Single vehicle routing with stochastic demands: approximate dynamic programming
Zhang, C.; Dellaert, N.P.; Zhao, L.; van Woensel, T.; Sever, D.

Published: 01/01/2013

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. Jan. 2019
Single Vehicle Routing with Stochastic Demands: Approximate Dynamic Programming

C. Zhang, N.P. Dellaert, L. Zhao, T. Van Woensel, D. Sever

Beta Working Paper series 425

<table>
<thead>
<tr>
<th>BETA publicatie</th>
<th>WP 425 (working paper)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISBN</td>
<td></td>
</tr>
<tr>
<td>ISSN</td>
<td></td>
</tr>
<tr>
<td>NUR</td>
<td>804</td>
</tr>
<tr>
<td>Eindhoven</td>
<td>July 2013</td>
</tr>
</tbody>
</table>
Authors are encouraged to submit new papers to INFORMS journals by means of a style file template, which includes the journal title. However, use of a template does not certify that the paper has been accepted for publication in the named journal. INFORMS journal templates are for the exclusive purpose of submitting to an INFORMS journal and should not be used to distribute the papers in print or online or to submit the papers to another publication.

Single Vehicle Routing with Stochastic Demands: Approximate Dynamic Programming

C. Zhang
Department of Industrial Engineering, Tsinghua University, Beijing, China, Zhangchen10@mails.tsinghua.edu.cn

N.P. Dellaert
Technische Universiteit Eindhoven, 5600 MB, Eindhoven, The Netherlands, n.p.dellaert@tue.nl

L. Zhao
Department of Industrial Engineering, Tsinghua University, Beijing, China, lzhao@tsinghua.edu.cn

T. Van Woensel
Technische Universiteit Eindhoven, 5600 MB, Eindhoven, The Netherlands, t.v.woensel@tm.tue.nl

D. Sever
Technische Universiteit Eindhoven, 5600 MB, Eindhoven, The Netherlands, d.sever@tue.nl

This paper deals with the single vehicle routing problem with stochastic demands (VRPSD). We formulate a stochastic dynamic programming model and implement Approximate Dynamic Programming (ADP) algorithms to overcome the curses of dimensionality. The developed ADP algorithms are based on Value Function Approximations (VFA) with lookup table representation. The standard VFA algorithm is extended and improved for the VRPSD. In the improved VFA algorithm (VFA+), we consider a Q-learning algorithm with bounded lookup tables and efficient maintenance. The VFA+ reduces the computational time significantly and still delivers high quality solutions. The significant reduction in computational time enables solving larger scale instances, which is important for real-life decision making. Test instances found in the literature are used to validate and benchmark our obtained results.

Key words: stochastic vehicle routing problem; approximate dynamic programming; value function approximation; Q-learning; rollout
1. Introduction

Consider a vehicle starting at a central bank and filling up ATMs at different places. For security reasons, it is not allowed to carry a large amount of money. Consequently, the vehicle is forced to make several short tours during its operating period (e.g., a working day) going back and forth to the central bank. Moreover, the needed cash in the ATMs is not known beforehand. Other similar examples of this described problem family in real-life are: beer distribution to retail outlets, the re-supply of baked goods at food stores, replenishment of liquid gas at research laboratories, stocking of vending machines (Yang et al. 2000), local deposit collection from bank branches, less-than-truckload package collection, garbage collection, home heating oil delivery, and forklift routing (Ak and Erera 2007).

The above described problem is related to the well-known Vehicle Routing Problem (VRP). The standard deterministic VRP is described extensively in the literature (see e.g., Laporte 2007). In contrast, this paper studies a single vehicle routing problem where stochastic demand is incurred (denoted as the VRPSD), very similar to Secomandi (2001). In general, this problem is similar to the standard vehicle routing problem where the aim is to construct a set of shortest routes for a fleet of fixed capacity. In the stochastic case, however, each customer has a given and known demand distribution and the actual demand realization is unknown until the vehicle arrives at the customer, when the customer’s actual demand is observed. In the VRPSD, the vehicle may be unable to satisfy the actual customer’s demand realization when visiting the customer (denoted as a failure). As such, the vehicle needs to return to the depot for a refill and return back to the partially served customer. In general, the vehicle serves every customer once unless a failure occurs, in which case a detour-to-depot is executed.

In this paper, we formulate the VRPSD using a stochastic dynamic programming model. Dynamic programming (DP) provides an elegant framework to model stochastic optimization problems. However, DP faces the well-known three curses of dimensionality (states, outcomes, and decisions) and cannot deal with practical size problems. In addition, the single vehicle routing problem with stochastic demands is a difficult and computationally demanding problem. Over the past years, computing power has increased dramatically, giving a sound basis to efficiently handle stochastic and dynamic vehicle routing problems. Our paper employs an Approximate Dynamic Programming (ADP) strategy. ADP emerges as an efficient and effective tool in solving large scale stochastic optimization problems, combining the flexibility of simulation with the intelligence of optimization. It is a powerful approach to model and solve problems which are large, complex and
with stochastic and/or dynamic elements (Powell 2007). Referring to Pillac et al. (2012), ADP successfully solves large-scale freight transport and fleet management problems while coping with the scalability problems of DP (see also e.g., Godfrey and Powell 2002, Powell et al. 2002, Powell and Van Roy 2004, Simão et al. 2009).

In this paper, we develop ADP algorithms based on Value Function Approximations (VFA) with lookup table representation. We first design a standard VFA algorithm by using the ADP framework. To achieve good computational performance and solution quality, several adaptations are needed. Using post-decision state variables in ADP allows making decisions without having to compute the expectation. However, for the VRPSD, post-decision state variables omit important information about the current state and the decision. Therefore, we consider a Q-learning algorithm with lookup table representation in which we store the state-decision pairs and their values (i.e., Q-factors). However, the size of a standard lookup table increases exponentially (as it depends on both the state and decision). For this reason, we improve the standard Q-learning algorithm with bounded lookup tables and efficient maintenance strategies. We also design effective exploration/exploitation strategies such that we obtain higher quality solutions with lower computational time, as compared to the rollout algorithm in Secomandi (2001). We denote this improved VFA algorithm as VFA$^+$. 

The contribution of this paper to the literature is as follows. First, we formulate the vehicle routing problem with stochastic demands using a unified stochastic dynamic programming modeling framework (Powell 2007, 2011, Powell et al. 2012), which allows for flexible extensions of the problem in real-life applications. Second, we design efficient ADP algorithms that allow us to solve large scale problems in reasonable time. The standard VFA with lookup table representation is improved using a Q-learning algorithm with bounded lookup tables and efficient maintenance. Our algorithm comparisons on test instances from the literature (Secomandi 2001, Solomon 1987) show that, for small size test instances, the VFA$^+$ algorithm on average covers more than 50% of the performance gap between the Rollout and the optimal solution; for large size test instances, VFA$^+$ consistently outperforms the Rollout algorithm with better solution quality and less computational time. The significant reduction in computational time enables solving larger scale instances, which is important for real-life decision making. Further, we analyze the effect of the depot location on the relative performances of the algorithms. Last, this paper provides important insights on applying ADP to deal with stochastic and combinatorial problems such as VRPSD, using bounded lookup tables with efficient maintenance and exploration-and-exploitation strategies.
The paper is structured as follows. The literature review is given in Section 2. The problem formulation is presented in Section 3, and the ADP algorithms are described in Section 4. The experiment design and numerical results are given in Sections 5 and 6. Section 7 concludes this paper.

2. Literature review

Stochastic vehicle routing problems are characterized by some random elements in their problem definition (Gendreau et al. 1996a). In the literature, researchers consider stochastic demands (see e.g., Bertsimas 1992, Dror et al. 1993), stochastic customers (see e.g., Bent and Van Hentenryck 2004), stochastic demand and customers (see e.g., Gendreau et al. 1995, Gendreau et al. 1996b) and stochastic travel times (see e.g., Laporte et al. 1992, Kenyon and Morton 2003). Gendreau et al. (1996a) review the literature on stochastic VRPs and their different flavours.

There are a number of papers closely related to our paper. Secomandi (2000, 2001) deals with the VRPSD, considering detour-to-depot schemes and allowing for early replenishment. Secomandi (2000) presents a stochastic shortest path problem formulation based on a Markov Decision Process (MDP), and develops two heuristics: a rollout algorithm and an approximate policy iteration. Secomandi (2001) gives more details on the rollout algorithm, which uses a nearest insertion and a 2-int heuristic as its base sequence, and a cyclic heuristic to generate new partial routes. According to the author, the rollout policy is the first computationally tractable algorithm for approximately solving the problem under the re-optimization approach. Novoa and Storer (2009) extend the rollout algorithm by implementing different base sequences, two-step look-ahead policies and pruning schemes. Goodson et al. (2013) present a rollout policy framework for general stochastic dynamic programs and apply the framework to solve for VRPs with stochastic demands and duration limits. The base sequence uses local search utilizing a relocation neighborhood and a first-improving search criteria, as well as a combined pre- and post-decision state heuristic. Furthermore, the algorithm is enhanced by a problem-specific dynamic decomposition scheme.

