as $\mathrm{P_{s-a}} = \sum_{\substack{pl\in \mathrm{Pls} \\ s' \leftarrow s, a_{s_k|pl}}} (s, a_{s|pl})$.

In fact, each state-action pair determines a group of constraints. For example, state-action pair $(s_k, a_{s_k}) = (l, q, \mathcal{R}, j, u_{j,l,\mathcal{R}}(q))$ (if $|\mathcal{R}| \geq 2, \mathcal{R} \neq \mathcal{N}$) implies that constraints (26) are selected.

$$\begin{cases} \theta_{l,0,q} + \sum\limits_{t\in\mathcal{R}} (\Delta_{lt} \cdot \alpha_{l,t,q} + \omega_{lt} \cdot q) \leq \Delta_{lj} + \sum\limits_{e} p_j(e) \cdot \theta_{j,0,Q-e} + \sum\limits_{t\in\mathcal{R}\backslash j} \Delta_{jt} \cdot (\sum\limits_{e} p_j(e) \cdot \alpha_{j,t,Q-e}) + \\ \qquad\qquad \sum\limits_{t\in\mathcal{R}\backslash j} \omega_{jt} \cdot (\sum\limits_{e} p_j(e) \cdot (Q - e)), \qquad u_{j,l,\mathcal{R}(q)} = 1, \\ \theta_{l,0,q} + \sum\limits_{t\in\mathcal{R}} (\Delta_{lt} \cdot \alpha_{l,t,q} + \omega_{lt} \cdot q) \leq (\sum\limits_{e\leq q} p_j(e) \cdot \theta_{j,0,q-e} + \sum\limits_{e>q} p_j(e) \cdot \theta_{j,0,0}) + b \cdot \sum\limits_{e>q} p_j(e) \cdot (e - q) + \\ \qquad \sum\limits_{t\in\mathcal{R}\backslash j} \Delta_{jt} \cdot (\sum\limits_{e\leq q} p_j(e) \cdot \alpha_{j,t,Q-e} + \sum\limits_{e>q} p_j(e) \cdot \alpha_{j,t,0}) + \sum\limits_{t\in\mathcal{R}\backslash j} \omega_{jt} \cdot (\sum\limits_{e\leq q} p_j(e) \cdot (q - e)), \qquad u_{j,l,\mathcal{R}}(q) = 0, \end{cases}$$

$$q \in Q^{fe}_{|\mathcal{R}|},\ 2 \leq |\mathcal{R}| \leq N \quad (26)$$

Constraints (26) imply that visiting customer j next is regarded as a promising action when the vehicle is located at customer l and given unvisited customer set $\mathcal{R}$ $(j \in \mathcal{R})$. Note that all feasible residual capacity $q \in Q^{fe}_{|\mathcal{R}|}$ is included, and whether the restocking action is taken $(u_{j,l,\mathcal{R}} = 1)$ or not $(u_{j,l,\mathcal{R}} = 0)$ is taken into account. All feasible residual capacities $q \in Q^{fe}_{|\mathcal{R}|}$ are considered, since some residual capacities may not be observed on sampling. Also, constraints indicating with and without restocking are included, to reveal the threshold-type restocking nature is given different realized residual capacities. Similarly, when the state-action pair is $(l, q, \{j\}, j, u_{j,l,\{j\}}(q))$ $(l \in \mathcal{N},\ j \in \mathcal{N}\backslash l, \mathcal{R} = \{j\})$, the selection of constraints is discussed identically, indicated by (27).

$$\begin{cases} \theta_{l,0,q} + \Delta_{lj} \cdot \alpha_{l,j,q} + \omega_{lj} \cdot q \leq \Delta_{lj}, & u_{j,l,\mathcal{R}}(q) = 1, \\ \theta_{l,0,q} + \Delta_{lj} \cdot \alpha_{l,j,q} + \omega_{lj} \cdot q \leq b \cdot \sum\limits_{e>q} p_j(e) \cdot (e - q), & u_{j,l,\mathcal{R}}(q) = 0, \end{cases} \qquad q \in Q^{fe}_1 \quad (27)$$

As for a state-action pair $(0,\ q \equiv Q,\ j,\ u_{j,0,\mathcal{N}}(q) \equiv 0)$ $(j \in \mathcal{N})$ at beginning stage N, the selected constraint is naturally the same form as in (25.2). Therefore, the promising constraint set is formed. All promising constraints are selected

based on promising state-action pairs $(s_k, a_{s_k|m}) \in \mathrm{P_{s-a}}$, according to (26), (27) and (25.2).

