Contents lists available at ScienceDirect
European Journal of Operational Research
journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

Faster rollout search for the vehicle routing problem with stochastic demands and restocking

Luca Bertazzi[a,*], Nicola Secomandi[b]

[a] Department of Economics and Management, University of Brescia, Contrada Santa Chiara, 50, Brescia 25122, Italy
[b] Tepper School of Business, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 22 April 2017
Accepted 26 March 2018
Available online xxx

Keywords:
Routing
Rollout algorithms
Restocking
Stochastic vehicle routing problem

ABSTRACT

Rollout algorithms lead to effective heuristics for the single vehicle routing problem with stochastic demands (VRPSD), a prototypical model of logistics under uncertainty. However, they can be computationally intensive. To reduce their run time, we introduce a novel approach to approximate the expected cost of a route when executing any rollout algorithm for VRPSD with restocking. With a sufficiently large number of customers its theoretical speed-up factor is of big-o order 1/3. On a set of instances from the literature, our proposed technique applied to a known rollout algorithm and three variants thereof achieves speed-up factors that range from 0.26 to 0.34 when there are more than fifty customers, degrading only marginally the quality of the resulting routes. Our method also applies to the a priori case, in which case it is exact.

1. Introduction

Given a set of geographically dispersed customers, a quantity to deliver to each customer, and a fleet of capacitated vehicles located at a depot, the vehicle routing problem consists of determining a set of minimal cost routes, each starting and ending at the depot, such that the demand of all the customers is satisfied without exceeding the capacity of the vehicles. Since its introduction by Dantzig and Ramser (1959), this problem and variants thereof have been well studied (see Fisher, 1995; Laporte, 1992; Toth & Vigo, 2014; and Laporte, 2009 for reviews).

In the vehicle routing problem with stochastic demands (VRPSD), given probability distributions describe the customer demands and the realization of the demand of a customer becomes known upon the first visit to this customer. If the realized demand of a customer exceeds the remaining capacity of a vehicle when this customer is visited then a route failure occurs and a recourse action must be taken. VRPSD is relevant in both strategic distribution planning, when only estimates of customer demands are typically available, and tactical and operational decision making, when there remains residual uncertainty about the demands of the customers.

The extant literature includes three VRPSD solution strategies: a priori, restocking, and reoptimization (see Bertsimas & Simchi-Levi, 1996; Dror, 2002; Dror, Laporte, & Trudeau, 1989; Gendreau, Laporte, & Séguin, 1996a; Stewart & Golden, 1983; and Gendreau, Jabali, & Rei, 2016 for reviews). Under the a priori strategy, vehicles follow a given set of routes and back-and-forth replenishment trips to the depot are performed when a failure occurs (Bertsimas, 1992; Bertsimas, Chervi, & Peterson, 1995; Bertsimas, Jaillet, & Odoni, 1990; Gendreau, Laporte, & Séguin, 1995; 1996b; Goodson, Ohlmann, & Thomas, 2012; Gupta, Viswanath, & Ravi, 2012; Hjorring & Holt, 1999; Jabali, Rei, Gendreau, & Laporte, 2012; Laporte, Louveaux, & Van Hamme, 2002; Rei, Gendreau, & Soriano, 2010; Secomandi, 2003). The restocking strategy modifies the a priori approach by allowing preventing replenishment trips to the depot to avoid potentially costly route failures (Yee & Golden, 1980, Bertsimas et al., 1995; Secomandi, 2003; Yang, Mathur, & Ballou, 2000). With reoptimization, the decisions of which customer to visit next or whether to replenish depend on the demand observed and served so far (Goodson, Ohlmann, & Thomas, 2013; Goodson, Thomas, & Ohlmann, 2016; Novoa & Storer, 2009; Secomandi, 2000; 2001; Secomandi & Margot, 2009). In other words, as discussed by Secomandi and Margot (2009), both routing and replenishment decisions are static in the a priori case, routing decisions are static and replenishment decisions are dynamic in the restocking case, and both types of decisions are dynamic in the reoptimization case. Although dominated by reoptimization, the a priori and restocking strategies are appealing in practice because static routing creates regular service that is appreciated by both

* Corresponding author.
E-mail addresses: luca.bertazzi@unibs.it (L. Bertazzi), ns7@andrew.cmu.edu (N. Secomandi).

https://doi.org/10.1016/j.ejor.2018.03.034
0377-2217/© 2018 Elsevier B.V. All rights reserved.

customers and drivers (Bertsimas & Simchi-Levi, 1996). Further, the restocking approach outperforms the a priori approach (Yee and Golden 1980; Bertsimas et al., 1995; Secomandi, 2003) in terms of expected delivery cost. We thus focus our attention on the restocking strategy, but our methodological development is also relevant for the a priori case.

Heuristics are widespread in the VRPSD literature because exact methods can be used only on moderately-sized instances (Dror et al., 1989; Gendreau et al., 1995; Hjorring & Holt, 1999; Laporte et al., 2002; Bianchi et al., 2006; Jabali et al., 2012). Rollout search is an approximate dynamic programming approach that uses the expected cost of a known (base) heuristic to approximate the optimal value function when making a decision (Bertazzi, 2012; Bertsekas, 2005; Bertsekas & Castanon, 1999; Bertsekas & Tsitsiklis, 1996; Goodson, Thomas, & Ohlmann, 2017). Secomandi (2001, 2003), Novoa and Storer (2009) and Goodson et al. (2013, 2016) develop rollout methods for VRPSD.

The nested execution of a base heuristic, a defining element of rollout algorithms, can make rollout search computationally intensive, especially for instances that feature many customers. Thus, in this paper we develop a method to reduce the computationally requirement of this heuristic search approach applied to VRPSD under the restocking strategy when there is a single vehicle. In contrast to the standard backward dynamic programming evaluation of the expected cost of a restocking route (Yee and Golden 1980; Bertsimas et al., 1995; Secomandi, 2003; Yang et al., 2000), we first derive a novel forward dynamic programming approach to evaluate this cost. We then combine the forward and backward methods in a hybrid fashion when executing any rollout algorithm for the VRPSD version that we consider. This hybrid approach eliminates redundant computations at the expense of additional bookkeeping but approximates the expected cost of a restocking route. Even if it is outside the scope of our research, our proposed technique applies without approximation to the a priori strategy; that is, it yields an exact evaluation of the expected cost of an a priori route. The computational requirement of this method is theoretically appealing when applying VRPSD rollout algorithms to instances with a sufficiently large number of customers, in which case its big-o order speed-up factor equals 1/3.

We assess the performance of our proposed approach by applying it on the instances of Secomandi and Margot (2009) using as benchmarks the basic rollout algorithm of Secomandi (2003) for VRPSD with a single vehicle and three variants thereof. One of these variants bears some similarities with the rollout algorithms developed by Novoa and Storer (2009) to obtain a rollout policy for the reoptimization version of VRPSD. Hence, our considered rollout algorithms loosely represent extant single vehicle VRPSD rollout algorithms. Consistent with our theoretical analysis, across all the examined rollout algorithms, the observed speed-up factors vary between 0.26 and 0.34 for instances with at least 50 customers. These computational savings are associated with only a marginal degradation of the quality of the resulting routes. Although these results are specific to the rollout algorithms that are the subject of our numerical study, our expected cost computation approach is relevant to other VRPSD rollout algorithms one may design or use.

Other researchers have investigated the possibility of speeding up rollout search. In the context of the traveling salesman problem, Guerriero, Mancini, and Musmanno (2002) propose the pruned and relaxed rollout algorithms, which selectively execute the base heuristic at each iteration. Ciavotta, Meloni, and Pranzo (2016) develop related techniques for parallel machine scheduling problems. Guerriero and Mancini (2005) use parallel computing to reduce the run time of the traditional and pruned rollout algorithms applied to the traveling salesman and sequential ordering problems. In contrast, our approach always executes the base heuristic at each iteration, relies on sequential computing, and deals with a different application. The rollout policies of Novoa and Storer (2009) for VRPSD use a base heuristic of the a priori type (see also Birattari, Balaprakash, Stützle, & Dorigo, 2008 for the use of Monte Carlo simulation in local search for stochastic combinatorial optimization). These authors obtain improved computational efficiency by evaluating the expected cost of an a priori route using Monte Carlo simulation, without observing any deterioration in the obtained solutions even with small sample sizes. Monte Carlo evaluation cannot be applied to estimate the expected cost of a restocking route. However, in the a priori case one could combine Monte Carlo simulation and our proposed technique to obtain a possibly even faster method.

Bianchi et al. (2006) study the effectiveness of approximating the VRPSD objective function under the a priori strategy with the one of deterministic versions of this problem, which is easier to compute than the exact one, for various metaheuristics. Whereas their focus is on the solution quality of the resulting methods, our main attention is on the speed up of the approach that we put forth (although we observe that it has minimal impact on solution value). Moreover, our proposed method does not involve any approximation when applied to VRPSD under the a priori strategy.

We introduce VRPSD under the restocking strategy in Section 2. We present the backward and forward dynamic programming approaches to compute the expected cost of a VRPSD restocking route in Section 3. We discuss our proposed hybrid approach for the computation of the expected cost of such a route for any rollout algorithm in Section 4. We conduct our numerical study in Section 5. We conclude in Section 6. The online supplementary material includes additional information about our numerical results.

2. VRPSD with a single vehicle

We formulate VRPSD with a single vehicle as an optimization model. Let $G(V, E)$ be a given complete graph, where $V := \{0, 1, \dots, n\}$ is the set of $n+1$ nodes and E is the corresponding set of edges. The depot is located at node 0 and the customers at nodes 1 through n. The cost to travel from node i to node j is denoted by $c(i, j)$. Travel costs are symmetric, $c(i, j) = c(j, i)$ for each i and $j \in V$, and satisfy the triangle inequality, $c(i, j) \leq c(i, k) + c(k, j)$ for each i, j, and $k \in V$. A single and fully loaded vehicle with capacity Q is initially located at the depot. The demand of each customer i is the integer-valued random variable D_i. These random variables are independent. Their probability distributions are known. We denote the probability that the demand of customer i is equal to ℓ_i as $p_i(\ell_i)$, that is, $p_i(\ell_i) := \Pr\{D_i = \ell_i\}$. We let $\underline{L}_i \geq 1$, and $\overline{L}_i \leq Q$, respectively, be the minimal and maximal values of the support of the random variable D_i. The realization of the demand of a customer becomes known when, and only when, the vehicle arrives at the location of this customer.