This paper can also be situated in the family of re-optimization algorithms for the VRP with stochastic demands. Dror et al. (1989, 1993) are the early papers that introduce the re-optimization strategies. They optimally re-sequence the unvisited customers whenever a vehicle arrives at a customer and observes the demand. Secomandi and Margot (2009) also considers a VRPSD under re-optimization. They formulate the problem in terms of a finite horizon MDP for the single vehicle
case. They develop a partial re-optimization methodology to compute suboptimal re-optimization policies for the problem. In this methodology, they select a set of states for the MDP by using two heuristics: the partitioning heuristic and the sliding heuristic. They compute an optimal policy on this restricted set of states by a backward dynamic programming.

In the VRPSD literature, there are a number of papers dealing with developing an optimal restocking policy with a predefined customer sequence. Yang et al. (2000) study strategies of planning preventive returns to the depot at strategic points along the vehicle routes. They prove that for each customer, there exists a threshold number such that the optimal decision is to continue to next customer if the remaining load is greater than or equal to the threshold number or otherwise to return to the depot for replenishment.

Tatarakis and Minis (2009) study the multi-product delivery routing with stochastic demands. They develop a dynamic programming algorithm to solve a compartmentalized case of multi-product delivery to derive the optimal policy in a reasonable amount of time. Minis and Tatarakis (2011) extend the problem to a pickup and delivery case of the VRPSD. They provide an algorithm to determine the minimum expected routing cost and a policy to make the optimal decisions including the detour-to-depot decisions for the stock replenishment. Recently, Pandelis et al. (2012) prove the optimal structure of the same problem for any positive number of multiple products.

The VRPSD is also studied with additional constraints. Erera et al. (2010) consider VRP with stochastic demands and constraints on the travel time durations of the tours. The authors define and study various restocking detour policies in the paper. Lei et al. (2011) study the VRP problem with stochastic customers with time windows. The problem is modeled using stochastic programming with recourse and the solution strategy is proposed as a large neighborhood search heuristic.

In this paper, we model and solve the VRPSD using an ADP framework. Powell (2007, 2011) provide a comprehensive introduction to the basic ideas of ADP and address key algorithmic issues when designing ADP algorithms. For the dynamic VRP problems, a unified framework is presented in Powell et al. (2012) where various polices including the ADP approach are explained.

3. Problem description and model formulation

We study a single vehicle routing problem with stochastic customer demands (VRPSD). On an undirected graph $G = (V, E)$, $V = \{0, \ldots, N\}$ is the vertex set and $E$ is the edge set. Vertex 0
denotes the depot, whereas vertices \( i = 1, \ldots, N \) denote the customers to be served. A nonnegative distance \( d_{ij} \) is associated with each edge \((i, j) \in E\), representing the travel distance (or cost, time, etc.) between vertices \( i \) and \( j \).

A single vehicle with full capacity \( Q \) starts from the depot, serves all the customers to perform only deliveries (or only pick-ups), and returns to the depot at the end of the tour. Each customer \( i \in V \) is associated with a stochastic demand \( D_i \), the true value of which is revealed upon the arrival of the vehicle at the customer. If the vehicle does not have sufficient capacity to serve a customer (a “failure” occurs), it partially serves the customer, returns to the depot to replenish, and comes back to the customer to fulfill the remaining demand. The vehicle then continues its tour to the next customer or the depot (at the end of the tour). The objective is to minimize the expected total travel distance. We assume that the depot has plenty quantity of the commodity and the maximum possible demand of each customer is smaller than the vehicle capacity.

If early replenishment is allowed, the vehicle can return to the depot to replenish before encountering a failure. In the offline planning version of the problem, the vehicle always follows a predetermined order to visit the customers, while in the online planning version, the vehicle is allowed to re-route (re-optimize) after serving each customer. In this paper, our focus is on the online planning problem with early replenishment (as in Secomandi 2000, 2001).

Next, we formulate the problem as a stochastic dynamic program, using the notation as described in Powell (2007, 2011).

The problem is divided into \( t = 0, 1, \ldots, N, N + 1 \) stages. \( t = 0 \) represents the start of the tour at the depot, \( t = N + 1 \) represents the end of the tour back to the depot, and \( t = 1, \ldots, N \) represents the number of customers that have been visited during the tour.

**State**

The state variable is defined as:

\[
S_t = (i_t, l_t, J_t), \quad t = \{0, 1, \ldots, N, N + 1\},
\]  
(1)
where
describes the current customer being served (or the depot when \( t = 0 \) and \( t = N + 1 \));

\( l_t \): the current capacity in the vehicle after serving the current customer \((0 \leq l_t \leq Q)\);

\( J_t \): the vector \((j_{t,1}, \ldots, j_{t,N})\) that represents the customers’ service status: \( j_{t,i} = 1 \), if
customer \( i \) has already been served; \( j_{t,i} = 0 \), otherwise.

Therefore, when the vehicle starts at the depot, the initial state is \( S_0 = (0, Q, 0, \ldots, 0) \), and
when the vehicle returns to the depot after serving all the customers, the final state becomes
\( S_{N+1} = (N+1, l_{N+1}, 1, \ldots, 1) \). Note that \( l_{N+1} = l_N \) is the vehicle’s remaining capacity after serving
the last customer.

**Decision variables**

After serving the current customer, two types of decisions are to be made: which customer to serve
next and whether to return to the depot before visiting the next customer. The decision variables
are defined as:

\[
x_t = (i_{t+1}, r_t), \quad t = \{0, 1, \ldots, N\},
\]

where

\( i_{t+1} \): the next customer to be served;

\( r_t \): \( r_t = 1 \) indicates returning to the depot before visiting the next customer; \( r_t = 0 \)
otherwise.

Further, we define \( X_t^*(S_t) \) as the decision function that determines decision \( x_t \) at stage \( t \) under
policy \( \pi \), given state \( S_t \). Each \( \pi \in \Pi \) refers to a different policy and \( \Pi \) denotes the set of all
implementable policies.

**Exogenous information**

The customer demand \( D_{i_{t+1}} \) has a customer specific discrete distribution. The actual customer
demand \( \hat{D}_{i_{t+1}} \) is only revealed after the vehicle arrives at customer \( i_{t+1} \).

\[
W_{t+1} = \hat{D}_{i_{t+1}}, \quad t = \{0, 1, \ldots, N - 1\}.
\]
State transition function

Given the current state \( S_t = (i_t, l_t, J_t) \), the decision \( x_t = (i_{t+1}, r_t) \), and the exogenous information \( W_{t+1} = \hat{D}_{t+1} \), the state transition function is defined as, for \( t = \{0, 1, \ldots, N\} \),

\[
S_{t+1} = S^M(S_t, x_t, W_{t+1})
\]

\[
= \begin{cases} 
(i_{t+1}, Q - \hat{D}_{t+1}, J_{t+1}) , & \text{if } r_t = 1, \\
(i_{t+1}, l_t - \hat{D}_{t+1}, J_{t+1}) , & \text{if } r_t = 0 \text{ and } l_t \geq \hat{D}_{t+1}, \\
(i_{t+1}, l_t + Q - \hat{D}_{t+1}, J_{t+1}) , & \text{if } r_t = 0 \text{ and } l_t < \hat{D}_{t+1},
\end{cases} \tag{4}
\]

where the service status vector \( J_{t+1} \) is updated as: \( j_{t+1,i} = 1 \), if \( i = i_{t+1} \); \( j_{t+1,i} = j_{t,i} \), otherwise. That is, the service status of customer \( i_{t+1} \) is changed to 1 (being served) and the service statuses of other customers remain unchanged. Note the letter “\( M \)” in the first equation of Equation (4) represents “model” as in Powell (2007, 2011).

Cost function

The vehicle’s actual travel distance or cost depends on both the decision and realized demand at the next customer \( W_{t+1} = \hat{D}_{t+1} \). Therefore, the (expected) cost function \( c_t(S_t, x_t) \) can be decomposed into a deterministic and a stochastic parts, as below.

\[
c_t(S_t, x_t) = C_t(S_t, x_t) + E[\Delta C_{t+1}(S_t, x_t, W_{t+1})], \tag{5}
\]

where

\[
C_t(S_t, x_t) = \begin{cases} 
d_{i_{t},0} + d_{0,i_{t+1}}, & \text{if } r_t = 1, \\
d_{i_{t},i_{t+1}}, & \text{if } r_t = 0, 
\end{cases} \tag{6}
\]
The calculations of (6) and (7) follow the same logic as in the state transition function (4).