By solving ALP formulation (25) with the constraints sampled by state-action pairs $(s_k, a_{s_k|m}) \in \mathrm{P_{s-a}}$, the values of parameters θ, α and ω are approximated. The lower bounds of expected penalty costs $L^J_{s_k}$ ($J \in \mathcal{R}$, $s_k = (l, q, \mathcal{R}) \in \Psi$) are subsequently obtained based on (28.1)-(28.3).

$$L^J_{s_N} \approx \sum_e p_J(e) \cdot \theta_{J,0,Q-e} + \sum_{t \in \mathcal{N}\backslash J} \Delta_{Jt} \cdot (\sum_e p_J(e) \cdot \alpha_{J,t,Q-e}) + \sum_{t \in \mathcal{N}\backslash J} \omega_{Jt} \cdot (\sum_e p_J(e) \cdot (Q-e)),$$
$$s_N = (0, Q, \mathcal{N}),\ j \in \mathcal{N}, \tag{28.1}$$

$$L^J_{s_k} = \min\left\{L^{R(J)}_{s_k}, L^{D(J)}_{s_k}\right\} \approx$$
$$\min\left\{\begin{array}{ll} \Delta_{lJ} + \sum\limits_e p_J(e) \cdot \theta_{J,0,Q-e} + \sum\limits_{t \in \mathcal{R}\backslash j} \Delta_{Jt} \cdot (\sum\limits_e p_J(e) \cdot \alpha_{J,t,Q-e}) + \sum\limits_{t \in \mathcal{R}\backslash J} \omega_{Jt} \cdot (\sum\limits_e p_J(e) \cdot (Q-e)), & u_{J,l,\mathcal{R}}(q) = 1, \\ (\sum\limits_{e \le q} p_J(e) \cdot \theta_{J,0,q-e} + \sum\limits_{e > q} p_J(e) \cdot \theta_{J,0,0}) + \sum\limits_{t \in \mathcal{R}\backslash J} \Delta_{Jk} \cdot (\sum\limits_{e \le q} p_J(e) \cdot \alpha_{J,t,Q-e} + \sum\limits_{e > q} \cdot p_J(e) \cdot \alpha_{J,t,0}) + & \\ \qquad \sum\limits_{t \in \mathcal{R}\backslash J} \omega_{Jt} \cdot (\sum\limits_{e \le q} p_J(e) \cdot (q-e)) + b \cdot \sum\limits_{e > q} p_J(e) \cdot (e-q), & u_{J,l,\mathcal{R}}(q) = 0, \end{array}\right.$$
$$\forall\ s_k = (l, q, \mathcal{R}) \in \Psi,\ J \in \mathcal{N}\backslash l,\ l \in \mathcal{N},\ q \in Q^{fe}_k,\ 2 \le k \le N-1, \tag{28.2}$$

$$L^J_{s_1} \approx \min\left\{\Delta_{lJ}, b \cdot \sum_{e > q} p_J(e) \cdot (e-q)\right\}, \quad \forall\ s_0 = (l, q, \{J\}) \in \Psi,\ J \in \mathcal{N}\backslash l,\ l \in \mathcal{N}, q \in Q^{fe}_1. \tag{28.3}$$

(28.1)-(28.3) approximates lower bounds for each state s_k ($s_k \in \Psi$, $s_k \in \mathrm{P_{s-a}}$) along with each routing choice J ($J \in \mathcal{R}$). (28.1)-(28.3) is derived based on (24.1)-(24.3) and (20). Values θ, α and ω are substituted into (24.1)-(24.3), to obtain a lower bound for each $f_{l,\mathcal{R}}(q)$ ($\forall\ l \in \mathcal{N}$, $\mathcal{R} \subseteq \mathcal{N}\backslash l$, $q \in Q^{fe}_{|\mathcal{R}|}$, $s \in \mathrm{P_{s-a}}$), since any feasible solution of ALP formulation (25) can derive a lower bound of the corresponding value function. Then, the lower bound of $f^J_{l,\mathcal{R}}(q)$, $L^J_{s_k}$, is obtained by substituting the lower bound of $f_{l,\mathcal{R}}(q)$ into (20). Finally, expected cost-to-go values $V_k(l, q, \mathcal{R})$ ($s_k \in \Psi$, $s_k \in \mathrm{P_{s-a}}$) are approximated, by substituting $L^J_{s_k}$ into (17).

5. Price-directed policy

ALP methods can lead to price-directed policies. In fact, any feasible solution of an ALP formulation generates lower bounds of value functions (for a minimum problem). The proof can be found in relevant articles (e.g., in Toriello et al., 2014; Adelman, 2007). In a price-directed policy, lower bounds are utilized to approximate the value functions, then to guide decision-making with respect to each realized state.