A route is feasible if it starts and ends at the depot and visits all the customers exactly once. A route failure occurs when the vehicle does not have enough capacity to fully satisfy the demand of a customer. The VRPSD objective is to find a feasible route with minimal expected cost such that the demands of all the customers are satisfied. Formally, denote a feasible route (tour) by τ and the set of all feasible routes by $\mathcal{T}$. Let the expected cost of route τ under the restocking strategy be $\mathbb{E}[C_\tau]$. VRPSD is $\min_{\tau \in \mathcal{T}} \mathbb{E}[C_\tau]$.

3. Expected cost computation for a given route

In Section 3.1 we present a known backward recursion to compute the expected cost of a given route for the restocking case. In Section 3.2 we introduce a new forward recursion to compute this cost. In Section 3.3 we illustrate these recursions using a simple example. Given a feasible route τ, for expositional simplicity we renumber the customers of the given route so that it can be represented as $(0, 1, 2 \ldots, n, n+1 \equiv 0)$.

3.1. Backward recursion

The material in this section is in part based on Secomandi (2003) and references therein. Let $\gamma_i(q;\ \tau)$ be the expected cost of serving all customer demands when the vehicle follows route τ under the restocking strategy starting from customer i with q units of available capacity, after the demand of customer i has been satisfied. In dynamic programming terminology, $\gamma_i(\cdot\,;\ \tau)$ can be given the interpretation of cost-to-go function when the vehicle follows the residual part of route τ starting from customer i. With this notation, the quantity $\gamma_0(Q;\ \tau)$ corresponds to the expected cost of route τ, $\mathbb{E}[C_\tau]$. The function $\gamma_i(\cdot\,;\ \tau)$ can be computed by the backward recursion (1)–(3), where we suppress the argument τ for notational simplicity. This recursion, which is based on the two auxiliary functions $\gamma_i^{(1)}(\cdot)$ and $\gamma_i^{(2)}$, follows:

$$\gamma_n(q) = c(n,0), \qquad \forall q = 0,\ldots,Q-1, \tag{1}$$

$$\gamma_i(q) = \min\left\{\gamma_i^{(1)}(q), \gamma_i^{(2)}\right\}, \quad \forall i = n-1,\ldots,1, \quad q = 0,\ldots,Q-1, \tag{2}$$

$$\gamma_0(Q) = c(0,1) + \sum_{\ell_1=\underline{L}_1}^{\bar{L}_1} p_1(\ell_1)\gamma_1(Q-\ell_1), \tag{3}$$

with

$$\begin{aligned}
\gamma_i^{(1)}(q) &:= \sum_{\ell_{i+1}=\underline{L}_{i+1}}^{\min\{q,\bar{L}_{i+1}\}} p_{i+1}(\ell_{i+1})[c(i,i+1) + \gamma_{i+1}(q-\ell_{i+1})] \\
&\quad + \sum_{\ell_{i+1}=q+1}^{\bar{L}_{i+1}} p_{i+1}(\ell_{i+1})[c(i,i+1) \\
&\quad + 2c(i+1,0) + \gamma_{i+1}(q+Q-\ell_{i+1})], \\
\forall i &= n-1,\ldots,1, \quad q = 0,\ldots,Q-1, \\
\gamma_i^{(2)} &:= \sum_{\ell_{i+1}=\underline{L}_{i+1}}^{\bar{L}_{i+1}} p_{i+1}(\ell_{i+1})[c(i,0)+c(0,i+1)+\gamma_{i+1}(Q-\ell_{i+1})], \\
\forall i &= n-1,\ldots,1.
\end{aligned}$$

Expression (1) is a boundary condition that captures the deterministic cost to reach the depot from customer n, $c(n, 0)$, for all possible capacity levels after serving this customer demand. The recursive expression (2) sets the cost-to-go for all nodes from $n-1$ through 1 equal to the minimum between the cost-to-go of traveling directly to the next customer along route τ, $\gamma_i^{(1)}(q)$, which accounts for the possibility of a route failure and the corresponding roundtrip between this customer and the depot, and the cost-to-go of performing a replenishment trip to the depot on the way to this customer, $\gamma_i^{(2)}$. Finally, expression (3) includes the cost of traveling from the depot to the first customer in route τ and the cost-to-go from this customer with available capacity equal to $Q-\ell_1$.

Executing recursion (1)–(3) requires $O(nQ)$ space and $O(nLQ)$ time, where

$$L := 1 + \max_{i\in\{1,\ldots,n\}} \left\{\bar{L}_i - \underline{L}_i\right\}.$$

Moreover, the optimal restocking policy has a threshold structure in the available capacity (Yee & Golden, 1980; Secomandi, 1998; Yang et al., 2000): at each customer location along a given route, excluding the last one, it is optimal to restock provided that the available capacity is equal to or smaller than a critical value, which depends on this customer and this route. Formally, for each customer $i = 1, 2, \ldots, n-1$, because $\gamma_i^{(1)}(\cdot)$ is nonincreasing and $\gamma_i^{(2)}$ is constant, define $\hat{q}_i$ as the maximal value of q such that $\gamma_i^{(1)}(q) \geq \gamma_i^{(2)}$; the triangle inequality implies $\gamma_i^{(1)}(0) \geq \gamma_i^{(2)}$ so that $\hat{q}_i$ belongs to set $\{1, 2, \ldots, Q\}$. It is optimal to restock whenever the remaining vehicle capacity after serving the demand of customer i is less than or equal to $\hat{q}_i$. This property facilitates the execution of recursion (1)–(3), because the second argument in the minimization on the right-hand side of (2) is no larger than the first argument in this minimization when $\hat{q}_i$ exceeds q. We also exploit this property in Section 3.2.

3.2. Forward recursion

The recursion (1)–(3) computes the expected cost of a route under the restocking strategy in a backward fashion, that is, starting from the end of a route. In contrast, we now propose a recursion that computes such cost in a forward fashion, that is, starting from the beginning of a route.

Starting from the depot and following route τ, we denote by $\pi_i(q;\ \tau)$ and $\chi_i(q;\ \tau)$, respectively, the probability and expected cost that the vehicle reaches customer i with q units of remaining capacity after serving the demand of this customer; by $\mathbb{Q}_{i-1}(\tau)$ the vehicle available capacity after serving the demand of customer $i-1$; by $\underline{\mathbb{Q}}_{i-1}(\tau)$ and $\overline{\mathbb{Q}}_{i-1}(\tau)$, respectively, the minimal and maximal values of $\mathbb{Q}_{i-1}(\tau)$ such that $\pi_{i-1}(q_{i-1};\ \tau)$ is positive. In dynamic programming terminology, the functions $\chi_i(\cdot\,;\ \tau)$ and $\pi_i(\cdot\,;\ \tau)$, respectively, can be interpreted as the cost-to-come and the probability-to-come to customer i when using route τ. For notational simplicity, in the ensuing analysis we remove the dependence of these functions on the route.

The forward recursions that we present below assume that the capacity threshold values $\hat{q}_i$'s have already been determined by executing the recursion (1)–(3). Although it may seem pointless to focus on forward recursions that assume the prior execution of this backward recursion, such forward recursions can be useful in speeding up the execution of rollout algorithms for the single vehicle VRPSD, as discussed in Section 4.

The function $\pi_i(\cdot)$ satisfies the following recursion:

$$\pi_1(q) = 0, \quad \forall q \in \{0,\ldots,Q-\bar{L}_1-1\} \cup \{Q-\underline{L}_1+1,\ldots,Q-1\}, \tag{4}$$

$$\pi_1(q) = p_1(Q-q), \qquad \forall q = Q-\bar{L}_1,\ldots,Q-\underline{L}_1, \tag{5}$$

$$\pi_i(q) = \pi_i^A(q) + \pi_i^B(q) + \pi_i^C(q), \quad \forall i = 2,\ldots,n, \quad q = 0,\ldots,Q-1, \tag{6}$$

with

$$\begin{aligned}
\pi_i^A(q) &:= \sum_{\mathbb{Q}_{i-1}=\underline{q}_{i-1}}^{\hat{q}_{i-1}} \pi_{i-1}(\mathbb{Q}_{i-1})p_i(Q-q), \\
\forall q &= Q-\bar{L}_i,\ldots,Q-\underline{L}_i, \\
\pi_i^B(q) &:= \sum_{\mathbb{Q}_{i-1}=\max\{\underline{\mathbb{Q}}_{i-1},\hat{q}_{i-1}+1,q+\underline{L}_i\}}^{\min\{\overline{\mathbb{Q}}_{i-1},q+\bar{L}_i\}} \pi_{i-1}(\mathbb{Q}_{i-1})p_i(\mathbb{Q}_{i-1}-q), \\
\forall q &= \max\{0, \hat{\mathbb{Q}}_{i-1}+1-\bar{L}_i, \underline{\mathbb{Q}}_{i-1}-\bar{L}_i\},\ldots,\overline{\mathbb{Q}}_{i-1}-\underline{L}_i, \\
\pi_i^C(q) &:= \sum_{\mathbb{Q}_{i-1}=\max\{\underline{\mathbb{Q}}_{i-1},\hat{q}_{i-1}+1,q-Q+\underline{L}_i\}}^{\min\{\overline{\mathbb{Q}}_{i-1},q-Q+\bar{L}_i\}} \pi_{i-1}(\mathbb{Q}_{i-1})p_i(\mathbb{Q}_{i-1}+Q-q), \\
\forall q &= \max\{\underline{\mathbb{Q}}_{i-1}+Q-\bar{L}_i, \hat{q}_{i-1}+1+Q-\bar{L}_i\},\ldots, \\
&\quad \min\{\overline{\mathbb{Q}}_{i-1}+Q-\underline{L}_i, Q-1\},
\end{aligned}$$

and $\pi_i^{\mathrm{A}}(q)$, $\pi_i^{\mathrm{B}}(q)$, and $\pi_i^{\mathrm{C}}(q)$ set to zero for any other value of q. Expressions (4) and (5) set the probability of having q units of available capacity after serving the first customer equal to zero for q below $Q-\overline{L}_1$ or above $Q-\underline{L}_1$, and $p_1(Q-q)$ otherwise. For all customers 2 through n, the probability $\pi_i(q)$ in (6) is the sum of three probabilities corresponding to mutually exclusive and exhaustive events: $\pi_i^{\mathrm{A}}(q)$, $\pi_i^{\mathrm{B}}(q)$, and $\pi_i^{\mathrm{C}}(q)$, which are, respectively, the probability of reaching customer i when the vehicle restocks after serving customer $i-1$; does not restock after serving customer $i-1$ and no failure is experienced at customer i; and does not replenish after serving customer $i-1$ and a failure occurs at customer i. We now describe how to obtain each of these probabilities given that the function $\pi_{i-1}(\cdot)$ is available. The idea is to condition on the value of the vehicle capacity that is available after serving customer $i-1$, that is, $\mathbb{Q}_{i-1}$.