**Objective function**

The objective of the stochastic dynamic program is to find the optimal policy \( \pi \in \Pi \) to minimize the expected total cost (travel distance) to serve all the customers, that is,

\[
\min_{\pi \in \Pi} \sum_{t=0}^{N} c_t(S_t, x_t)
\]

\[
= \min_{\pi \in \Pi} \sum_{t=0}^{N} c_t(S_t, X_\pi^t(S_t)),
\]

where \( x_t = X_\pi^t(S_t) \) is the decision made according to the decision function \( X_\pi^t(S_t) \) under policy \( \pi \), given the current state \( S_t \). Note that the expectation is embedded in the calculation of the cost function \( c_t(S_t, x_t) \).

4. **Approximate Dynamic Programming**

If the state, decision, and outcome spaces are finite discrete, the stochastic dynamic program (8) can be solved recursively using Bellman’s equations,

\[
V_t(S_t) = \min_{x_t \in X_t} \left( C_t(S_t, x_t) + E[V_{t+1}(S_{t+1})] \right).
\]

The value function \( V_t(S_t) \) specifies the value of being in a state \( S_t \), in which \( C_t(S_t, x_t) \) accounts for the immediate cost associated with the current state \( S_t \) and decision \( x_t \), while the value function \( V_{t+1}(S_{t+1}) = V_{t+1}(S_{t+1}, S_t, x_t, W_{t+1}) \) evaluates the future impact of the decision \( x_t \) under the realized exogenous information \( W_{t+1} \).

To overcome the three curses of dimensionality (states, decisions, and outcomes) associated with the classical DP approach, in *approximate dynamic programming* (ADP), we replace the exact
value function $V_{t+1}(\cdot)$ in Equation (9) with an approximation $\bar{V}_{t+1}(\cdot)$ as in Equation (10). Instead of the exact evaluation of $V_{t+1}(\cdot)$ often in a backward manner, $\bar{V}_{t+1}(\cdot)$ can be evaluated via step-forward simulation, by integrating a variety of rich classes of stochastic optimization and simulation methodologies.

$$V_t(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + E[\bar{V}_{t+1}(S_{t+1})]).$$

(10)

While the approximate value function $\bar{V}_{t+1}(\cdot)$ can take a variety of forms (such as weighted sum of basis functions, piecewise linear functions, regression models, neural networks), the lookup table representation is a generic model-free form that is often used when the value function structure can hardly be clearly defined, which is the case of the VRPSD under study. In this section, we first introduce a generic value function approximation (VFA) with lookup table representation, and then describe an improved version (VFA$^+$), which addresses the problem characteristics of the VRPSD.

4.1. Value Function Approximation Algorithm (VFA)

Algorithm 4.1 depicts a generic value function approximation approach with lookup table representation. In Step 0a, we use the Rollout algorithm (Secomandi 2001) as the heuristic to initialize the state values.

Striking a good balance between exploration and exploitation remains an important and cutting-edge research question in ADP and other related research areas such as simulation optimization and machine learning. In our VFA algorithm, we use the fixed exploration rate strategy. That is, with probability $\rho$ (for example, 0.10), we explore the impact of a randomly selected decision; otherwise, we stick to the optimal decision based on the current value function (Step 2a). In Section 4.2, we describe an improvement on the exploration and exploitation strategy using preventive returns and restocking.

In step 2b, we use the exponential smoothing function to update the value function approximation $\bar{V}_t(S_t)$ with the observed value $\hat{v}_t(S_t)$. That is,

$$\bar{V}_n^t(S_t) = (1 - \alpha_{n-1})\bar{V}_{n-1}^t(S_t) + \alpha_{n-1} \hat{v}_t^n,$$

(11)

where $\alpha_{n-1}$ is the stepsize.
Algorithm 4.1 A generic VFA approach with lookup table representation.

Step 0. Initialization.

Step 0a. Initialize $\bar{V}^0_t(S_t)$ for all states $S_t$.

Step 0b. Choose an initial state $S_0^i$.

Step 0c. Set the iteration counter $n = 1$.

Step 1. Choose a sample path $\omega^n$.

Step 2. For $t = 0, 1, \ldots, N$, do:

Step 2a. If exploitation, solve

$$\hat{v}^n_t = \min_{x_t \in X_t}(C_t(S_t, x_t) + \mathbb{E}[\bar{V}^{n-1}_{t+1}(S_{t+1})]),$$  

and let $x^n_t$ be the solution.

If exploration, randomly choose a solution $x^n_t \in X_t$.

Step 2b. Update $\bar{V}^n_t(S_t)$ using

$$\bar{V}^n_t(S_t) = \begin{cases} (1 - \alpha_{n-1})\bar{V}^{n-1}_t(S^n_t) + \alpha_{n-1}\hat{v}^n_t, & \text{if } S_t = S^n_t, \\ \bar{V}^{n-1}_t(S_t), & \text{otherwise.} \end{cases}$$

Step 2c. State transition.

$$S_{t+1}^i = S^M(S^n_t, x^n_t, W_{t+1}(\omega^n)).$$

Step 3. Let $n = n + 1$. If $n \leq N$, go to step 1.

Note that $N$ denotes the pre-set maximum number of iterations.

Step 4. Output the value function, $\{\bar{V}^N_t(S^x_t)\}_{t=0}^{N-1}$.

For calculating $\alpha_{n-1}$, we apply the Bias-adjusted Kalman Filter (BAKF) stepsize rule (George and Powell 2006, Powell 2007), which is given by

$$\alpha_{n-1} = 1 - \frac{\hat{\sigma}^2^n}{(1 + \lambda^{n-1})\hat{\sigma}^2 + (\hat{\beta}^n)^2}. \quad (12)$$

$\hat{\sigma}^2^n$ denotes the estimate of the variance of the value function $\bar{V}^n_t(S^n_t)$ and $\hat{\beta}^n$ denotes the estimate of bias due to smoothing a nonstationary data series. The BAKF stepsize rule adaptively balances the estimate of the noise $\hat{\sigma}^2^n$ and the estimate of the bias $\hat{\beta}^n$ that is attributable to the transient nature of the data in the ADP solution process. We refer to George and Powell (2006) and Section 6.5.3 in Powell (2007) for more details.

In Equation (10), the approximate value function $\tilde{V}_t(S_t)$ is associated with the pre-decision state $S_t$. Solving for Equation ($\ast$) in Step 2a requires the calculation of the expected value of $\bar{V}_{t+1}(S_{t+1})$ within the min operator, which is computationally demanding. To improve the computational efficiency in ADP, Powell (2007) introduces the notion of post-decision state $S^n_t = S^{M,x}(S_t, x_t)$.
which captures the state of the system immediately after the decision making, but before new information arrives. With the approximate value function around post-decision state $\bar{V}_t^{n-1}(S'_t)$, we can solve for $\hat{v}_t^n(S_t)$ in Step 2a with

$$
\hat{v}_t^n(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \bar{V}_t^{n-1}(S'_t)),
$$

which avoids the expectation within the min operator, but normally requires more effort in estimating $\bar{V}_t^{n-1}(S'_t)$.

In VRPSD, the post-decision state omits certain critical information. For example, besides knowing the next customer to visit, the actual cost or travel distance depends on both the realized demand of the next customer $\hat{D}_{i_{t+1}}$ and the current capacity $l_t$, which can vary significantly (refer to equation (7)). Further, the tradeoff between traveling directly to the next customer or detour-to-depot also depends on the relative distances between the current customer and the next customer or depot, respectively. Consequently, we somehow lose the “memoryless” property in VRPSD. Our numerical experiments also show that the approximate value function around post-decision state does not work very well, which confirms our observation.

Q-learning is another algorithm that has certain similarities to DP using the value function around post-decision states. In Q-learning, a Q-factor, $Q_t(S_t, x_t)$, stores the value of a state-decision pair and it captures the value of being in a state and taking a particular decision (Bertsekas and Tsitsiklis 1995, Sutton and Barto 1998, Powell 2007, Bertsekas 2012). However, a potential problem with Q-learning is that the size of the lookup table increases exponentially because it depends on both the state and the decision. In the next section, we describe how to better utilize the lookup tables with Q-factors via efficient maintenance and exploration/exploitation strategies in our improved algorithm (VFA$^+$.)

4.2. Improved Value Function Approximation Algorithm (VFA$^+$)

The value function approximation (VFA) with lookup table representation as described in Algorithm 4.1 is a generic ADP approach. As previously mentioned, an alternative way to store the value function is to use Q-factors. Q-factors, $Q_t(S_t, x_t)$, store the value of a state-decision pair in the lookup table. The state-decision pair at stage $t$ is: $(S_t, x_t) = ((i_t, l_t, J_t), (i_{t+1}, r_t))$. We should note that the decision consists of the next customer to visit at stage $t+1$ and whether to return to the depot before visiting the next customer.
For the VRPSD, we improve the computational performance and the solution quality of the standard VFA algorithm with Q-learning by considering the approximate values of the Q-factors. We define $\bar{Q}^n_t(S_t, x_t)$ as the approximate value of $Q_t(S_t, x_t)$ after $n$ iterations. We use a double pass algorithm to update $\bar{Q}_t(S_t, x_t)$, as shown in Algorithm 4.2. At each iteration, we first find a customer sequence based on the values of the state-decision pairs (Q-factors) from the past iterations in the forward pass. Then, we update the Q-factors using the realized cost in the backward pass.