We develop an ALP method for approximating expected penalty costs (recourse costs). Based on (17), lower bounds for the value functions are generated. Our price-directed policy determines next routing location $j_k(l, q, \mathcal{R})$ and makes restocking decision $u_{j,l,\mathcal{R}}(q)$ for state $s_k = (l, q, \mathcal{R})$ ($s_k \in \Psi$, $k \in \Omega\backslash\{N, 0\}$, $s_k \in \mathrm{P_{s-a}}$) based on (29.1)-(29.2).

$$u_{J,l,\mathcal{R}}(q) = \left\{\begin{array}{ll} 1, & \text{if } L^{R(J)}_{s_k} \le L^{D(J)}_{s_k} \\ 0, & \text{if } L^{D(J)}_{s_k} > L^{R(J)}_{s_k} \end{array}\right., \quad \forall\ J \in \mathcal{R},\ J \in \mathrm{P_{s-a}} \tag{29.1}$$

$$j_k(l,q,\mathcal{R}) = \arg\min_{J\in\mathcal{R},J\in \mathrm{P_{s-a}}} \left\{d_{lJ} + l_{tsp}^{J,\mathcal{R}\setminus J} + L_{s_k}^{J}\right\}, \quad \text{where } L_{s_k}^{J} = \min\left\{L_{s_k}^{D(J)}, L_{s_k}^{R(J)}\right\} \tag{29.2}$$

For state $s_k = (l, q, \mathcal{R})$ at stage $k \in \{N-1, \ldots, 1\}$, (29.1) is applied first to make the restocking decision for each routing choice $J \in \mathcal{R}$. Then, the optimal routing option $j_k(l, q, \mathcal{R})$ is selected based on (29.2), and restocking decision $u_{j,l,\mathcal{R}}(q)$ is made accordingly based on routing choice $j_k(l, q, \mathcal{R})$. $L_{s_k}^{R(J)}$ and $L_{s_k}^{D(J)}$ are calculated as in (28.2) or (28.3) (respectively, for different stages), when restocking occurs ($u_{J,l,\mathcal{R}}(q) = 1$) at customer J or not ($u_{J,l,\mathcal{R}}(q) = 0$). Additionally, (30) determines routing decision ($j_N(0, Q, \mathcal{N})$) at the beginning, when the vehicle departs from depot 0 with full capacity Q and goes directly to the first customer (i.e., $u_{j,0,\mathcal{N}}(Q) \equiv 0$), where $L_{s_N}^{J}$ is obtained from (28.1).

$$j_N(0,Q,\mathcal{N}) = \arg\min_{J\in\mathcal{N},J\in \mathrm{P_{s-a}}} \left\{d_{0J} + l_{tsp}^{J,\mathcal{N}\setminus J} + L_{s_N}^{J}\right\} \tag{30}$$

Lastly, at final stage 0, the vehicle proceeds directly to the depot (i.e., $j_0(l, q, \phi) = 0$ and $u_{0,l,\phi}(q) = 0$).

Note that observed states s and routing decisions j are restricted by the promising state-action space $\mathrm{P_{s-a}}$, for the tractability of the corresponding ALP formulation. Theoretically, our price-directed policy can infinitely approach the optimal policy if the state-action space $\mathrm{P_{s-a}}$ is well-selected.

6. Computational study

In this paper, we propose a partial-outsourcing strategy and develop an ALP method to compute it. The ALP method is used to approximate value functions which are then used within the Bellman equation to derive a price-directed policy (PD). We demonstrate the solution quality of our PD policy using a computationally effective a posteriori bound. We also compare our method with an approach where the routing sequence is fixed. We use the experimental results to evaluate the solution quality and computational cost of our method

Instance generation

We generate instances following the instance generation procedures used in Gendreau et al. (1995). Customer locations are randomly generated in a 1,000 by 1,000 square. The depot is located at coordinates (0, 0) or (500, 500), labeled as *corner* and *midpoint*, respectively. Customer demand corresponds to a discrete uniform random variable with support $\{1, \ldots, 5\}$. The mean demand of any customer in any instance is 3. The problem size ranges from 10 to 40 customers, in increments of 5. We only consider instances with 40 customers or fewer because off the computational difficulty of solving larger ones, an experience shared by Toriello et al. (2014). The expected filling rate is determined by $\bar{f} = \sum_{l=1}^{N} E(\tilde{\xi}_l)/Q$, and takes 1.9, 2.5 or 3.4 to show different failure frequencies. Vehicle capacity Q is computed by rounding $3N/\bar{f}$ to the nearest integer.

Outsourcing price b takes value $\bar{\Delta}_{lj} \Big/ \sum_e (p_j(e) \cdot e)$, using restocking to prevent the worst case, a failure. $\bar{\Delta}_{lj}$ denotes the average cost of a restocking. The worst case, a failure, occurs when the vehicle arrives at a customer with empty residual capacity; in this case the outsourcing cost is $b \cdot \sum_{e>q} p_j(e) \cdot (e - q) = b \cdot \sum_e p_j(e) \cdot e$. Price b reveals a

threshold. If a higher price is incurred, restocking may save costs. Otherwise, the depot is remote, and it is better to outsource. $\bar{\Delta}_{lj}$ is the average restocking cost between consecutive customers on a feasible route. We use a TSP route for convenience. Also, price b can fluctuate to reflect the outsourcing market. We generate 20 instances for each combination of settings.