The vehicle replenishes after serving customer $i-1$ if $\underline{\mathbb{Q}}_{i-1} \le \mathbb{Q}_{i-1} \le \hat{q}_{i-1}$. Because the vehicle is full after replenishment, the probability $\pi_i^{\mathrm{A}}(q)$ can exceed zero only when $Q-\overline{L}_i \le q \le Q-\underline{L}_i$. For each such value of q, the corresponding demand at customer i is $Q-q$. Thus, $\pi_i^{\mathrm{A}}(q)$ is the sum of the probabilities $\pi_{i-1}(\mathbb{Q}_{i-1})p_i(Q-q)$ when $\mathbb{Q}_{i-1}$ varies from $\underline{\mathbb{Q}}_{i-1}$ through $\hat{q}_{i-1}$, inclusive.

There is no replenishment after serving customer $i-1$ if $\max\{\underline{\mathbb{Q}}_{i-1}, \hat{q}_{i-1}+1\} \le \mathbb{Q}_{i-1} \le \overline{\mathbb{Q}}_{i-1}$. If no failure occurs at customer i, the probability $\pi^{\mathrm{B}}(q)$ can be greater than zero only when $\max\{0, \hat{q}_{i-1}+1-\overline{L}_i, \underline{\mathbb{Q}}_{i-1}-\overline{L}_i\} \le q \le \overline{\mathbb{Q}}_{i-1}-\underline{L}_i$. Because for each of these values of q it holds that $q+\underline{L}_i \le \mathbb{Q}_{i-1} \le q+\overline{L}_i$, the probability $\pi_i^{\mathrm{B}}(q)$ is the sum of the probabilities $\pi_{i-1}(\mathbb{Q}_{i-1})p_i(\mathbb{Q}_{i-1}-q)$ for values of $\mathbb{Q}_{i-1}$ between $\max\{\underline{\mathbb{Q}}_{i-1}, \hat{q}_{i-1}+1, q+\underline{L}_i\}$ and $\min\{\overline{\mathbb{Q}}_{i-1}, q+\overline{L}_i\}$, inclusive. If a failure does occur at customer i, the probability $\pi^{\mathrm{C}}(q)$ can exceed zero only when $\max\{\underline{\mathbb{Q}}_{i-1}+Q-\overline{L}_i, \hat{q}_{i-1}+1+Q-\overline{L}_i\} \le q \le \min\{\overline{\mathbb{Q}}_{i-1}+Q-\underline{L}_i, Q-1\}$. Because for each of these values of q we have $q-Q+\underline{L}_i \le \mathbb{Q}_{i-1} \le q-Q+\overline{L}_i$, the probability $\pi_i^{\mathrm{C}}(q)$ is the sum of the probabilities $\pi_{i-1}(\mathbb{Q}_{i-1})p_i(\mathbb{Q}_{i-1}+Q-q)$ when $\mathbb{Q}_{i-1}$ is between $\max\{\underline{\mathbb{Q}}_{i-1}, \hat{q}_{i-1}+1, q-Q+\underline{L}_i\}$ and $\min\{\overline{\mathbb{Q}}_{i-1}, q-Q+\overline{L}_i\}$, inclusive.

Availability of the function $\pi_i(\cdot)$ allows us to obtain the following forward recursion for the function $\chi_i(\cdot)$:

$$\chi_1(q) = 0, \quad \forall q \in \{0,\dots,Q-\overline{L}_1-1\} \cup \{Q-\underline{L}_1+1,\dots,Q-1\}, \tag{7}$$

$$\chi_1(q) = c(0,1)p_1(Q-q), \qquad \forall q = Q-\overline{L}_1,\dots,Q-\underline{L}_1, \tag{8}$$

$$\chi_i(q) = \chi_i^{\mathrm{A}}(q) + \chi_i^{\mathrm{B}}(q) + \chi_i^{\mathrm{C}}(q), \quad \forall i = 2,\dots,n, \quad q = 0,\dots,Q-1, \tag{9}$$

$$\chi_{n+1}(q) = \chi_n(q) + c(n,0)\pi_n(q), \qquad \forall q = 0,\dots,Q-1, \tag{10}$$

with

$$\begin{aligned}
\chi_i^{\mathrm{A}}(q) &:= \sum_{\mathbb{Q}_{i-1}=\underline{\mathbb{Q}}_{i-1}}^{\hat{q}_{i-1}} \{\chi_{i-1}(\mathbb{Q}_{i-1}) + [c(i-1,0)+c(0,i)]\pi_{i-1}(\mathbb{Q}_{i-1})\} \\
&\quad p_i(Q-q), \\
\forall q &= Q-\overline{L}_i,\dots,Q-\underline{L}_i, \\
\chi_i^{\mathrm{B}}(q) &:= \sum_{\mathbb{Q}_{i-1}=\max\{\underline{\mathbb{Q}}_{i-1},\hat{q}_{i-1}+1,q+\underline{L}_i\}}^{\min\{\overline{\mathbb{Q}}_{i-1},q+\overline{L}_i\}} [\chi_{i-1}(\mathbb{Q}_{i-1}) + c(i-1,i)\pi_{i-1}(\mathbb{Q}_{i-1})] \\
&\quad p_i(\mathbb{Q}_{i-1}-q), \\
\forall q &= \max\{0, \hat{\mathbb{Q}}_{i-1}+1-\overline{L}_i, \underline{\mathbb{Q}}_{i-1}-L_i\},\dots,\overline{\mathbb{Q}}_{i-1}-\underline{L}_i,
\end{aligned}$$

Table 1
Travel costs $c(i, j)$'s for the example in Section 3.3.

	j			
i	0	1	2	3
0	0	1	$\sqrt{2}$	1
1	1	0	1	$\sqrt{2}$
2	$\sqrt{2}$	1	0	1
3	1	$\sqrt{2}$	1	0

$$\begin{aligned}
\chi_i^{\mathrm{C}}(q) &:= \sum_{\mathbb{Q}_{i-1}=\max\left\{\underline{\mathbb{Q}}_{i-1},\hat{q}_{i-1}+1,q-Q+\underline{L}_i\right\}}^{\min\left\{\overline{\mathbb{Q}}_{i-1},q-Q+\overline{L}_i\right\}} \{\chi_{i-1}(\mathbb{Q}_{i-1}) \\
&+ [c(i-1,i)+2c(i,0)]\pi_{i-1}(\mathbb{Q}_{i-1})\}p_i(\mathbb{Q}_{i-1}+Q-q), \\
\forall q &= \max\{\underline{\mathbb{Q}}_{i-1}+Q-\overline{L}_i, \hat{q}_{i-1}+1+Q-L_i\},\dots, \\
&\quad \min\{\overline{\mathbb{Q}}_{i-1}+Q-\underline{L}_i, Q-1\},
\end{aligned}$$

and $\chi_i^{\mathrm{A}}(q)$, $\chi_i^{\mathrm{B}}(q)$, and $\chi_i^{\mathrm{C}}(q)$ set to zero for any other value of q. Because the vehicle leaves the depot fully loaded, expressions (7) and (8) set to zero and $c(0,1)p_1(Q-q)$, respectively, the expected cost of having an amount of available capacity after serving the demand of the first customer that is incompatible and compatible with the possible realizations of this demand. For all customers 2 through n, the expected cost $\chi_i(q)$ in (9) is the sum of the expected costs of reaching such a customer when restocking is performed after serving the demand of the previous customer, $\chi_i^{\mathrm{A}}(q)$, and when no such restocking is performed and a route failure at the current customer does not occur, $\chi_i^{\mathrm{B}}(q)$, and does occur, $\chi_i^{\mathrm{C}}(q)$, respectively. Expression (10) sets the expected cost of having an amount of available capacity q when the vehicle returns to the depot after serving the last customer equal to the sum of the expected cost of having such capacity available after serving this customer, $\chi_n(q)$, and the cost of traveling from this customer to the depot, $c(n, 0)$, multiplied by the probability of having such available capacity at this juncture. The expression

$$\sum_{q=0}^{Q-1} \chi_{n+1}(q) \tag{11}$$

is the expected cost of route τ under the restocking strategy, that is, $\mathbb{E}[C_\tau]$.

Executing each of the forward recursions (4)–(6) and (7)–(10) requires $O(nQ)$ space and $O(nLQ)$ time.

3.3. Example

We consider the following instance: the depot and three customers are located at the corners of a unit square. Specifically, the depot location is $(0,0)$ and the respective locations of customers 1, 2, and 3 are $(0,1)$, $(1,1)$, and $(1,0)$. Travel costs correspond to Euclidean distances. Table 1 displays them. The customer demand probability mass functions are $p_1(1) = p_1(2) = 0.5$; $p_2(1) = 0.2$ and $p_2(2) = 0.8$; and $p_3(1) = 0.8$ and $p_3(2) = 0.2$. The vehicle capacity Q equals 3 units. We focus on route $\tau = (0,1,2,3,0)$.