Algorithm 4.2 Q-learning approach for the VFA$^+$ algorithm with double pass

Step 0. Initialization.

Step 0a. Initialize $\bar{Q}^0_t(S_t, x_t)$ for all states $S_t$ and decisions $x_t \in X_t$.

Step 0b. Choose an initial state ($S^0_0$).

Step 0c. Set the iteration counter $n = 1$.

Step 1. Choose a sample path $\omega^n$.

Step 2. (Forward pass) For $t = 0, 1, \ldots, N$, do:

Step 2a. If exploitation, find

$$x^n_t = \arg\min_{x_t \in X_t} (\bar{Q}^{n-1}_t(S_t, x_t) + SD(S_t, S^n_{t+1})).$$

If exploration, choose a solution $x^n_t \in X_t$

based on the exploration-and-exploitation strategies described in the paper.

Step 2b. Compute the next state $S^n_{t+1} = SM(S^n_t, x^n_t, W_{t+1}(\omega^n))$.

Step 3. (Backward pass) Set $\hat{q}^0_{N+1} = 0$ and do for all $t = N, N - 1, \ldots, 0$:

Step 3a. Calculate:

$$\hat{q}^n_t = C_t(S^n_t, x^n_t) + \hat{q}^n_{t+1}$$

Step 3b. Update $\bar{Q}^n_t(S_t, x^n_t)$ using

$$\bar{Q}^n_t(S^n_t, x^n_t) = (1 - \alpha_{n-1})\bar{Q}^{n-1}_t(S^n_t, x^n_t) + \alpha_{n-1}\hat{q}^n_t.$$

Step 4. Let $n = n + 1$. If $n \leq \textbf{N}$, go to step 1.

Note that $\textbf{N}$ denotes the pre-set maximum number of iterations.

Step 5. Return the Q-factors, $\{\bar{Q}^N_t\}_{t=0}^{N-1}$.

As the size of the lookup table with Q-factors grows exponentially with both state and decision, we limit the number of stored Q-factors in the lookup table (bounded lookup table). At each iteration, we mostly visit the state-decision pairs that are already in the lookup table. Consequently, the values of the state-decision pairs (Q-factors) in the lookup table are more frequently updated, and therefore more accurate.
In Step 2a of Algorithm 4.2, one important notion is the state difference, \( SD(S_t, S'_t) \). Due to the limited size of the bounded lookup table, we may frequently come into states that are not contained in the lookup table. In this case, we look at the most similar states. To determine these most similar states, we define the state difference \( SD \) between two states with identical \( i_t \) value as (where \( c_1 \) is a constant):

\[
SD(S_t, S'_t) = \sum |j_t - j'_t| + c_1 \times |l_t - l'_t|
\]  

(14)

As the number of states-decision pairs grows with the vehicle capacity and with the number of customers, then, we consider a bounded lookup table with a subset of state-decision pairs. When we arrive at a state-decision pair that is not contained in the bounded lookup table, we identify the “nearest” state-decision pair according to Equation (14). Accordingly, we update the value of the nearest state-decision pair, already in the bounded lookup table with the minimum state difference, instead of the state-decision pair that is not in the bounded lookup table.

Further, the exponential growth in the state and decision space in the VRPSD forces us to find a good balance between exploration and exploitation. In VFA\(^+\), the exploitation is obtained by focusing on a limited number of state-decision pairs, such that good cost estimates can be found by frequent visits to these pairs. The exploration is obtained by using a variety of randomized heuristics for our travel decisions.

Further, we improve the performance in terms of solution quality and computational time by considering the following algorithmic strategies:

1. Use different initialization heuristics;
2. Organize and maintain the bounded lookup table;
3. Explore and exploit using preventive returns and restocking.

We give more details on the algorithmic strategies below.

**Use different initialization heuristics**

We use a mix of three simple initialization heuristics. The first one is based on the cyclic tours that Secomandi (2001) derives in his a priori approach. By considering all the possible customers being visited first in the route, we obtain Q-factor values associated with each customer visited at stage \( t \), i.e., \( i_t \). The second heuristic is a randomized nearest neighbor heuristic, where early replenishments are only done when this gives an immediate advantage. In the third heuristic, we
use a variation of the *cone covering* method, introduced by Fisher and Jaikumar (1981) and then applied by Fan et al. (2006) to a similar problem. The advantage of the third heuristic lies in that it considers both the geographic location and the expected demand of each customer, thus generates routes with approximately the same expected total demand.

For the initialization of the Q-factors, we first cluster the customers into a few groups (depending on the expected number of replenishments). Then, we apply the nearest neighbor heuristic to the clusters to create one tour per group. Finally, we create one tour for all customers by applying a savings algorithm to the customers linked to the depot, until there are no intermediate visits to the depot in the tour. For the resulting customer sequence, we determine the optimal replenishment visits similar to Secomandi (2001). In the clustering, we apply randomization to get a larger solution set. For each of the heuristics, we find 600 sample paths to create an initial set of state-decision pairs and their values (Q-factors) in the lookup table. For state-decision pairs that are visited multiple times, we take the minimum of the Q-factors.

**Organize and maintain the bounded lookup table**

We denote the set of state-decision pairs that have the same $i_t$ as an “$i_t$ set”, i.e., $\{(S_t, x_t) | S_t = (i_t, \cdot, \cdot)\}$, and let $|i_t|$ denote its size. That is, an $i_t$ set consists of all possible combinations of the state-decision variables: $(i_t, L_t, J_t)$ and the decision: $(i_{t+1}, R_t)$ with the same $i_t$. We provide a detailed illustration of $i_t$ sets in the Appendix. To control the exponential growth of the lookup table, we limit the number of stored Q-factors in each $i_t$ set to a maximum (denoted as $|i_t|_{\text{max}}$).

We use a three-level pruning to maintain the bounded lookup table. Pruning Procedure I examines and maintains the size of each $i_t$ set at each iteration, while Pruning Procedures II and III provide additional maintenance and value update of the $i_t$ set every fixed number of iterations.

**Pruning Procedure I:**

When the size of an $i_t$ set reaches $|i_t|_{\text{max}}$, we perform the pruning procedure I (Algorithm 4.3). We remove the state-decision pairs that have been visited only once. In addition, we remove the state-decision pairs that have a value higher than the average value and also a number of visits less than the average number of visits in the $i_t$ set. Usually, about half of the state-decision pairs is removed in this procedure. Note that, apart from the Q-factor, we also record the number of visits
to each state-decision pair, \( n'(S_t, x_t) \), which is used in Pruning Procedure I as well as in Equation (15) of the value updating procedure.

Based on our numerical evaluation, we set \( |i_t|_{max} \) as 150 state-decision pairs for instances with less than 20 customers and 250 state-decision pairs for larger instances. Higher \( |i_t|_{max} \) values lead not only to longer computational time but also to poorer results, because the most relevant Q-factors will be updated less often.

**Algorithm 4.3 Pruning Procedure I**

1. **Step 1.** Check the size of the \( i_t \) set.
2. **Step 2.** Keep the state-decision pairs in the \( i_t \) set, if \( |i_t| < |i_t|_{max} \).
3. **Step 3.** If the size of the \( i_t \) set reaches its maximum (\( |i_t| \geq |i_t|_{max} \)), remove the state-decision pairs from the \( i_t \) set under the following criteria:
   
   **Step 3a.** If their value is greater than the average Q-factor value of the \( i_t \) set, that is,
   \[
   Q(S_t, x_t) \geq \frac{\sum_{(S_t, x_t) | S_t = (i_t, \cdot, \cdot)} Q(S_t, x_t)}{|i_t|};
   \]
   and if also their number of visits is less than the average number of visits of the \( i_t \) set, that is,
   \[
   n'(S_t, x_t) \leq \frac{\sum_{(S_t, x_t) | S_t = (i_t, \cdot, \cdot)} n'(S_t, x_t)}{|i_t|}.
   \]

**Pruning Procedures II and III:**

Different from Pruning Procedure I, Pruning Procedures II and III limit the number of potential next customers in the \( i_t \) set. These two procedures are implemented every fixed number of iterations.

In Pruning Procedure II (Algorithm 4.4), we look at the decision on the next customer to visit \((i_{t+1})\) among the state-decision pairs in the \( i_t \) set. For each \( i_{t+1} \), we calculate the average Q-factor value of the state-decision pairs with \( x_t = (i_{t+1}, \cdot, \cdot) \), denoted as \( \bar{Q}_{i_{t+1}} \). We only keep the state-decision pairs whose \( \bar{Q}_{i_{t+1}} \) are among the best \( k_t \) potential next customers. The number \( k_t \) depends on the stage \( t \), and decreases during the ADP, to gradually focus more on the best decisions on the potential next customer to visit.