Settings of PD policy

The PD policy is developed based on the multi-policy sampling framework. In practice, we make a tiny adjustment to elicit better performance. Specifically, an action is taken only if it can be obtained from the state-action pool. Namely, the next customer is only selected from the customers who are regarded as promising by the policy set. We also arbitrarily adjust the composition of policies in the set. For each instance, we choose the best combination of polices to find the solution of our PD policy. The candidate policies in our framework are described in Appendix A. In our implementation, the states and relevant actions are generated by implementing each candidate policy to learn on each sample. A sample is a set of realized customer demands ($\{\xi\}^{sam}$, as defined in Section 4.3). To determine state-action pool $\mathrm{P_{s-a}}$, 1000 samples are generated for learning by each candidate policy.

Solution quality of PD policy

We compare performance in terms of solution quality and solution time against the benchmark approaches in the literature, namely partial reoptimization (PR) (Secomandi and Margot, 2009) and the rollout algorithm (RA) (Secomandi, 2003), which are adapted to generate solutions for the VRPSD with outsourcing. More specifically, we adapt PR to realize partial-outsourcing and generate an effective posteriori bound (denoted as B^{PR}) for evaluating solution quality. In our experiment, PH(10) (a method to realize PR, refers to Secomandi and Margot (2009)) is implemented to generate the posteriori bound. We also compare our method with RA (Secomandi, 2003). RA can generate a static customer visiting sequence. We adapt RA to follow the partial-outsourcing scheme. The comparison with RA demonstrates the benefit of routing dynamically.

Two aspects of the solution quality obtained by our policy are evaluated. First, relative gap ε^{PR} is used to evaluate the difference between the results obtained by our PD policy and the posteriori bound ($\varepsilon^{PR} = \frac{V_N^{PD}(0,Q,\mathcal{N})-B^{PR}}{B^{PR}}$). Second, improvement rate γ^{RA} is adopted to reveal the improvement percentage of policies PD vs. RA ($\gamma^{RA} = \frac{V_N^{PD}(0,Q,\mathcal{N})-V_N^{RA}(0,Q,\mathcal{N})}{V_N^{RA}(0,Q,\mathcal{N})}$). By implementing policies PD, PR and RA, the actual costs are observed for each problem setting. We have 20 instances for each problem setting. The results in each line of Tables 1 and 2 reveal the averages over the 20 instances in a set. Tables 1 and 2 indicate the cases where the depot is located at the midpoint and in the corner, respectively.

Our method demonstrates good performance. The solution quality is demonstrated using posterior bound B^{PR}. We observe a smaller gap, of less than 10%, for our ALP approach versus the posterior bound B^{PR}. A similar finding was made by Toriello et al. (2014) in their study of a TSP problem. They reported that their ALP approach was within a 20% gap of the posteriori bound similarly produced as bound B^{PR} in our analysis. Additionally, our policy (PD)

Table 1: Total costs based on different approaches (midpoint depot)

problem setting No.	customers & capacity (N,Q)	our method (PD)	posteriori bound B^{PR}	fixed route (RA)	relative gap ε^{PR}	improvement rate γ^{RA}
# 1	(10, 16)	4124.4	4125.6	4396.2	-0.03%	-6.18%
3	(15, 24)	4709.3	4305.1	4415.1	9.39%	6.66%
5	(20, 24)	4346.8	4008.2	4485.4	8.45%	-3.09%
7	(25, 30)	5785.1	5506.1	5641.1	5.07%	2.55%
9	(30, 36)	5490.4	5394.4	5539.1	1.78%	-0.88%
11	(35, 31)	6118.2	5654.7	6118.2	8.19%	0%
13	(40, 35)	6133.5	5925.9	6320.9	3.50%	-2.96%

Table 2: Total costs based on different approaches (corner depot)

problem setting No.	customers & capacity (N,Q)	our method (PD)	posteriori bound B^{PR}	fixed route (RA)	relative gap ε^{PR}	improvement rate γ^{RA}
2	(10, 16)	6557.9	6160.4	6835.5	6.45%	-4.06%
4	(15, 24)	6771.4	6537.1	6714.7	3.58%	0.84%
# 6	(20, 24)	7207.6	7321.4	8177.7	-1.55%	-11.86%
8	(25, 30)	8131.2	7499.9	8717.7	8.42%	-6.73%
# 10	(30, 36)	8215.7	8389.5	9508.6	-2.07%	-13.60%
12	(35, 31)	10079.8	9601	9887.3	4.99%	1.95%
14	(40, 35)	11076.3	10085	11093	9.83%	-0.15%

generally outperforms the fixed routing policy (RA).

The results demonstrate the potential of our algorithm in solving VRPSD. Competitive performance of our policy versus PR is shown in problem settings 10, 6, and 1. Our algorithm is realized by incorporating a series of enhancement techniques, including decomposition-based value function approximations, lower bounding procedures based on affine functions, and constraint sampling. It should be noted that we use a simple and intuitive constraint sampling, called multi-policy sampling. This constraint sampling approach exploits a local solution space restricted by current heuristic policies, which essentially impedes the performance of our solution framework. A better version of constraint sampling is to exploit the entire space. The encouraging results from problem settings 10 and 6 make us optimistic about the potential of our solution framework for solving VRPSD.