Backward recursion: we first execute the backward recursion. The terminal conditions, which correspond to customer 3, are $\gamma_3(q) = c(3,0) = 1$ for each $q = 0,1,2$. For customer 2 we have

$$\begin{aligned}
\gamma_2^{(1)}(1) &= c(2,3) + p_3(1)\gamma_3(0) + p_3(2)[2c(3,0)+\gamma_3(2)] \\
&= 1 + 0.8\cdot 1 + 0.2(2+1) = 1 + 0.8 + 0.6 = 2.4, \\
\gamma_2^{(1)}(2) &= c(2,3) + p_3(1)\gamma_3(1) + p_3(2)\gamma_3(0) \\
&= 1 + 0.8\cdot 1 + 0.2\cdot 1 = 2, \\
\gamma_2^{(2)} &= c(2,0) + c(0,3) + p_3(1)\gamma_3(2) + p_3(2)\gamma_3(1) \\
&= \sqrt{2} + 1 + 0.8\cdot 1 + 0.2\cdot 1 = 2 + \sqrt{2} \approx 3.4142.
\end{aligned}$$

It follows that $\gamma_2(0) = 2 + \sqrt{2}$, because the triangle inequality implies $\gamma_2^{(1)}(0) \geq \gamma_2^{(2)}$, $\gamma_2(1) = 2.4$, $\gamma_2(2) = 2$, and $\hat{q}_2 = 0$. For customer 1 it holds that

$$\begin{aligned}\gamma_1^{(1)}(1) &= c(1,2) + p_2(1)\gamma_2(0) + p_2(2)[2c(2,0) + \gamma_2(2)]\\ &= 1{+}0.2(\sqrt{2}+2) + 0.8(2\sqrt{2}{+}2) = 3 + 1.8\sqrt{2} \approx 5.5456,\\ \gamma_1^{(1)}(2) &= c(1,2) + p_2(1)\gamma_2(1) + p_2(2)\gamma_2(0)\\ &= 1 + 0.2\cdot 2.4 + 0.8(\sqrt{2}+2) = 3.08 + 0.8\sqrt{2} \approx 4.2114,\\ \gamma_1^{(2)} &= c(1,0) + c(0,2) + p_2(1)\gamma_2(2) + p_2(2)\gamma_2(1)\\ &= 1 + \sqrt{2} + 0.2\cdot 2 + 0.8\cdot 2.4 = 3.32 + \sqrt{2} \approx 4.7342.\end{aligned}$$

We thus have $\gamma_1(0) = 3.32 + \sqrt{2}$, given that $\gamma_2^{(1)}(0) \geq \gamma_2^{(2)}$ holds by the triangle inequality, $\gamma_1(1) = 3.32 + \sqrt{2}$, $\gamma_1(2) = 3.08 + 0.8\sqrt{2}$, and $\hat{q}_1 = 1$. The expected cost of the given tour τ, $\mathbb{E}[C_\tau]$, corresponds to

$$\begin{aligned}\gamma_0(3) &= c(0,1) + p_1(1)\gamma_1(2) + p_1(2)\gamma_1(1)\\ &= 1 + 0.5(0.8\sqrt{2} + 3.08) + 0.5(1.8\sqrt{2} + 3) = 4.2 + 0.9\sqrt{2}.\end{aligned}$$

Forward recursion: we first evaluate the probabilities of having $q = 0, 1, 2$ units on board the vehicle after serving the demand of customer $i = 1, 2, 3$, that is, the quantities $\pi_i(q)$'s. Because the vehicle is fully loaded at the depot, $Q = 3$, $\underline{L}_1 = 1$, and $\overline{L}_1 = 2$, we have $\pi_1(0) = 0$, $\pi_1(1) = p_1(2) = 0.5$, and $\pi_1(2) = p_1(1) = 0.5$. For each customer $i = 2, 3$, the quantity $\pi_i(q)$ is the sum of

1. $\pi_i^{\mathrm{A}}(q)$: the probability of having q units left after serving the demand of customer i when the vehicle restocks on its way to this customer;
2. $\pi_i^{\mathrm{B}}(q)$: the probability of having q units left after serving the demand of customer i when the vehicle does not restock on its way to this customer and there is no failure at this customer; and
3. $\pi_i^{\mathrm{C}}(q)$: the probability of having q units left after serving the demand of customer i when the vehicle does not restock on its way to this customer and there is failure at this customer.

Given that $\hat{q}_1 = 1$, that is, restocking after serving the demand of customer 1 occurs when the residual number of units is 0 or 1, $\underline{L}_2 = 1$, and $\overline{L}_2 = 2$, it holds that

$$\begin{aligned}\pi_2^{\mathrm{A}}(0) &= 0,\\ \pi_2^{\mathrm{A}}(1) &= \pi_1(1)p_2(2) = 0.5\cdot 0.8 = 0.4,\\ \pi_2^{\mathrm{A}}(2) &= \pi_1(1)p_2(1) = 0.5\cdot 0.2 = 0.1,\\ \pi_2^{\mathrm{B}}(0) &= \pi_1(2)p_2(2) = 0.5\cdot 0.8 = 0.4,\\ \pi_2^{\mathrm{B}}(1) &= \pi_1(2)p_2(1) = 0.5\cdot 0.2 = 0.1,\\ \pi_2^{\mathrm{B}}(2) &= 0,\\ \pi_2^{\mathrm{C}}(0) &= \pi_2^{\mathrm{C}}(1) = \pi_2^{\mathrm{C}}(2) = 0.\end{aligned}$$

It follows that

$$\begin{aligned}\pi_2(0) &= 0 + 0.4 + 0 = 0.4,\\ \pi_2(1) &= 0.4 + 0.1 + 0 = 0.5,\\ \pi_2(2) &= 0.1 + 0 + 0 = 0.1.\end{aligned}$$

We have $\hat{q}_2 = 0$, that is, the vehicle restocks if and only if it is empty after serving the demand of customer 2, $\underline{L}_2 = 1$, and $\overline{L}_2 = 2$. Hence, it holds that

$$\begin{aligned}\pi_3^{\mathrm{A}}(0) &= 0,\\ \pi_3^{\mathrm{A}}(1) &= \pi_2(0)p_3(2) = 0.4\cdot 0.2 = 0.08,\\ \pi_3^{\mathrm{A}}(2) &= \pi_2(0)p_3(1) = 0.4\cdot 0.8 = 0.32,\\ \pi_3^{\mathrm{B}}(0) &= \pi_2(1)p_3(1) + \pi_2(2)p_3(2) = 0.5\cdot 0.8\\ &\quad + 0.1\cdot 0.2 = 0.4 + 0.02 = 0.42,\\ \pi_3^{\mathrm{B}}(1) &= \pi_2(2)p_3(1) = 0.1\cdot 0.8 = 0.08,\\ \pi_3^{\mathrm{B}}(2) &= 0,\\ \pi_3^{\mathrm{C}}(0) &= \pi_3^{\mathrm{C}}(1) = 0,\\ \pi_3^{\mathrm{C}}(2) &= \pi_2(1)p_3(2) = 0.5\cdot 0.2 = 0.1.\end{aligned}$$

Consequently, we have

$$\begin{aligned}\pi_3(0) &= 0 + 0.42 + 0 = 0.42,\\ \pi_3(1) &= 0.08 + 0.08 + 0 = 0.16,\\ \pi_3(2) &= 0.32 + 0 + 0.1 = 0.42.\end{aligned}$$

We now evaluate the expected cost $\chi_i(q)$ of serving the demand of customers 1 through i and having q units left in the vehicle for $i = 1, 2, 3$ and $q = 0, 1, 2$, as well as the expected cost $\chi_4(q)$ of reaching the depot with q units on board the vehicle after satisfying the demand of all the three customers for $q = 0, 1, 2$. Given that the vehicle starts full, $Q = 3$, $\underline{L}_1 = 1$, and $\overline{L}_1 = 2$, it holds that $\chi_1(0) = 0$, $\chi_1(1) = c(0,1)p_1(2) = 1\cdot 0.5 = 0.5$, and $\chi_1(2) = c(0,1)p_1(1) = 1\cdot 0.5 = 0.5$. For each customer $i = 2, 3$, the term $\chi_i(q)$ is the total of

1. $\chi_i^{A}(q)$: the expected cost of having q units on board after serving the demand of customer i when the vehicle restocks right before reaching this customer;
2. $\chi_i^{B}(q)$: the expected cost of having q units on board after serving the demand of customer i when the vehicle does not restock right before reaching this customer and there is no failure at this customer; and
3. $\chi_i^{C}(q)$: the expected cost of having q units on board after serving the demand of customer i when the vehicle does not restock after serving this customer and there is failure at this customer.

We have $\hat{q}_1 = 1$, $\underline{L}_2 = 1$, and $\overline{L}_2 = 2$. It follows that

$$\begin{aligned}\chi_2^{\mathrm{A}}(0) &= 0,\\ \chi_2^{\mathrm{A}}(1) &= \{\chi_1(1) + [c(1,0) + c(0,2)]\pi_1(1)\}p_2(2)\\ &= \left[0.5 + (1+\sqrt{2})0.5\right]0.8 = 0.8 + 0.4\sqrt{2},\\ \chi_2^{\mathrm{A}}(2) &= \{\chi_1(1) + [c(1,0) + c(0,2)]\pi_1(1)\}p_2(1)\\ &= [0.5 + (1 + \sqrt{2}0.5]0.2 = 0.2 + 0.1\sqrt{2},\\ \chi_2^{\mathrm{B}}(0) &= [\chi_1(2) + c(1,2)\pi_1(2)]p_2(2) = (0.5 + 1\cdot 0.5)0.8 = 0.8,\\ \chi_2^{\mathrm{B}}(1) &= [\chi_1(2) + c(1,2)\pi_1(2)]p_2(1) = (0.5 + 1\cdot 0.5)0.2 = 0.2,\\ \chi_2^{\mathrm{B}}(2) &= 0,\\ \chi_2^{\mathrm{C}}(0) &= \chi_2^{\mathrm{C}}(1) = \chi_2^{\mathrm{C}}(2) = 0,\end{aligned}$$

and

$$\begin{aligned}\chi_2(0) &= 0 + 0.8 + 0 = 0.8,\\ \chi_2(1) &= 0.8 + 0.4\sqrt{2} + 0.2 + 0 = 1 + 0.4\sqrt{2},\\ \chi_2(2) &= 0.2 + 0.1\sqrt{2} + 0 + 0 = 0.2 + 0.1\sqrt{2}.\end{aligned}$$