Pruning Procedure III uses the double pass DP approach, and at the same time updates the Q-factors. In this procedure, we update the Q-factors in the backward pass while pruning the lookup table in the forward pass.
Algorithm 4.4 Pruning Procedures II

Step 1. For each \( i_{t+1} \) of the state-decision pairs in the \( i_t \) set, calculate the average Q-factor

\[
\bar{Q}_{i_{t+1}} = \frac{\sum_{(S_t, x_t)} Q(S_t, x_t)}{n(i_t, i_{t+1})},
\]

where \( n(i_t, i_{t+1}) \) denotes the number of state-decision pairs in the \( i_t \) set with the same \( i_{t+1} \).

Step 2. Sort the customers, \( i_{t+1} \), according to an ascending order of \( \bar{Q}_{i_{t+1}} \).

Step 3. Select the first \( k_t \) customers and remove all the state-decision pairs \((S_t, x_t)\) whose next customer, \( i_{t+1} \), are not among these \( k_t \) customers.

When we have a sample path, we only update the values of the state-decision pairs that we actually visit as in Algorithm 4.2. To obtain better estimates, we also update the values of the state-decision pairs that use a part of the sample path. To achieve this, we perform an update process every fixed number of iterations by using a backward DP algorithm for all state-decision pairs in the bounded lookup table. For the update process, we start from the last stage and compute the minimum expected cost for each state-decision pair in the bounded lookup table. The values of the state-decision pairs, including the ones using a part of the sample path, are updated using the minimum expected cost, and recursively, the updated values are used for the update of the Q-factors in previous stages.

After the backward DP, we remove the state-decision pairs with the same next customer, \( i_{t+1} \) that never give the minimum expected value, \( Q_{\min}(S_t) \), in the backward DP procedure. The update and the pruning procedure is described in Algorithm 4.5.

Algorithm 4.5 Pruning Procedures III

Step 1. (Backward DP) For all \( t = N, N - 1, \ldots, 0 \):

Step 1a. Find the minimum state value among the possible next customers:

\[
Q_{\min}(S_t) = \min_{i_{t+1}} \left[ \mathbb{E}[C_t(S_t, x_t)] + \min_{x_{t+1} \in X_{t+1}} (SD(S_{t+1}, S'_{t+1}) + Q(S_{t+1}, x_{t+1})) \right].
\]

Step 1b. Update all \( Q_t(S_t, x_t) \) in the lookup table with the decision \( x_t \) using:

\[
Q_t(S_t, x_t) = (1 - \alpha_{n-1})Q_t(S_t, x_t) + \alpha_{n-1}Q_{\min}(S_t).
\]

Step 2. (Forward DP) Pruning: do for all \( t = 0, 1, \ldots, N + 1 \):

Remove the state-decision pairs with the same \( i_{t+1} \) that are never qualified for \( Q_{\min}(S_t) \).

Updating Procedure:
In every sample path simulation, we update the value of the visited state-decision pairs using the harmonic stepsize rule.

\[
\alpha_{n-1} = \frac{a}{a + n'(S_t, x_t) - 1},
\]

where \(a\) is a constant. Note that the stepsize \(\alpha_{n-1}\) depends on the number of visits to the state-decision pair, \(n'(S_t, x_t)\), rather than the iteration counter \(n\).

In VFA\(^+\), the value of \(a\) depends on the decision. If all decisions taken after the visit of a state-decision pair in the sample path are optimal (exploitation decisions), we assign a high value to \(a\). If however after the visit of a state-decision pair, we take an exploration decision, we either assign a low value to \(a\) (in case of improvement) or set \(a\) equal to zero (in case of non-improvement). In this way, we avoid that the state-decision pair becomes unattractive by not considering the best route afterwards.

**Exploration and exploitation using preventive returns and restocking**

In each step of the ADP algorithm, we may select the best decision based on the current Q-factors in the bounded lookup table (exploitation), or we may select an alternative decision in order to discover potentially better decisions (exploration). In VFA\(^+\), the exploitation options are:

- the decision with the best Q-factor value in the lookup table (and perfect match for remaining customers and capacity),
- the decision with the lowest sum of the Q-factor and the state difference in the lookup table.

The exploration options are:

- the decision with the second best Q-factor value (as this is the most promising alternative),
- the decision where the next customer is randomly selected from the lookup table, combined with a randomized early replenishment decision \(r_t\),
- the decision where the next customer, \(i_{t+1}\) is randomly selected from the set of \(m_t\) nearest (in distance) unvisited customers, combined with a randomized early replenishment decision \(r_t\). The value of \(m_t\) may be different for different stages.

The options above are considered with different probabilities. Note that once we have applied an exploration decision, the further decisions are preferably exploitation decisions, in order to obtain good cost estimates for this explorative decision.
At the end of the tour, when the vehicle capacity gets scarcer, it makes sense to return to the depot for early replenishment if the vehicle is close to the depot. Using the demand probability distribution of the non-visited customers, we determine the minimum expected number of remaining depot visits, both with and without returning to the depot. If the difference between the two values exceeds 0.9, we first return to the depot and then serve the remaining customers.

Parameter settings

The algorithmic strategies in VFA+ described above are all designed to improve the computational performance, but they also create a large number of parameter settings, such as probabilities in the sample path selection, updating parameters, parameters in the maintenance of the bounded lookup table, etc.

We conduct a number of preliminary tests to set the parameter settings to be used. Based on our preliminary computational evaluation, we fix some of the parameter settings and limit the possibilities for others to two or three options. For instance, the total duration of the ADP is fixed on 250,000 iterations; Pruning Procedures II and III are performed every 10,000 iterations; the value $a$ in the harmonic stepsize rule, Equation (15) is set to 0 (exploration without improvement), 1 (exploration with improvement), or 5 (exploitation). We use this setting for all instances and report the results in the paper.

5. Experimental design

In this section, we describe the test instances used in the numerical experiments and our methodology to evaluate the algorithms.

5.1. Test Instances

We use two sources of instance generation in the literature, from Secomandi (2001) and Solomon (1987). In both sets of test instances, we consider delivery to customers from the depot. We use the test instances of Secomandi (2001) to compare our value function approximation algorithms (VFA and VFA+) with the Rollout algorithm in Secomandi (2001). We also test the algorithms
with the Solomon instances which is a standard reference in the VRP literature. Two different sets of Solomon instances are generated to evaluate the effect of the depot location, which is either at the center or at the corner.

The first set of instances are based on Secomandi (2001). The instances are generated with different number of customers. Specifically, the instances with 5 to 19 customers are denoted as the “small size instances,” and the instances with 20, 30, . . ., 60 customers are denoted as the “large size instances.” The test instances also differ in the values of the expected fill rate \( \bar{f} = \sum_{i=1}^{N} E(D_i)/Q. \) \( \bar{f}' \equiv max\{0, \bar{f} - 1\} \) can be viewed as the expected number of route failures, and \( \bar{f}' \) is in the set \{0.75, 1.25, 1.75\}. Therefore, there are 3 variants for each small size and large size instances. The values of \( Q \) for all possible \((\bar{f}', N)\) pairs are computed by rounding \(3N/\bar{f}'\) to the nearest integer. For each instance, the customer demands are divided into low, medium, and high categories, following three discrete uniform probability distributions. Every customer is assigned to one of these demand categories with equal probability (1/3). For the small size instances \((N < 20)\), the demand categories are \(U(1,3), U(2,4), U(3,5)\), and for the large size instances \((N \geq 20)\), they are \(U(1,5), U(6,10), U(11,15)\). The depot is fixed at the corner. For each \((\bar{f}', N)\) pair, 10 replications are generated for each of the small size instances and 5 replications are generated for each of the large size instances. We refer to these instances as the “Secomandi instances” in the rest of the paper.

The Solomon instances are based on the RC instances \((RC101 to RC105 and RC201 to RC205)\) from Solomon (1987). As the number of the customers and the demand distributions in Solomon (1987) are not comparable to the Secomandi instances, we modify the Solomon instances in two ways: the demand distribution and the customer selection. The demand distributions are modified as follows. Originally, the customer demands in the Solomon instances are between 0 and 40. We denote the demand types in the Solomon instances between the intervals \((0,10], (10,20], (20,30], (30,40] as 1, 2, 3, 4, respectively. We then assign the demand distributions \(U(0,4), U(2,6), U(4,8)\) and \(U(6,10)\) to the four demand types respectively. The customer selection process is based on the customer ready times. The customers are ordered based on their ready times without using their time windows. We pick the first \(N\) customers to construct our instances. For the small size instances, we select the first 5 to 15 customers. The vehicle capacity \(Q\) is then set to \([8N/1.75]\). After the modification, we generate two instance sets according to the location of the depot: Solomon A, where the depot is located at the center, and Solomon B, where the depot is located at the corner. For Solomon A instances, the depot is located at \((40,50)\), which is approximately at the center of the customers. For Solomon B instances, we swap the depot located at \((40,50)\) and the customer located at \((5,5)\). We generate 10 replications for each of the Solomon instances.
5.2. Evaluation methodology

We study three algorithms for the VRPSD: the Rollout algorithm, the VFA, and the VFA+. The solution quality is evaluated by policy simulation and the solution time is measured by the central processing unit (CPU) time. We evaluate the policy of each algorithm using simulation with a sample size of 2000 and report the mean of the evaluated objective values and solution times.