The experiments were conducted on a personal computer with an Intel Core 3.0 GHz processor and 8 GB RAM, using Gurobi as the LP solver. We will now discuss the compute time of our algorithm when solving the problem with different settings.

Computational costs

Table 3: CPU times of different approaches in seconds (midpoint depot)

problem setting No.	Customers & Capacity (N,Q)	our method (PD)			posteriori bound B^{PR}	fixed route (RA)	time(PD)/time(B^{PR})
		prep.	imple.	total			
1	(10,16)	1652.8	27.3	1680.1	1618.5	3.1	103.81%
3	(15,24)	4487.2	108.6	4595.8	4485.1	5.1	102.47%
5	(20,24)	8212.1	164.9	8377	8205.8	18.7	102.09%
7	(25,30)	12838.5	1127.9	13966.4	12679.6	26.6	110.15%
9	(30,36)	36668.7	4577	41245.7	33989	48	121.35%
11	(35,31)	32324.3	1207.8	33532.1	32069.2	143.2	104.56%
13	(40,35)	37003.2	1792.9	38796.1	36967	183.6	104.95%

Table 4: CPU times of different approaches in seconds (corner depot)

problem setting No.	Customers & Capacity (N,Q)	our method (PD)			posteriori bound B^{PR}	fixed route (RA)	time(PD)/time(B^{PR})
		prep.	imple.	total			
2	(10,16)	1611.9	62.4	1674.3	1661.3	5.3	100.78%
4	(15,24)	4475.5	147.1	4622.6	4475.2	4	103.29%
6	(20,24)	8734.7	697.8	9432.5	8731.9	27.2	108.02%
8	(25,30)	12641.2	8057.7	20698.9	12458.5	34.5	166.14%
10	(30,36)	36348	4551.6	40899.6	35621.6	51.6	114.82%
12	(35,31)	23495.2	5700.7	29195.9	23450.3	161.3	124.50%
14	(40,35)	48579.8	8688.1	57267.9	48530.6	366.6	118%

The total time for PD policy is composed of two parts, i.e., the pre-compute time (indicated by prep.), and the time for implementation (indicated by imple.). The pre-compute time mainly consists of the time required to prepare constraints (i.e., the formation of the state-action pool), and the time required to choose the best subset of constraints (see Appendix A). As shown in Tables 3 and 4, it takes an almost equal amount of computational effort to obtain the solution by PD policy as it does to obtain the posteriori bound by PR policy.

Solution quality of PD policy in time limits

We further improve computational efficiency by limiting runtimes for solving the ALP formulation (25), to see if the performance of our algorithm noticeably degrades. We capture the performance of our PD policy at different runtimes (at 1 minute, 5 minutes, 10 minutes, 30 minutes, and 1 hour), and draw comparisons with posteriori bound B^{PR}. Table 5 shows the resulting gaps. As revealed in the table, our algorithm can yield good quality solutions within limited runtimes. The relative gaps are reduced along with the increased time limits, and achieve within 15% after a runtime of 30 minutes.

Table 5: Relative gaps ε^{PR} at different runtimes

No.	1 min	5 min	10 min	30 min	1 hour
1	-0.03%	-0.03%	-0.03%	-0.03%	-0.03%
2	6.72%	6.72%	6.72%	6.72%	6.72%
3	11.76%	11.76%	11.76%	11.76%	11.76%
4	7.27%	7.27%	7.27%	7.27%	7.27%
5	8.45%	8.45%	8.45%	8.45%	8.45%
6	8.37%	-0.77%	-0.77%	-0.77%	-0.77%
7	5.18%	4.27%	4.27%	4.27%	4.27%
8	16.76%	16.76%	16.76%	13.43%	13.43%
9	2.38%	2.38%	2.38%	2.38%	2.38%
10	9.95%	2.68%	2.68%	2.68%	2.68%
11	8.20%	8.20%	8.20%	8.20%	8.20%
12	/	4.99%	4.99%	4.99%	4.99%
13	3.12%	3.12%	3.12%	3.12%	3.12%
14	21.25%	21.25%	11.31%	9.83%	9.83%

In summary, we provide a new method for solving the multi-stage stochastic VRP with uncertain demand. Our approach results in good performance. We also identify a method of improving our approach by refining our constraint sampling technique. Future cooperation with experts in machine learning could facilitate our research towards achieving that goal.

7. Conclusions

In this paper, we study the VRPSD with a single vehicle. We propose a new recourse policy to handle unmet demand that outsources it to other carriers, and thus we develop the partial-outsourcing strategy. We formulate the strategy with an MDP formulation, and develop an ALP method to solve it. Lower bounds of value functions are generated based on the proposed decomposition-based ALP framework, which are then used to guide decision-making and obtain a price-directed policy.