Given that $\hat{q}_2 = 0$, $\underline{L}_2 = 1$, and $\overline{L}_2 = 2$, we have

$$\begin{aligned}\chi_3^{\mathrm{A}}(0) &= 0,\\ \chi_3^{\mathrm{A}}(1) &= \{\chi_2(0) + [c(2,0) + c(0,3)]\pi_2(0)\}p_3(2)\\ &= [0.8 + (\sqrt{2}+1)0.4]0.2 = 0.24 + 0.08\sqrt{2},\\ \chi_3^{\mathrm{A}}(2) &= \{\chi_2(0) + [c(2,0) + c(0,3)]\pi_2(0)\}p_3(1)\\ &= [0.8 + (\sqrt{2}+1)0.4]0.8 = 0.96 + 0.32\sqrt{2},\\ \chi_3^{\mathrm{B}}(0) &= [\chi_2(1) + c(2,3)\pi_2(1)]p_3(1)\\ &\quad + [\chi_2(2) + c(2,3)\pi_2(2)]p_3(2)\\ &= (1 + 0.4\sqrt{2} + 1\cdot 0.5)0.8 + (0.2 + 0.1\sqrt{2} + 1\cdot 0.1)0.2\\ &= 1.2 + 0.32\sqrt{2} + 0.06 + 0.02\sqrt{2} = 1.26 + 0.34\sqrt{2},\\ \chi_3^{\mathrm{B}}(1) &= [\chi_2(2) + c(2,3)\pi_2(2)]p_3(1)\\ &= (0.2 + 0.1\sqrt{2} + 1\cdot 0.1)0.8 = 0.24 + 0.08\sqrt{2},\end{aligned}$$

$$\begin{aligned}\chi_3^{\mathrm{B}}(2) &= 0,\\ \chi_3^{\mathrm{C}}(0) &= \chi_3^{\mathrm{C}}(1) = 0,\\ \chi_3^{\mathrm{C}}(2) &= \{\chi_2(1) + [c(2,3) + 2c(3,0)]\pi_2(1)\}p_3(2)\\ &= [1 + 0.4\sqrt{2} + (1 + 2\cdot 1)0.5]0.2 = 0.5 + 0.08\sqrt{2},\end{aligned}$$

and

$$\begin{aligned}\chi_3(0) &= 0 + 1.26 + 0.34\sqrt{2} + 0 = 1.26 + 0.34\sqrt{2},\\ \chi_3(1) &= 0.24 + 0.08\sqrt{2} + 0.24 + 0.08\sqrt{2} + 0 = 0.48 + 0.16\sqrt{2},\\ \chi_3(2) &= 0.96 + 0.32\sqrt{2} + 0.5 + 0.08\sqrt{2} = 1.46 + 0.4.\end{aligned}$$

Finally, we obtain

$$\begin{aligned}\chi_4(0) &= \chi_3(0) + c(3,0)\pi_3(0) = 1.26 + 0.34\sqrt{2} + 1\cdot 0.42\\ &= 1.68 + 0.34\sqrt{2},\\ \chi_4(1) &= \chi_3(1) + c(3,0)\pi_3(1) = 0.48 + 0.16\sqrt{2} + 1\cdot 0.16\\ &= 0.64 + 0.16\sqrt{2},\\ \chi_4(2) &= 1.46 + 0.4\sqrt{2} + 1\cdot 0.42 = 1.88 + 0.4\sqrt{2},\end{aligned}$$

so that the expected cost of tour τ, $\mathbb{E}[C_\tau]$, evaluates to

$$\sum_{q=0}^{2}\chi_4(0) = 1.68 + 0.34\sqrt{2} + 0.64 + 0.16\sqrt{2} + 1.88 + 0.4\sqrt{2} = 4.2 + 0.9\sqrt{2}.$$

4. Proposed expected cost computation approach for rollout algorithms

In this section we integrate the forward and backward recursions introduced in Sections 3.1 and 3.2 into a hybrid recursion that approximately computes the expected cost of a route in the restocking case, and establish its theoretical benefit for any rollout algorithm for VRPSD with a single vehicle under the restocking strategy. We present a general rollout algorithm for this VRPSD version in Section 4.1. We discuss our proposed method and its role for such algorithm in Section 4.2.

4.1. General rollout algorithm

Rollout algorithms build a feasible route in an iterative fashion. At each iteration, they add a customer right after the last customer, if any, in the current partial (infeasible) route, which is interpreted as the head of a complete route. Specifically, assume we have available a base heuristic $\mathcal{H}$ that can generate a complete (feasible) route starting from any current partial route with $0 \le u < n$ customers by adding to this partial route all the remaining $n-u$ customers. Let $\tau_u = (0, i_1, i_2, \ldots, i_u)$ be the current partial route, with $\tau_0 := (0)$; $N(n-u)$ the set of customers not in τ_u; and $H(i, \tau_u)$ the expected cost under the restocking strategy of the feasible route obtained by inserting node $i \in N(n-u)$ in position $u+1$ in τ_u and then applying heuristic $\mathcal{H}$ to the resulting partial route. At iteration $u = 0, \ldots, n-1$ the rollout algorithm based on this heuristic appends a new customer $i' \in N(n-u)$ in position $u+1$ in τ_u to generate a new partial route τ_{u+1} by selecting i' as follows:

$$i' \in \arg\min_{i \in N(n-u)} H(i, \tau_u). \tag{12}$$

If $u+1 = n$, this rollout algorithm terminates by appending the depot to partial route τ_{u+1} to obtain a complete route. Otherwise, the algorithm proceeds to iteration $u+1$. In Section 5.1 we describe four rollout algorithms by specifying the base heuristic that they use.

4.2. Hybrid recursion and its role for rollout algorithms

The backward recursion (1)–(3) and the forward recursion (7)–(10) have the same computational complexity. However, the backward recursion is computationally more efficient than the forward recursion: the forward recursion requires that the backward recursion be executed first, the forward recursion involves computing two functions rather than one function, and the computational complexity of determining each of these functions is the same. Use of the backward recursion when executing a rollout algorithm for VRPSD with a single vehicle and restocking is thus preferable to using the forward recursion. However, combining the forward and backward recursions into a hybrid recursion provides a potentially computationally advantageous, yet approximate, alternative to implement such algorithms.

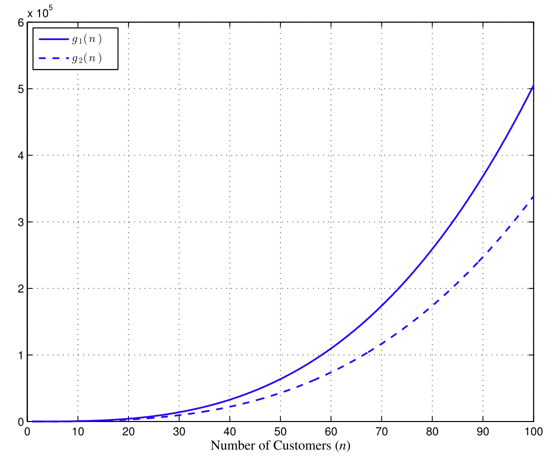
Recall that τ_u denotes the partial route available at iteration $u \ge 1$ of a rollout algorithm, that is, $(0, i_1, i_2, \ldots, i_u)$. Let $\tau^\diamond$ be the best feasible route obtained by this algorithm at this iteration; that is, τ_u is the head of $\tau^\diamond$. Applying the forward recursion to route $\tau^\diamond$ yields the probability-to-come and cost-to-come functions $\pi_u(\cdot;\, \tau^\diamond)$ and $\chi_u(q;\, \tau^\diamond)$. The uth iteration of the rollout algorithm involves computing the expected costs of a number of complete routes proportional to $n-u$, each with head equal to the partial route τ_u but with different tails (the exact number of complete routes that need to be evaluated depends on the chosen rollout algorithm). The tail of each of these routes can be represented as a partial route $\overline{\tau}_{u+1}$ that starts from some customer i_{u+1}, includes all the customers in set $N(n-u)$, and ends at the depot. Denote by $\tau^\oplus$ the complete route obtained by joining the head partial route τ_u and the tail partial route $\overline{\tau}_{u+1}$. The backward recursion can be applied to the tail partial route that results from joining i_u and $\overline{\tau}_{u+1}$ to determine $\gamma_u(\cdot;\, \tau^\oplus)$. The expected restocking cost of the route $\tau^\oplus$, $\mathbb{E}[C_{\tau^\oplus}]$, can then be computed as

$$\sum_{q=0}^{Q-1}\left[\chi_u(q;\, \tau^\diamond) + \pi_u(q;\, \tau^\diamond)\gamma_u(q;\, \tau^\oplus)\right]. \tag{13}$$

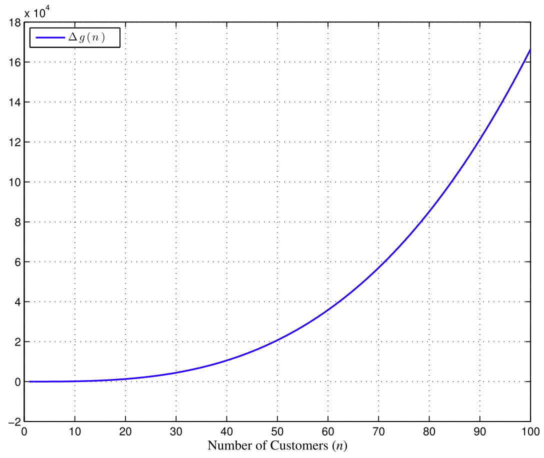
Expression (13) suggests executing any rollout algorithm using a hybrid recursion: a version of the forward recursion based on *approximate* replenishment thresholds incrementally approximates the probability-to-come and cost-to-come functions for the head of the route being constructed, whereas the backward recursion computes the cost-to-go from the last customer in the head of this route and gives the replenishment threshold associated with this customer for any complete route that includes this route head. This approach approximates the probability-to-come and cost-to-come functions because when the rollout algorithm adds a customer to a partial route the replenishment threshold corresponding to this customer, which is used by the incremental forward recursion at the completion of the next rollout iteration, is fixed to be the one determined by the application of the backward recursion at this rollout iteration. In other words, when the incremental forward recursion evaluates the probability-to-come and cost-to-come functions for the customer that the rollout algorithm adds to a partial route, it does not recompute the replenishment thresholds of the other customers in this route; that is, the values of these thresholds are the ones obtained when these customers were added in their respective previous iterations of the rollout algorithm. Formally, letting $\check{\pi}_u(\cdot;\tau^\diamond)$ and $\check{\chi}_u(\cdot;\tau^\diamond)$ be these approximate probability-to-come and cost-to-come functions for customer i_u in route $\tau^\diamond$, our approach approximates (13) with

$$\sum_{q=0}^{Q-1}\left[\check{\chi}_u(q;\, \tau^\diamond) + \check{\pi}_u(q;\, \tau^\diamond)\gamma_u(q;\, \tau^\oplus)\right]. \tag{14}$$