In the simulation, the same set of random seeds are used such that the different algorithms use the same demand samples. We report the improvements of VFA and VFA+ relative to the solution from the Rollout algorithm. For small size instances, we find the optimal objective values by solving a standard backward MDP using Equation (9). We also report the optimal objective values and their improvements compared to the Rollout algorithm. Note that we choose the Rollout algorithm as the benchmark because the optimal policy can only be obtained for small size instances.

6. Numerical results

This section provides the numerical results and analysis. All evaluations are run on a computer with an Intel Xeon CPU X7560 (2.27GHz) and 63.9GB RAM. The programming language is Java.

Algorithm comparison: Solution quality

We first compare the solutions quality of the algorithms for the VRPSD. Specifically, we compare the evaluated objective values of the Rollout algorithm, the VFA, the VFA+, as well as the optimal values (for small size instances). Further, we also provide the improvements of the latter three algorithms relative to the Rollout algorithm.

Table 1 summarizes the results for the Seokmandi instances for both small and large size instances. The entries are the averages of all variants of the fill rate and demand distribution for each instance size. When we consider the instances with $N \leq 15$, the optimal algorithm performs on average 3.78% better than the Rollout algorithm. On the same instances with $N \leq 15$, we see that both the VFA and the VFA+ on average perform better than the Rollout algorithm. For instance, the VFA+ on average improves the solution of the Rollout algorithm by 1.97%, covering the performance gap between the Rollout and the optimal solution by more than 50%.
Table 1 also demonstrates that the difference between the solution quality of the VFA+ and VFA is on average not very large for small size instances with \( N \leq 15 \). However, as the problem size increases from \( N > 8 \), the performance of the VFA algorithm gets worse whereas the VFA+ algorithm still outperforms the Rollout algorithm.

When the number of customers increases from 16 to 19, obtaining the optimal solution becomes computationally intractable. For these instances, the performance of the VFA gets worse than the Rollout algorithm. The VFA+, however, still outperforms the Rollout algorithm.

For large size instances \( (N \geq 20) \), both the optimal algorithm and the VFA become computationally intractable. The VFA+ algorithm outperforms the Rollout algorithm and the percentage improvement of the VFA+ algorithm increases from 1.87\% (small size instances) to 2.97\% (large size instances).

Table 1  
Overview of the performance of the Rollout, VFA, VFA+ and the Optimal Algorithm on Secomandi instances

<table>
<thead>
<tr>
<th>Secomandi instances</th>
<th>Rollout Value</th>
<th>VFA Value</th>
<th>VFA+ Value</th>
<th>Opt Value</th>
<th>VFA+ Imprv.*%</th>
<th>Opt Imprv.*%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.51</td>
<td>5.37</td>
<td>5.55</td>
<td>5.34</td>
<td>2.41%</td>
<td>-0.75%</td>
</tr>
<tr>
<td>6</td>
<td>5.40</td>
<td>5.28</td>
<td>5.29</td>
<td>5.21</td>
<td>2.24%</td>
<td>2.08%</td>
</tr>
<tr>
<td>7</td>
<td>5.27</td>
<td>5.13</td>
<td>5.13</td>
<td>5.07</td>
<td>2.76%</td>
<td>2.67%</td>
</tr>
<tr>
<td>8</td>
<td>5.54</td>
<td>5.44</td>
<td>5.40</td>
<td>5.31</td>
<td>1.82%</td>
<td>2.57%</td>
</tr>
<tr>
<td>9</td>
<td>5.36</td>
<td>5.29</td>
<td>5.25</td>
<td>5.16</td>
<td>1.36%</td>
<td>2.10%</td>
</tr>
<tr>
<td>10</td>
<td>5.44</td>
<td>5.35</td>
<td>5.33</td>
<td>5.26</td>
<td>1.59%</td>
<td>2.09%</td>
</tr>
<tr>
<td>11</td>
<td>6.08</td>
<td>6.01</td>
<td>5.97</td>
<td>5.84</td>
<td>1.10%</td>
<td>1.88%</td>
</tr>
<tr>
<td>12</td>
<td>6.27</td>
<td>6.20</td>
<td>6.11</td>
<td>5.95</td>
<td>1.26%</td>
<td>2.56%</td>
</tr>
<tr>
<td>13</td>
<td>6.20</td>
<td>6.10</td>
<td>6.11</td>
<td>6.00</td>
<td>1.56%</td>
<td>1.42%</td>
</tr>
<tr>
<td>14</td>
<td>6.21</td>
<td>6.14</td>
<td>6.08</td>
<td>6.01</td>
<td>1.22%</td>
<td>2.08%</td>
</tr>
<tr>
<td>15</td>
<td>6.22</td>
<td>6.10</td>
<td>6.04</td>
<td>5.95</td>
<td>1.92%</td>
<td>2.91%</td>
</tr>
<tr>
<td>Average</td>
<td>5.77</td>
<td>5.67</td>
<td>5.66</td>
<td>5.56</td>
<td>1.73%</td>
<td>1.97%</td>
</tr>
<tr>
<td>16</td>
<td>6.64</td>
<td>6.61</td>
<td>6.51</td>
<td>6.51</td>
<td>0.40%</td>
<td>1.89%</td>
</tr>
<tr>
<td>17</td>
<td>6.36</td>
<td>6.41</td>
<td>6.28</td>
<td>6.28</td>
<td>-0.71%</td>
<td>1.27%</td>
</tr>
<tr>
<td>18</td>
<td>6.23</td>
<td>6.27</td>
<td>6.12</td>
<td>6.12</td>
<td>-0.54%</td>
<td>1.89%</td>
</tr>
<tr>
<td>19</td>
<td>6.66</td>
<td>6.72</td>
<td>6.57</td>
<td>6.57</td>
<td>-0.98%</td>
<td>1.36%</td>
</tr>
<tr>
<td>Average (Small size instances)</td>
<td>5.96</td>
<td>5.89</td>
<td>5.85</td>
<td>5.85</td>
<td>1.10%</td>
<td>1.87%</td>
</tr>
<tr>
<td>20</td>
<td>6.86</td>
<td>6.73</td>
<td>6.61</td>
<td>6.61</td>
<td>1.81%</td>
<td>3.64%</td>
</tr>
<tr>
<td>30</td>
<td>7.51</td>
<td>7.52</td>
<td>7.21</td>
<td>7.21</td>
<td>-0.10%</td>
<td>4.04%</td>
</tr>
<tr>
<td>40</td>
<td>8.14</td>
<td>7.98</td>
<td>7.98</td>
<td>7.98</td>
<td>1.98%</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>8.49</td>
<td>8.29</td>
<td>8.29</td>
<td>8.29</td>
<td>2.29%</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>9.07</td>
<td>8.79</td>
<td>8.79</td>
<td>8.79</td>
<td>3.09%</td>
<td></td>
</tr>
<tr>
<td>Average (Large size instances)</td>
<td>8.01</td>
<td>7.78</td>
<td>7.78</td>
<td>7.78</td>
<td>2.97%</td>
<td></td>
</tr>
<tr>
<td>Grand Average</td>
<td>6.47</td>
<td>6.33</td>
<td>6.33</td>
<td>6.33</td>
<td>2.21%</td>
<td></td>
</tr>
</tbody>
</table>

* Percentage cost improvement relative to the Rollout algorithm.

Table 2 presents the results of the Solomon A and the Solomon B instances with the number of customers: \( N \leq 15 \). The results represented are the average values for each instance size. The left part of the table shows the results when the depot is approximately at the center of the customers
(Solomon A) and the right part of the table presents the results when the depot is approximately
at the corner (Solomon B). We use these two sets of instances to analyze the effect of the depot
location on the performance of the algorithms.

The results for Solomon A instances show that when the depot is at the center, the relative
performance of the VFA, the VFA\(^+\) algorithm and the optimal algorithm is on average not much
different from the performance of the Rollout algorithm. This suggests that the Rollout algorithm
performs good and also the VFA algorithms and the optimal algorithm behaves similarly with the
central depot.