We develop the multi-policy sampling framework to select constraints for our ALP formulation. Based on this simple version of constraint sampling, the sub-optimal solution spaces discovered by different heuristic policies are considered using an integrated approach. We provide experimental results showing that our approach generally yields competitive solutions to the VRPSD.

We suggest that future efforts in improving the constraint sampling procedure would obtain a better approach for VRPSD with single vehicle. The performance of our methodology can be improved if constraints can be selected by escaping the solution space identified by the leveraging heuristics. The efficiency of our methodology can similarly be further improved by realizing a better version of the constraint sampling procedure. We anticipate that these improvements will come to fruition by cooperating with experts in machine learning.

We consider a new trait in the distribution industry, introducing outsourcing in routing optimization. We call for more observations on new characteristics of the current distribution industry and we anticipate the creation of new categories of recourse policies as the VRP continues to evolve.

Acknowledgment

This work was supported by the China Scholarship Council under Grant 201908310007. This support is gratefully acknowledged.

Adelman, D., 2003. Price-directed replenishment of subsets: Methodology and its application to inventory routing. Manufacturing & Service Operations Management 5 (4), 348–371.

Adelman, D., 2004. A price-directed approach to stochastic inventory/routing. Operations Research 52 (4), 499–514.

Adelman, D., 2007. Dynamic bid prices in revenue management. Operations Research 55 (4), 647–661.

Adelman, D., Barz, C., 2014. A price-directed heuristic for the economic lot scheduling problem. IIE Transactions 46 (12), 1343–1356.

Barz, C., Rajaram, K., 2015. Elective patient admission and scheduling under multiple resource constraints. Production and Operations Management 24 (12), 1907–1930.

Bertsimas, D. J., 1992. A vehicle routing problem with stochastic demand. Operations Research 40 (3), 574–585.

Crowder, H., Padberg, M. W., 1980. Solving large-scale symmetric travelling salesman problems to optimality. Management Science 26 (5), 495–509.

De Farias, D. P., Van Roy, B., 2003. The linear programming approach to approximate dynamic programming. Operations research 51 (6), 850–865.

De Farias, D. P., Van Roy, B., 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. Mathematics of operations research 29 (3), 462–478.

Gendreau, M., Jabali, O., Rei, W., 2016. 50th anniversary invited articlełfuture research directions in stochastic vehicle routing. Transportation Science 50 (4), 1163–1173.

Gendreau, M., Laporte, G., Séguin, R., 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation science 29 (2), 143–155.

Grötschel, M., Holland, O., 1991. Solution of large-scale symmetric travelling salesman problems. Mathematical Programming 51 (1), 141–202.

Hao, J., Orlin, J. B., 1994. A faster algorithm for finding the minimum cut in a directed graph. Journal of Algorithms 17 (3), 424–446.

Kafle, N., Zou, B., Lin, J., 2017. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. Transportation research part B: methodological 99, 62–82.

Kunnumkal, S., Topaloglu, H., 2010. Computing time-dependent bid prices in network revenue management problems. Transportation Science 44 (1), 38–62.

Laporte, G., Louveaux, F. V., 1993. The integer l-shaped method for stochastic integer programs with complete recourse. Operations research letters 13 (3), 133–142.

Louveaux, F. V., Salazar-González, J.-J., 2018. Exact approach for the vehicle routing problem with stochastic demands and preventive returns. Transportation Science 52 (6), 1463–1478.

Novoa, C., Storer, R., 2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European journal of operational research 196 (2), 509–515.

Padberg, M., Rinaldi, G., 1990. An efficient algorithm for the minimum capacity cut problem. Mathematical Programming 47 (1), 19–36.

Psaraftis, H. N., 1995. Dynamic vehicle routing: Status and prospects. Annals of operations research 61 (1), 143–164.

Salavati-Khoshghalb, M., Gendreau, M., Jabali, O., Rei, W., 2019. A rule-based recourse for the vehicle routing problem with stochastic demands. Transportation Science 53 (5), 1334–1353.

Secomandi, N., 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers & 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., Margot, F., 2009. Reoptimization approaches for the vehicle-routing problem with stochastic demands. Operations Research 57 (1), 214–230.

Tong, C., Topaloglu, H., 2014. On the approximate linear programming approach for network revenue management problems. INFORMS Journal on Computing 26 (1), 121–134.

Toriello, A., Haskell, W. B., Poremba, M., 2014. A dynamic traveling salesman problem with stochastic arc costs. Operations Research 62 (5), 1107–1125.

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

Zhu, L., Louis-Martin, R., Rei, W., Li, B., 2014. Paired cooperative reoptimization strategy for the vehicle routing problem with stochastic demands. Computers & Operations Research 50, 1–13.

Appendix A. Policy set in multi-policy sampling framework

A set of policies is prepared to generate our price-directed policy. These policies determine the state-action pairs in the pool, and then determine the sampled constraints. Our policy set is composed of the following candidates: two reoptimization-type policies, two rollout-type polices, a category of a priori optimization policies, and a myopic policy.