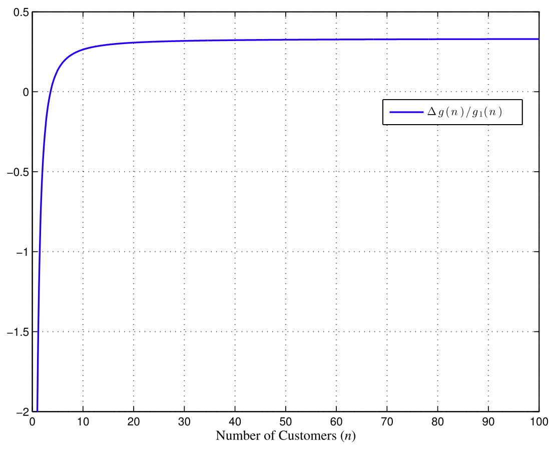
At iteration u, a rollout algorithm evaluates a number of routes proportional to $n-u$. With the backward recursion the computa-



(i) Theoretical Computational Efforts $g_1(n)$ and $g_2(n)$



(ii) Theoretical Speed-up $\Delta g(n)$



(iii) Theoretical Speed-up Factor $\Delta g(n)/g_1(n)$

Fig. 1. Illustration of the theoretical (i) computational requirements $g_1(n)$ and $g_2(n)$, (ii) speed-up $\Delta g(n)$, and (iii) speed-up factor $\Delta g(n)/g_1(n)$.

tional effort exerted to evaluate a single route is proportional to n. Consequently, the computational onus of implementing a rollout algorithm using the backward recursion is proportional to

$$g_1(n) = \sum_{u=0}^{n-1} n(n-u) = \frac{n^2(n+1)}{2}, \tag{15}$$

where we use the formula $\sum_{u=0}^{n-1} u = (n-1)n/2$. With the hybrid recursion the computational burden required to evaluate the expected cost of a given route at iteration u of a rollout algorithm is proportional to $n-u$. Moreover, computing the probability-to-come and the cost-to-come functions throughout the execution of this algorithm requires work proportional to $2n$. The computational requirement of executing a rollout algorithm using the hybrid recursion is then proportional to

$$g_2(n) = 2n + \sum_{u=0}^{n-1} (n-u)^2 = \frac{n(2n^2+3n+13)}{6}, \tag{16}$$

where we employ the formula $\sum_{u=0}^{n-1} u^2 = n(n-1)[2(n-1)+1]/6$.

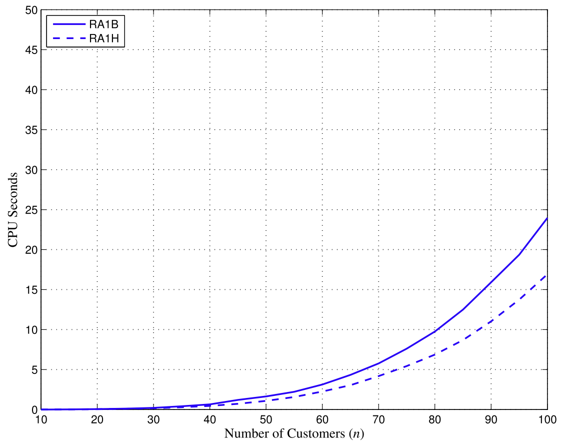
Theoretically, executing a rollout algorithm based on (14) rather than (3) is computationally advantageous in a big-o order sense when the function $g_1(n)$ is larger than the function $g_2(n)$. Panel (i) of Fig. 1 illustrates these theoretical computational requirements. The corresponding theoretical speed-up is the difference function $\Delta g(n) := g_1(n) - g_2(n) = n(n^2-13)/6$. Panel (ii) of Fig. 1 displays this speed-up. Thus, roughly, there should be an advantage from using (14) rather than (3) when the number of customers n exceeds 4, because $\sqrt{13} \approx 3.6$, and this advantage increases nonlinearly in this number. Further, the theoretical speed-up factor is the function $\Delta g(n)/g_1(n) = (n - 13/n)/[3(n+1)]$, which quickly increases to and remains constant at 1/3, as displayed in panel (iii) of Fig. 1. This analysis suggests that there should be a substantial reduction of the computational effort from implementing a rollout algorithm using the hybrid recursion rather than the backward recursion for a sufficiently large number of customers, with a corresponding theoretical speed up factor of 1/3, which should ensue even when this number is moderate.

5. Computational study

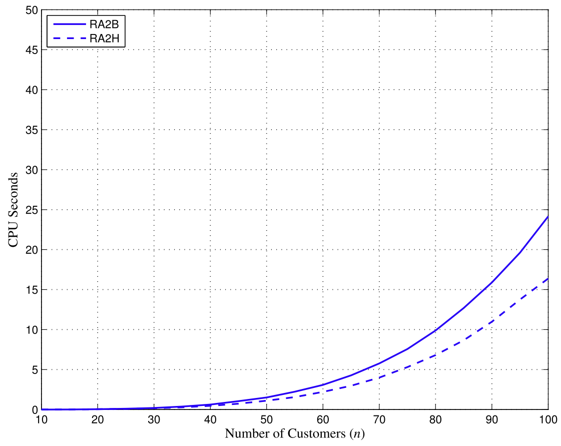
In this section we assess the performance of our expected cost evaluation method proposed in Section 4.2 applied to the basic rollout algorithm of Secomandi (2003) and three variants thereof on the instances of Secomandi and Margot (2009). We introduce these rollout algorithms in Section 5.1 and these instances in Section 5.2. We discuss our results in Section 5.3.

5.1. Specific rollout algorithms

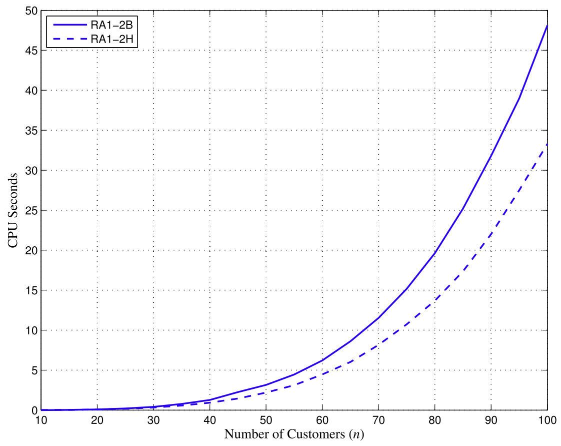
The basic rollout algorithm of Secomandi (2003), which we label RA1, uses as base heuristic the computational inexpensive cyclic heuristic (Bertsimas, 1992). Given the route $(0, 1, \ldots, n, 0)$, where the customers have been reordered for convenience, the



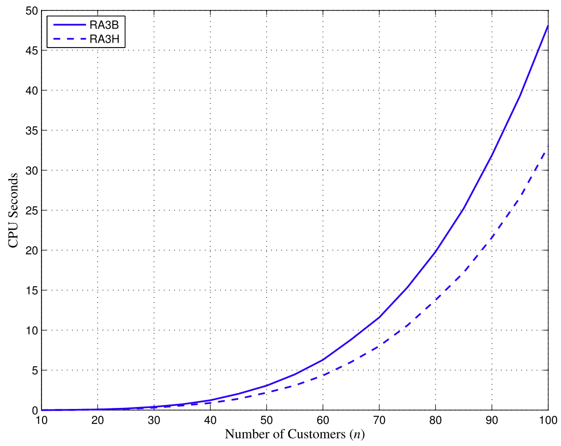
(i) RA1, Corner, Low $\overline{f}$



(ii) RA2, Corner, Low $\overline{f}$



(iii) RA1-2, Corner, Low $\overline{f}$



(iv) RA3, Corner, Low $\overline{f}$

Fig. 2. The observed computational requirement of the various rollout algorithms as a function of the number of customers on the corner instances with low $\overline{f}$.

route obtained by this heuristic starting from customer i is $(0, i, i+1, \ldots, n, 1, \ldots, i-1, 0)$. When applied to the partial route τ_u after having inserted customer i in position $u+1$, this heuristic skips customers that are already included in τ_u. For example, if $n = 3$ and the initial route is $(0, 1, 2, 3, 0)$, the routes generated at the first iteration are $(0, 1, 2, 3, 0)$, $(0, 2, 3, 1, 0)$, and $(0, 3, 1, 2, 0)$. Denote as $H1(i, \tau_u)$ the expected cost of the route generated by the cyclic heuristic when appending customer i to partial route τ_u. RA1 is the rollout algorithm corresponding to replacing $H(i, \tau_u)$ with $H1(i, \tau_u)$ in (12).

The first variant of RA1 uses as base heuristic the reversed cyclic heuristic. This heuristic simply travels a given route from right-to-left rather than left-to-right. Denote by $H2(i, \tau_u)$ the expected cost of the route obtained by the reversed cyclic heuristic when appending customer i to partial route τ_u. The rollout algorithm based on the reversed cyclic heuristic corresponds to using $H2(i, \tau_u)$ instead of $H(i, \tau_u)$ in (12). We label as RA2 this variant of RA1. For example, if $n = 3$ and the given route is $\tau = (0, 1, 2, 3, 0)$, then RA2 considers the following routes at the first iteration: $(0, 1, 3, 2, 0)$, $(0, 2, 1, 3, 0)$, and $(0, 3, 2, 1, 0)$.