When we look at Solomon B instances, the results are similar to the results of the Secomandi
instances with \(N \leq 15\). Both the VFA and the VFA\(^+\) algorithm improve the solution of the Rollout
algorithm by covering on average more than 50\% of the performance gap between the Rollout
and the optimal solution. Intuitively, moving the depot from the center to the corner increases
the average customer-depot distance. Therefore, when the depot is at the corner, the penalty of a
failure is higher as we have to travel on average longer distance back to the depot. This indicates
that in value function algorithms, the decisions for which customer to go next and for the early
replenishments are made efficiently such that the overall cost decreases.

| Table 2 | Overview of the performance of the Rollout, VFA, VFA\(^+\) and the Optimal Algorithm on Solomon A
and Solomon B instances |
|---------|-----------------------|

<table>
<thead>
<tr>
<th>N</th>
<th>Rollout Value</th>
<th>VFA Value</th>
<th>VFA(^+) Value</th>
<th>Opt Value</th>
<th>VFA (^+) Improv. (^%)</th>
<th>Opt Value</th>
<th>VFA (^+) Improv. (^%)</th>
<th>Rollout Value</th>
<th>VFA Value</th>
<th>VFA(^+) Value</th>
<th>Opt Value</th>
<th>VFA (^+) Improv. (^%)</th>
<th>Opt Value</th>
<th>VFA (^+) Improv. (^%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.423</td>
<td>1.406</td>
<td>1.391</td>
<td>1.387</td>
<td>1.23%</td>
<td>2.31%</td>
<td>2.57%</td>
<td>2.811</td>
<td>2.799</td>
<td>2.791</td>
<td>2.779</td>
<td>0.43%</td>
<td>0.74%</td>
<td>1.14%</td>
</tr>
<tr>
<td>6</td>
<td>1.645</td>
<td>1.652</td>
<td>1.623</td>
<td>1.612</td>
<td>-0.44%</td>
<td>1.35%</td>
<td>2.01%</td>
<td>2.817</td>
<td>2.776</td>
<td>2.775</td>
<td>2.766</td>
<td>1.46%</td>
<td>1.49%</td>
<td>1.81%</td>
</tr>
<tr>
<td>7</td>
<td>1.854</td>
<td>1.849</td>
<td>1.862</td>
<td>1.836</td>
<td>0.22%</td>
<td>-0.44%</td>
<td>0.96%</td>
<td>2.789</td>
<td>2.718</td>
<td>2.712</td>
<td>2.699</td>
<td>2.56%</td>
<td>2.76%</td>
<td>3.22%</td>
</tr>
<tr>
<td>8</td>
<td>2.090</td>
<td>2.079</td>
<td>2.106</td>
<td>2.065</td>
<td>0.54%</td>
<td>-0.75%</td>
<td>2.11%</td>
<td>3.047</td>
<td>2.911</td>
<td>2.935</td>
<td>2.890</td>
<td>4.48%</td>
<td>3.70%</td>
<td>5.16%</td>
</tr>
<tr>
<td>9</td>
<td>2.290</td>
<td>2.292</td>
<td>2.285</td>
<td>2.257</td>
<td>-0.08%</td>
<td>0.26%</td>
<td>1.43%</td>
<td>3.146</td>
<td>3.031</td>
<td>3.060</td>
<td>3.007</td>
<td>3.63%</td>
<td>2.73%</td>
<td>4.42%</td>
</tr>
<tr>
<td>10</td>
<td>2.358</td>
<td>2.350</td>
<td>2.336</td>
<td>2.323</td>
<td>0.32%</td>
<td>0.92%</td>
<td>1.47%</td>
<td>3.086</td>
<td>3.054</td>
<td>3.038</td>
<td>2.996</td>
<td>1.04%</td>
<td>1.58%</td>
<td>2.92%</td>
</tr>
<tr>
<td>11</td>
<td>2.502</td>
<td>2.513</td>
<td>2.487</td>
<td>2.468</td>
<td>-0.41%</td>
<td>0.60%</td>
<td>1.38%</td>
<td>3.261</td>
<td>3.230</td>
<td>3.217</td>
<td>3.169</td>
<td>0.95%</td>
<td>1.36%</td>
<td>2.83%</td>
</tr>
<tr>
<td>12</td>
<td>2.662</td>
<td>2.673</td>
<td>2.642</td>
<td>2.633</td>
<td>-0.39%</td>
<td>0.77%</td>
<td>1.09%</td>
<td>3.387</td>
<td>3.369</td>
<td>3.334</td>
<td>3.300</td>
<td>0.53%</td>
<td>1.57%</td>
<td>2.55%</td>
</tr>
<tr>
<td>13</td>
<td>2.714</td>
<td>2.717</td>
<td>2.693</td>
<td>2.673</td>
<td>-0.10%</td>
<td>0.80%</td>
<td>1.51%</td>
<td>3.358</td>
<td>3.335</td>
<td>3.331</td>
<td>3.283</td>
<td>0.67%</td>
<td>0.79%</td>
<td>2.22%</td>
</tr>
<tr>
<td>14</td>
<td>2.905</td>
<td>2.921</td>
<td>2.892</td>
<td>2.874</td>
<td>-0.56%</td>
<td>0.44%</td>
<td>1.04%</td>
<td>3.605</td>
<td>3.558</td>
<td>3.590</td>
<td>3.489</td>
<td>1.31%</td>
<td>0.43%</td>
<td>3.22%</td>
</tr>
<tr>
<td>15</td>
<td>3.014</td>
<td>3.017</td>
<td>2.981</td>
<td>2.954</td>
<td>-0.12%</td>
<td>1.09%</td>
<td>1.99%</td>
<td>3.673</td>
<td>3.607</td>
<td>3.627</td>
<td>3.519</td>
<td>1.78%</td>
<td>1.25%</td>
<td>4.18%</td>
</tr>
<tr>
<td>Average</td>
<td>2.314</td>
<td>2.315</td>
<td>2.306</td>
<td>2.290</td>
<td>-0.05%</td>
<td>0.63%</td>
<td>1.47%</td>
<td>3.180</td>
<td>3.126</td>
<td>3.128</td>
<td>3.082</td>
<td>1.69%</td>
<td>1.64%</td>
<td>3.09%</td>
</tr>
</tbody>
</table>

* Percentage cost improvement relative to the Rollout algorithm.

Algorithm comparison: Computational times

Figure 1 shows the normalized computational times (in logarithmic scale) for the different algo-
rithms in solving the Secomandi instances. The normalized time is calculated as the computational
time relative to solving the instances with $N = 5$ customers. Figure 1 shows that the MDP solution time increases at a much higher rate than other algorithms. The VFA$^+$ has the slowest rate of increase as the number of the customers increases. This shows that the VFA$^+$ algorithm reduces the computational time significantly when compared to the optimal algorithm, the Rollout algorithm and the VFA while providing good quality solutions by using bounded lookup tables with efficient maintenance.

In Figure 1, there is a sudden jump up in the computational time of the VFA and Rollout algorithms from the small size ($\leq 19$ customers) to the large size instances ($\geq 20$ customers). This is because, in the experimental design of the large size instances, we use demand categories with a wider range that increases the number of the state-decision pairs (see Section 5.1). However, due to the use of bounded lookup tables with efficient maintenance, the VFA$^+$ algorithm successfully mitigates this increase in the state-decision space (see “state difference” explanation in Section 4.2).
7. Conclusions

This paper deals with the single vehicle routing problem with stochastic demands (VRPSD). The VRPSD is a difficult stochastic combinatorial optimization problem that becomes intractable for large size instances. In this paper, we formulate a multi-stage stochastic dynamic programming model and implement Approximate Dynamic Programming (ADP) algorithms to overcome the curses of dimensionality for the VRPSD. The ADP algorithms are based on Value Function Approximations (VFA) with a lookup table representation. The standard VFA is improved for VRPSD with a Q-learning algorithm with bounded lookup tables and efficient maintenance (VFA$^+$), as well as exploration-and-exploitation strategies using preventive returns and restocking.

We validate and benchmark our proposed algorithms using test instances in the literature. The VFA$^+$ algorithm obtains good quality solutions with shorter computational time, especially for large size instances. For small size instances where the optimal solutions are available, VFA$^+$ improves the Rollout algorithm by covering on average more than 50% of the performance gap between the Rollout and the optimal solutions. For large size instances, VFA$^+$ outperforms both VFA and the Rollout algorithm. This demonstrates the effectiveness in the algorithm design of VFA$^+$. 