In the candidate policies, the value functions are either computed originally as their methods indicate, or based on the decomposition framework as in (17). For example, partial reoptimization (Secomandi and Margot, 2009) is applied as one heuristic policy. Another reoptimization-type policy is generated by implementing the partial reoptimization framework in the formulation of penalty cost, and the value functions are then obtained based on (17). So, for each heuristic policy, based on different value function evaluations (i.e., with or without decomposition), an original policy and its variant based on decomposition are generated. We introduce partial reoptimization policy (Secomandi and Margot, 2009), rollout dynamic policy (Secomandi, 2001) and rollout static policy (Secomandi, 2003) in the policy set. The variants based on the decomposition framework are generated accordingly.

Among the candidates, some polices belong to the category of a priori optimization methods. The fixed routing sequence is implemented, and only restocking decisions are made during the execution of the policy. We diversify the generation of the a priori route using the rollout static method (Secomandi, 2003), a variant of the rollout static method (i.e., based on decomposition as illustrated above) and the TSP method (Crowder and Padberg, 1980; Padberg and Rinaldi, 1990; Hao and Orlin, 1994).

In addition, we also diversify the candidate choice by introducing a myopic policy. Under the myopic paradigm, routing and restocking decisions are made by only considering the immediate cost of the current state. For example, assume that current state is $s_k = (l, q, \mathcal{R})$. The restocking decision is first made for each potential routing choice, i.e., $u_{j,l,\mathcal{R}}(q) = 1$, if $d_{l0} + d_{0j} \leq d_{lj} + b \cdot \sum_{e>q} p_j(e) \cdot (e - q)$, otherwise, $u_{j,l,\mathcal{R}}(q) = 0$, and let $c_{ime}(j)$ denote the immediate cost if traveling to customer j ($\forall\, j \in \mathcal{R}$). Then, the routing decision is made based on $\arg\min_{j\in\mathcal{R}}\{c_{ime}(j)\}$, and the restocking decision is determined accordingly.

Overall, eight candidate heuristic policies are included in our setting, called par-reopt, par-reopt-de, rollout-dynamic, rollout-dynamic-de, rollout-static, rollout-static-de, TSP and myopic policies, where -de represents the

policy variant based on decomposition. In practice, some of them may be selected to form the policy set, depending on the performance of the resulting price-directed policy. Policy candidates can also be hand-selected. Different combinations of polices can be used to sample constraints, with the goal of obtaining a better price-directed policy.

Appendix B. Properties of penalty costs

Appendix B.1. Monotonicity of penalty cost on residual capacity q

Proof. Penalty cost $f^j_{l,\mathcal{R}}(q)$ is defined as in equation (20). Proving it is non-increasing in q is to testify $f^j_{l,\mathcal{R}}(q_2) \le f^j_{l,\mathcal{R}}(q_1)$ given $0 \le q_1 \le q_2 \le Q$. Four situations need to be considered, when $u_{j,l,\mathcal{R}}(q_1)$ and $u_{j,l,\mathcal{R}}(q_2)$ take different values ($u_{j,l,\mathcal{R}}(q) \in \{0, 1\}$).

1) If both $u_{j,l,\mathcal{R}}(q_1) = 1$ (case R) and $u_{j,l,\mathcal{R}}(q_2) = 1$ (case R), then, $f^{j(R)}_{l,\mathcal{R}}(q_2) = f^{j(R)}_{l,\mathcal{R}}(q_1)$;

2) If $u_{j,l,\mathcal{R}}(q_2) = 0$ (case D) and $u_{j,l,\mathcal{R}}(q_1) = 1$, then, $f^{j(D)}_{l,\mathcal{R}}(q_2) \le f^{j(R)}_{l,\mathcal{R}}(q_2) = f^{j(R)}_{l,\mathcal{R}}(q_1)$;

3) If both $u_{j,l,\mathcal{R}}(q_1) = 0$ and $u_{j,l,\mathcal{R}}(q_2) = 0$, then,