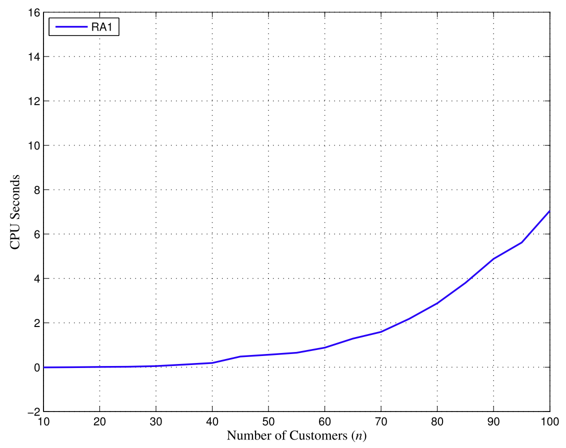
The second rollout algorithm variant runs both RA1 and RA2 and picks the best of the two resulting routes. We label this rollout algorithm as RA1-2. This algorithm is related to the “stostat” rollout algorithm of Novoa and Storer (2009).

We obtain a third variant of RA1 by using as base heuristic both the original cyclic heuristic and its reversed version. Specifically, this base heuristic runs both versions of the cyclic heuristic and picks the best of the two routes so obtained. Formally, define $H3(i, \tau_u)$ as the smallest of $H1(i, \tau_u)$ and $H2(i, \tau_u)$: $H3(i, \tau_u) := \min\{H1(i, \tau_u), H2(i, \tau_u)\}$. Our third rollout algorithm variant corresponds to using $H3(i, \tau_u)$ in lieu of $H(i, \tau_u)$ in (12). We dub it RA3. This algorithm resembles the “combi” and “stostat” rollout algorithms of Novoa and Storer (2009).

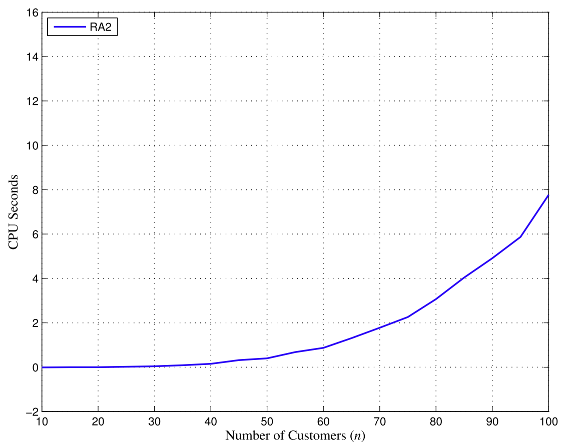
5.2. Instances

The distinguishing features of the instances of Secomandi and Margot (2009) follow:

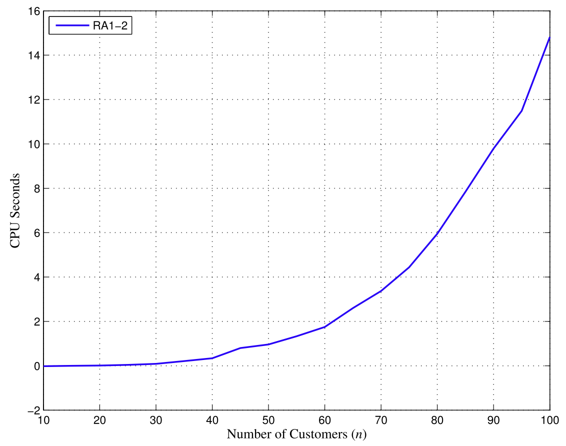
- Number of customers, n: values from 10 to 100 in increments of 5.
- Customer locations: random points, with integer coordinates, in a square with side equal to 1000.
- Depot location: the southwest corner, (0,0), or the center, (500,500), of this square.
- Customer demands: low, medium, and high discrete uniform random variables with respective supports $\{1, 2, \ldots, 5\}$, $\{6, 7, \ldots, 10\}$, and $\{11, 12, \ldots, 15\}$. The demand of each customer is assigned to one of these random variables with equal probability.
- Expected filling rate (total expected demand divided by vehicle capacity), $\overline{f} := \sum_{i=1}^{n} \mathbb{E}[D_i]/Q$: values of 1.6 and 1.9, referred to as low and high in the ensuing discussion.



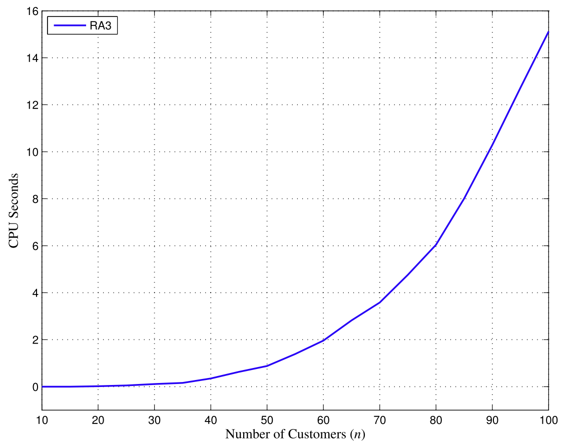
(i) RA1, Corner, Low $\overline{f}$



(ii) RA2, Corner, Low $\overline{f}$



(iii) RA1-2, Corner, Low $\overline{f}$



(iv) RA3, Corner, Low $\overline{f}$

Fig. 3. The observed speed-up of the hybrid expected cost computation for the various rollout algorithms as a function of the number of customers on the corner instances with low $\overline{f}$.

- Vehicle capacity, Q: the value taken by the ratio $8n/\overline{f}$ rounded off to the nearest integer (the average of the means of the low, medium, and high demand random variables is 8).

There are ten randomly generated instances for each combination of these parameters. The total number of instances is thus 760. Each instance includes an initial route through all the customers starting and ending at the depot. This route is generated by heuristically solving a traveling salesman problem, that is, ignoring customer demands, using a nearest neighbor algorithm followed by a 2-Int procedure, as in Secomandi (2003). All the considered rollout algorithms use this route as the initial one.

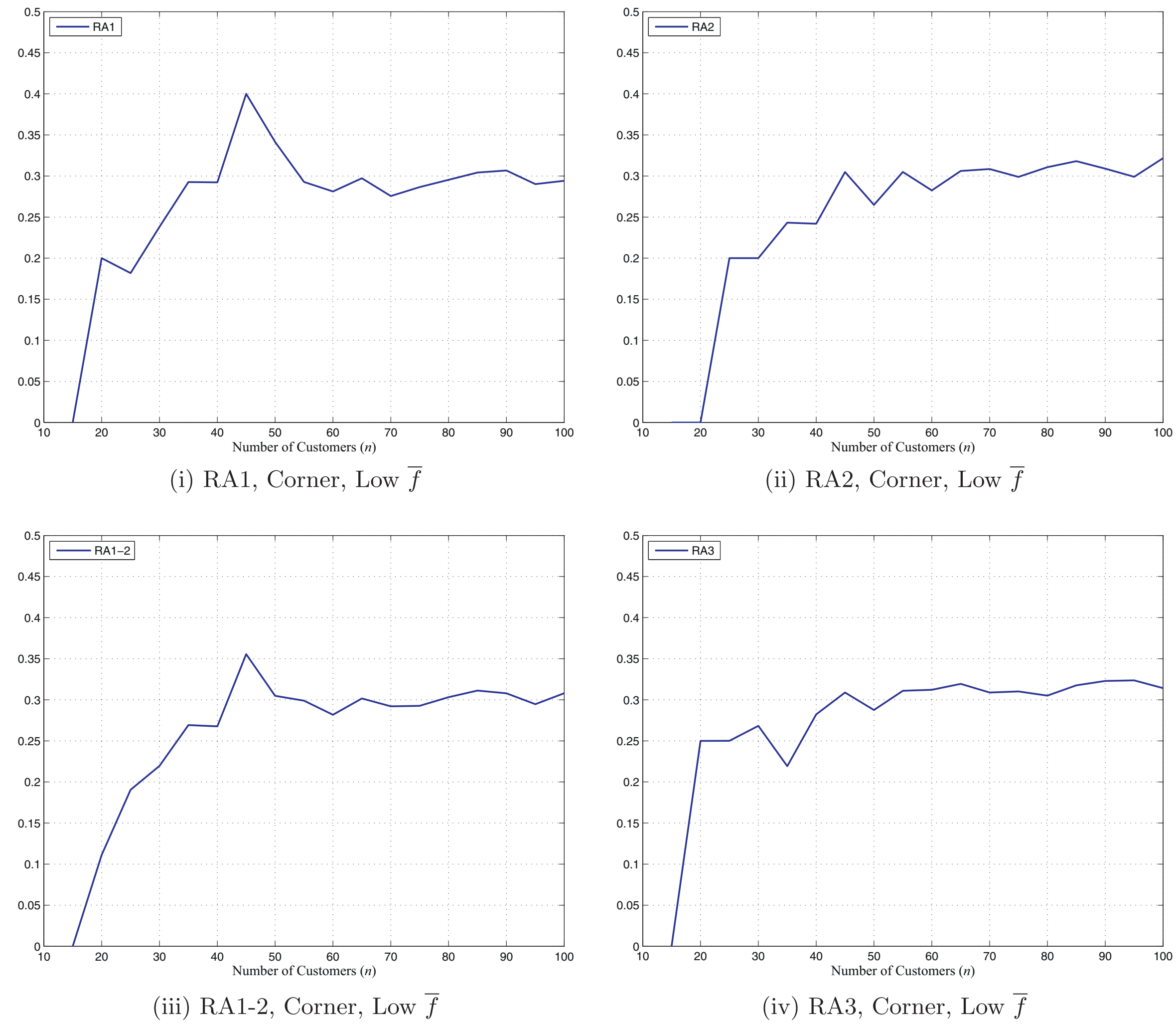
5.3. Results

Before presenting the computational requirements, speed-ups, and speed-up factors observed when applying our proposed expected cost evaluation method on the considered instances, we discuss the quality of the routes obtained by the rollout algorithms that we analyze—detailed results are available in the online supplementary material. This discussion sheds light on the magnitude of the effect of the expected cost evaluation error incurred by our proposed method on the quality of the resulting routes.