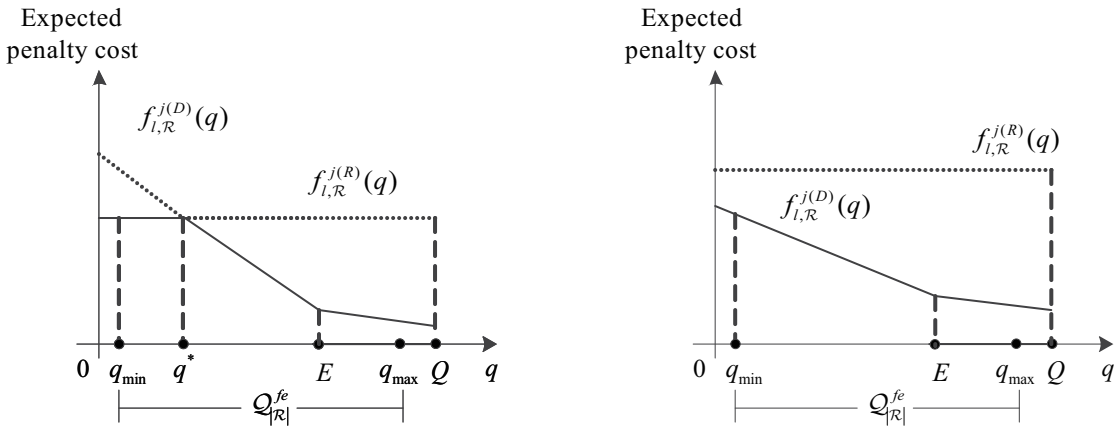
$$\begin{aligned}
f^{j(D)}_{l,\mathcal{R}}(q_2) - f^{j(D)}_{l,\mathcal{R}}(q_1) = &\sum_{e\le q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_2-e) + b\cdot\sum_{e>q_2} p_j(e)\cdot(e-q_2) + \sum_{e>q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(0) - \\
&\sum_{e\le q_1} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_1-e) - b\cdot\sum_{e>q_1} p_j(e)\cdot(e-q_1) - \sum_{e>q_1} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(0) \\
= &\, b\cdot\sum_{e>q_2} p_j(e)\cdot(q_1-q_2) - \sum_{e\le q_1} p_j(e)\cdot(f_{j,\mathcal{R}\setminus j}(q_1-e) - f_{j,\mathcal{R}\setminus j}(q_2-e)) - \\
&\, b\cdot\sum_{e>q_1}^{q_2} p_j(e)\cdot(e-q_1) - \sum_{e>q_1}^{q_2} p_j(e)\cdot(f_{j,\mathcal{R}\setminus j}(0) - f_{j,\mathcal{R}\setminus j}(q_2-e)) \\
\le &\, 0;
\end{aligned}$$

4) If $u_{j,l,\mathcal{R}}(q_2) = 1$ and $u_{j,l,\mathcal{R}}(q_1) = 0$, then,

$$\begin{aligned}
&f^{j(D)}_{l,\mathcal{R}}(q_1) \le f^{j(R)}_{l,\mathcal{R}}(q_1) = f^{j(R)}_{l,\mathcal{R}}(q_2), \text{ and} \\
&f^{j(D)}_{l,\mathcal{R}}(q_1) = \sum_{e\le q_1} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_1-e) + b\cdot\sum_{e>q_1} p_j(e)\cdot(e-q_1) + \sum_{e>q_1} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(0) \\
&\quad = \sum_{e\le q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_1-e) - \sum_{e>q_1}^{q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_1-e) + b\cdot\sum_{e>q_2} p_j(e)\cdot(e-q_1) + \\
&\qquad b\cdot\sum_{e>q_1}^{q_2} p_j(e)\cdot(e-q_1) + \sum_{e>q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(0) + \sum_{e>q_1}^{q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_2-e) \\
&\quad \ge b\cdot\sum_{e>q_2} p_j(e)\cdot(e-q_2) + \sum_{e>q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(0) + \sum_{e\le q_2} p_j(e)\cdot f_{j,\mathcal{R}\setminus j}(q_2-e) \\
&\quad \ge f^{j(R)}_{l,\mathcal{R}}(q_2) \\
&\therefore\ f^{j(D)}_{l,U}(q_1) = f^{j(R)}_{l,U}(q_2).\square
\end{aligned}$$

Appendix B.2. Possible threshold-type replenishment

For particular customer l^* and unvisited set $\mathcal{R}^*$ (not all), the optimal choice between replenishing and moving directly to the next customer is of threshold type in residual capacity $q \in \mathcal{Q}^{fe}_{|\mathcal{R}|}$.



(a) threshold-type (b) Non threshold-type

Figure B.1: Possible threshold-type replenishment

Because of the monotonicity of the expected penalty cost function, if $f^{j(D)}_{l,\mathcal{R}}(q_{\min}) > f^{j(R)}_{l,\mathcal{R}}(q_{\min})$, the decision for replenishing or proceeding to the next customer directly is of threshold-type in available capacity $q \in \mathcal{Q}^{fe}_{|\mathcal{R}|}$, as shown in Fig. B.1(a). ($q_{\min} \in \mathcal{Q}^{fe}_{|\mathcal{R}|}$ is the minimum feasible residual capacity when $|\mathcal{R}|$ number of customers remain unvisited.) Otherwise, the optimal decision is always to move directly to the next customer (case D) whatever the residual capacity q is, as shown in Fig. B.1(b).

Outsourcing price b appears to influence whether situation (a) or (b) happens. Visiting the next customer directly is always welcomed if the price is low enough. However, outsourcing is normally priced so that restocking is preferred in some circumstances and can be avoided in others.

We should note that the expected penalty cost function can be piece-wise linear because of the threshold type and also because the max customer demand is lower than vehicle capacity $E < Q$.