We denote by RA1B and RA1H, respectively, the versions of RA1 executed with the backward and hybrid evaluations of the expected cost of a route. We use the same labeling convention for RA2, RA1-2, and RA3. Table 2 reports the average expected cost of all these rollout algorithms on the center, corner, and all the instances. The entries corresponding to RA1-2B/H are not the minimum of their respective RA1B/H and RA2B/H entries because the average of the minimum of the expected costs of RA1B/H and RA2B/H is not the minimum of the average of these two expected costs. Even though the hybrid evaluation of the expected cost of a route can affect the quality of the route obtained by a rollout algorithm, the observed effect of this approximation on the effectiveness of these algorithms is marginal. Specifically, the additional expected costs resulting from this approximation are 0.12%, 0.01%, and 0.06% on the center, corner, and all the instances for RA1 (the average expected costs that are compared here are the ones corresponding to RA1H and RA1B; analogous comparisons are performed next for the other three rollout algorithms); 0.01% on each type of instance for RA2; 0.08%, 0.00%, and 0.04% on the center, corner, and all the instances for RA1-2; and 0.12% on each type of instance for RA3. This finding is consistent with the one of

Table 2
Average expected cost of the different rollout algorithms.

Instances	RA1B	RA2B	RA1-2B	RA3B
Center	6739	6755	6687	6688
Corner	7886	7892	7815	7810
All	7312	7324	7251	7249
	RA1H	RA2H	RA1-2H	RA3H
Center	6747	6756	6692	6696
Corner	7886	7893	7815	7820
All	7317	7324	7254	7258



(i) RA1, Corner, Low $\overline{f}$ (ii) RA2, Corner, Low $\overline{f}$

(iii) RA1-2, Corner, Low $\overline{f}$ (iv) RA3, Corner, Low $\overline{f}$

Fig. 4. The observed speed-up factor of the hybrid expected cost computation for the various rollout algorithms as a function of the number of customers on the corner instances with low $\overline{f}$. The observed speed-up factors for n equal to 10 are not displayed because they are undefined (the CPU seconds of both the backward and hybrid expected cost computation approaches are essentially zero).

Yang et al. (2000, p. 104) established in the context of a different approximation of (the change in) the expected cost of a restocking tour as a consequence of inserting or deleting a string of nodes in such tour when executing Or-opt-type heuristics. Incidentally, compared to their respective RA1 versions overall the considered RA2 versions yield routes with moderately inferior quality, whereas the examined RA1-2 and RA3 versions generate routes with superior, and roughly comparable, quality.

All the considered rollout algorithms are coded in C and run on an Intel Core i3 CPU M 350, 2.27 gigahertz, 4 gigabytes RAM computer. Figs. 2–4, respectively, display the observed computational requirement of each of these rollout algorithms executed under both the backward and hybrid expected cost computation approaches, the observed speed-up achieved by each of the hybrid versions of these algorithms, and the corresponding observed speed-up factor when varying the number of customers for the corner instances with low $\overline{f}$. The analogous charts for all the other cases are similar to the ones reported here. We omit them for brevity.

The plots in Figs. 2–4 resemble at least qualitatively their respective plots in panels (i)–(iii) of Fig. 1; as discussed below, there is also a quantitative resemblance between Fig. 4 and panel (iii) of Fig. 1, despite some notable variation in the charts displayed in Fig. 4 that is absent from the graph of the function given in panel (iii) of Fig. 1. Specifically, there is a computational advantage in using the hybrid rather than the backward computation of the expected cost of a route with about 30 customers. In contrast, the two approaches require similar computational effort with fewer customers. The observed benefit from using the hybrid approach becomes noticeable with about 40–50 customers and, with a few exceptions, raises at an increasing rate thereafter. With more than 50 customers the observed speed-up factors vary between 0.26 and 0.34. These ranges include the limiting value of 1/3 established in Section 4 for the theoretical big-o order speed-up factor $\Delta g(n)/g_1(n)$. Overall, these findings (i) are consistent with our theoretical analysis conducted in Section 4 and (ii) suggest that the hybrid approach yields substantial computational benefits for instances with a moderate to large number of customers, albeit at the cost of a minor degradation in solution quality.

6. Conclusions

Rollout algorithms are known to yield good solutions for VRPSD with a single vehicle, an important model for research and applications in logistics under uncertainty. Nonetheless, their computational burden can be substantial. To alleviate this issue, we develop a novel approach to approximate the expected cost of a VRPSD route under the restocking strategy that applies to any rollout algorithm. We establish that the theoretical speed-up factor of our technique is of big-o order equal to 1/3 when the number of customers is sufficiently large. Our numerical study, based on a set of instances from the literature and a known rollout algorithm and three variants thereof, indicates that our proposed method yields observed speed-up factors that vary between 0.26 and 0.34 when there are at least 50 customers. These ranges are in line with the 1/3 limiting value for the theoretical big-o order speed-up factor. These savings are achieved by only marginally degrading the quality of the resulting restocking routes.

Additional research could investigate extending our proposed approach to other heuristics for VRPSD under restocking (see, e.g., Bianchi et al., 2006; Yang et al., 2000) or variants of this problem that feature both multiple vehicles and restrictions on the tour cost (see, e.g., Yang et al., 2000) or different objective functions (see, e.g., Goodson et al., 2013; Goodson et al., 2016); compare the method put forth in this paper against techniques that approximate the objective function of the model that we study (e.g., a possible extension of the approach of Bianchi et al., 2006 from the a priori to the restocking case); and shed some light on the small observed impact of the approximate restocking thresholds used by our proposed method on the quality of the solutions found by the rollout algorithms that we consider when executed using this technique.

Acknowledgment

We thank three anonymous reviewers for their comments that led to an improved version of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ejor.2018.03.034.

References

Bertazzi, L. (2012). Minimum and worst-case performance ratios of rollout algorithms. *Journal of Optimization Theory and Applications, 152*(2), 378–393.

Bertsekas, D. (2005). *Dynamic programming and optimal control*: 1 (3rd). Belmont, MA: Athena Scientific.

Bertsekas, D., & Castanon, D. (1999). Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics, 5*(1), 89–108.

Bertsekas, D., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

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

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

Bertsimas, D., Jaillet, P., & Odoni, A. (1990). A priori optimization. *Operations Research, 38*(6), 1019–1033.

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

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, Q., & Schiavinotto, T. (2006). Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms, 5*(1), 91–110.

Birattari, M., Balaprakash, P., Stützle, T., & Dorigo, M. (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.

Ciavotta, M., Meloni, C., & Pranzo, M. (2016). Speeding up a rollout algorithm for complex parallel machine scheduling. *International Journal of Production Research, 54*(16), 4993–5009.

Dantzig, G., & Ramser, J. (1959). The truck dispatching problem. *Management Science, 6*(1), 80–91.

Dror, M. (2002). Vehicle routing with stochastic demands: models and computational methods. In M. Dror, P. L'Ecuyer, & F. Szidarouszky (Eds.), *Modeling uncertainty: An examination of stochastic theory, methods, and applications*. In International series in operations research and management science*: 46* (pp. 625–649). Boston, MA: Kluwer.

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

Fisher, M. (1995). Vehicle routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, & G. L. Nemhauser (Eds.), *Network routing*. In Handbooks in operations research and management science*: 8* (pp. 1–33). Amsterdam, The Netherlands: Elsevier.

Gendreau, M., Jabali, O., & Rei, W. (2016). 50th anniversary invited article—Future 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.

Gendreau, M., Laporte, G., & Séguin, R. (1996a). Stochastic vehicle routing. *European Journal of Operational Research, 88*(1), 3–12.

Gendreau, M., Laporte, G., & Séguin, R. (1996b). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research, 44*(3), 469–477.

Goodson, J., Ohlmann, J., & Thomas, B. (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., Ohlmann, J., & Thomas, B. (2013). Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research, 61*(1), 138–154.

Goodson, J., Thomas, B., & Ohlmann, J. (2016). Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Science, 50*(2), 591–607.

Goodson, J., Thomas, B., & Ohlmann, J. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research, 258*(1), 216–229.

Guerriero, F., & Mancini, M. (2005). Parallelization strategies for rollout algorithms. *Computational Optimization and Applications, 31*(2), 221–244.

Guerriero, F., Mancini, M., & Musmanno, R. (2002). New rollout algorithms for combinatorial optimization problems. *Optimization Methods and Software, 17*(4), 627–654.

Gupta, A., Viswanath, N., & Ravi, R. (2012). Approximation algorithms for VRP with stochastic demands. *Operations Research, 60*(1), 123–127.

Hjorring, C., & Holt, J. (1999). New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research, 86*(0), 569–584.

Jabali, O., Rei, W., Gendreau, M., & Laporte, G. (2012). New valid inequalities for the multi-vehicle routing problem with stochastic demands. *Technical Report CIRRELT-2012-58*. Montréal, Québec, Canada: CIRRELT, Université de Montréal.

Laporte, G. (1992). The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(3), 345–358.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408–416.

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

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.

Rei, W., Gendreau, M., & Soriano, P. (2010). A hybrid Monte Carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science, 44*(1), 136–146.

Secomandi, N. (1998). *Exact and heuristic dynamic programming approaches for the vehicle routing problem with stochastic demands*. Houston, TX: Ph.D. thesis. University of Houston.

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.

Stewart, W., & Golden, B. (1983). Stochastic vehicle routing: a comprehensive approach. *European Journal of Operational Research, 14*(4), 371–385.

Toth, P., & Vigo, D. (Eds.). (2014). *Vehicle routing: problems, methods, and applications* (2nd). Philadelphia, PA: Society for Industrial and Applied Mathematics.

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

Yee, J. R., & Golden, B. L. (1980). A note on determining operating strategies for probabilistic vehicle routing. *Naval Research Logistics Quarterly, 27*(1), 159–163.