In Approximate Dynamic Programming, the use of post-decision states helps to capture the state of the system immediately after the decision making but before the new (exogenous) information arrives. It also helps to avoid the expectation within the min or max operator. However, in a stochastic combinatorial optimization problem such as VRPSD, we need both the state and the decision information to evaluate the impact of the decision, where the practice of post-decision states appears to be inappropriate. Therefore, we improve the ADP algorithm with a Q-learning algorithm where we store the values of state-decision pairs, i.e., Q-factors. It however suffers more from the curses of dimensionality. We design bounded lookup tables with efficient maintenance to overcome this. Further, we design exploration-and-exploitation strategies using preventive returns for VRPSD. The combination of the above algorithmic strategies appear to play an important role in making better routing and restocking decisions in VRPSD. This paper provides an exploratory algorithmic research on the application of Q-learning algorithms with bounded lookup table and efficient maintenance as well as exploration-and-exploitation strategies in dealing with difficult stochastic and combinatorial problems. More in-depth research is called for along this line.
Appendix. The illustration for state-decision pairs and $i_t$ sets

In Figure 2, an illustration for the state-decision pairs and $i_t$ sets is given for a small single vehicle routing problem. In this example, there are only 3 customers, the capacity of the vehicle is 4 units and the demand of each customer follows a discrete uniform distribution $U(1,2)$. Here, we only show the branch from the customer 1 at state 1 until the termination state which ends at the depot. The $i_t$ sets are grouped according to the current customer.
References


<table>
<thead>
<tr>
<th>nr.</th>
<th>Year</th>
<th>Title</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>424</td>
<td>2013</td>
<td><strong>Influence of Spillback Effect on Dynamic Shortest Path Problems with Travel-Time-Dependent Network Disruptions</strong></td>
<td>Derya Sever, Nico Dellaert, Tom Van Woensel, Ton de Kok</td>
</tr>
<tr>
<td>423</td>
<td>2013</td>
<td><strong>Dynamic Shortest Path Problem with Travel-Time-Dependent Stochastic Disruptions: Hybrid Approximate Dynamic Programming Algorithms with a Clustering Approach</strong></td>
<td>Derya Sever, Lei Zhao, Nico Dellaert, Tom Van Woensel, Ton de Kok</td>
</tr>
<tr>
<td>422</td>
<td>2013</td>
<td><strong>System-oriented inventory models for spare parts</strong></td>
<td>R.J.I. Basten, G.J. van Houtum</td>
</tr>
<tr>
<td>421</td>
<td>2013</td>
<td><strong>Lost Sales Inventory Models with Batch Ordering And Handling Costs</strong></td>
<td>T. Van Woensel, N. Erkip, A. Curseu, J.C. Fransoo</td>
</tr>
<tr>
<td>420</td>
<td>2013</td>
<td><strong>Response speed and the bullwhip</strong></td>
<td>Maximiliano Udenio, Jan C. Fransoo, Eleni Vatamidou, Nico Dellaert</td>
</tr>
<tr>
<td>419</td>
<td>2013</td>
<td><strong>Anticipatory Routing of Police Helicopters</strong></td>
<td>Rick van Urk, Martijn R.K. Mes, Erwin W. Hans</td>
</tr>
<tr>
<td>418</td>
<td>2013</td>
<td><strong>Supply Chain Finance. A conceptual framework to advance research</strong></td>
<td>Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo</td>
</tr>
<tr>
<td>417</td>
<td>2013</td>
<td><strong>Improving the Performance of Sorter Systems By Scheduling Inbound Containers</strong></td>
<td>S.W.A. Haneyah, J.M.J. Schutten, K. Fikse</td>
</tr>
<tr>
<td>416</td>
<td>2013</td>
<td><strong>Regional logistics land allocation policies: Stimulating spatial concentration of logistics firms</strong></td>
<td>Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo</td>
</tr>
<tr>
<td>415</td>
<td>2013</td>
<td><strong>The development of measures of process harmonization</strong></td>
<td>Heidi L. Romero, Remco M. Dijkman, Paul W.P.J. Grefen, Arjan van Weele</td>
</tr>
<tr>
<td>414</td>
<td>2013</td>
<td><strong>BASE/X. Business Agility through Cross-Organizational Service Engineering</strong></td>
<td>Paul Grefen, Egon Lüftenegger, Eric van der Linden, Caren Weisleder</td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>The Time-Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times</td>
<td>Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Clearing the Sky - Understanding SLA Elements in Cloud Computing</td>
<td>Marco Comuzzi, Guus Jacobs, Paul Grefen</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Approximations for the waiting time distribution in an $M/G/c$ priority queue</td>
<td>A. Al Hanbali, E.M. Alvarez, M.C. van der van der Heijden</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>To co-locate or not? Location decisions and logistics concentration areas</td>
<td>Frank P. van den Heuvel, Karel H. van Donselaar, Rob A.C.M. Broekmeulen, Jan C. Fransoo, Peter W. de Langen</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>The Time-Dependent Pollution-Routing Problem</td>
<td>Anna Franceschetti, Dorothée Honhon, Tom van Woensel, Tolga Bektas, Gilbert Laporte.</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Scheduling the scheduling task: A time Management perspective on scheduling</td>
<td>J.A. Larco, V. Wiers, J. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Clustering Clinical Departments for Wards to Achieve a Prespecified Blocking Probability</td>
<td>J. Theresia van Essen, Mark van Houdenhoven, Johann L. Hurink</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>MyPHRMachines: Personal Health Desktops in the Cloud</td>
<td>Pieter Van Gorp, Marco Comuzzi</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Maximising the Value of Supply Chain Finance</td>
<td>Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Reaching 50 million nanostores: retail distribution in emerging megacities</td>
<td>Edgar E. Blanco, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>A Vehicle Routing Problem with Flexible Time Windows</td>
<td>Duygu Tas, Ola Jabali, Tom van Woensel</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>The Service Dominant Business Model: A Service Focused Conceptualization</td>
<td>Egon Lüftenegger, Marco Comuzzi, Paul Grefen, Caren Weisleder</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Relationship between freight accessibility and Logistics employment in US counties</td>
<td>Frank P. van den Heuvel, Liliana Rivera, Karel H. van Donselaar, Ad de Jong, Yossi Sheffi, Peter W. de Langen, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>400</td>
<td>2012</td>
<td>A Condition-Based Maintenance Policy for Multi-Component Systems with a High Maintenance Setup Cost</td>
<td>Qiushi Zhu, Hao Peng, Geert-Jan van Houtum</td>
</tr>
<tr>
<td>398</td>
<td>2012</td>
<td>Scheduled Service Network Design with Synchronization and Transshipment Constraints For Intermodal Container Transportation Networks</td>
<td>K. Sharypova, T.G. Crainic, T. van Woensel, J.C. Fransoo</td>
</tr>
<tr>
<td>397</td>
<td>2012</td>
<td>Destocking, the bullwhip effect, and the credit Crisis: empirical modeling of supply chain Dynamics</td>
<td>Maximiliano Udenio, Jan C. Fransoo, Robert Peels</td>
</tr>
<tr>
<td>394</td>
<td>2012</td>
<td>A Generalized Simulation Model of an Integrated Emergency Post</td>
<td>Martijn Mes, Manon Bruens</td>
</tr>
<tr>
<td>390</td>
<td>2012</td>
<td>How does development lead time affect performance over the ramp-up lifecycle?</td>
<td>Andres Pufall, Jan C. Fransoo, Ad de Jong</td>
</tr>
<tr>
<td>389</td>
<td>2012</td>
<td>Evidence from the consumer electronics industry</td>
<td>Andreas Pufall, Jan C. Fransoo, Ad de Jong, Ton de Kok</td>
</tr>
<tr>
<td>Page</td>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>386</td>
<td>2012</td>
<td>Proximity matters: Synergies through co-location of logistics establishments</td>
<td>Frank P. v.d.Heuvel, Peter W.de Langen, Karel H. v. Donselaar, Jan C. Fransoo</td>
</tr>
<tr>
<td>385</td>
<td>2012</td>
<td>Spatial concentration and location dynamics in logistics: the case of a Dutch province</td>
<td>Zhiqiang Yan, Remco Dijkman, Paul Grefen</td>
</tr>
<tr>
<td>384</td>
<td>2012</td>
<td>FNet: An Index for Advanced Business Process Querying</td>
<td>W.R. Dalinghaus, P.M.E. Van Gorp</td>
</tr>
<tr>
<td>383</td>
<td>2012</td>
<td>Defining Various Pathway Terms</td>
<td>Egon Lüftenegger, Paul Grefen, Caren Weisleder</td>
</tr>
<tr>
<td>382</td>
<td>2012</td>
<td>The Service Dominant Strategy Canvas: Defining and Visualizing a Service Dominant Strategy through the Traditional Strategic Lens</td>
<td>Stefano Fazi, Tom van Woensel, Jan C. Fransoo</td>
</tr>
<tr>
<td>381</td>
<td>2012</td>
<td>A Stochastic Variable Size Bin Packing Problem With Time Constraints</td>
<td>K. Sharypova, T. van Woensel, J.C. Fransoo</td>
</tr>
<tr>
<td>380</td>
<td>2012</td>
<td>Coordination and Analysis of Barge Container Hinterland Networks</td>
<td>Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo</td>
</tr>
<tr>
<td>379</td>
<td>2012</td>
<td>Proximity matters: Synergies through co-location of logistics establishments</td>
<td>Heidi Romero, Remco Dijkman, Paul Grefen, Arjan van Weele</td>
</tr>
<tr>
<td>375</td>
<td>2012</td>
<td>Improving the performance of sorter systems by scheduling inbound containers</td>
<td>Albert Douma, Martijn Mes</td>
</tr>
<tr>
<td>Page</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>374</td>
<td>Strategies for dynamic appointment making by container terminals</td>
<td>Pieter van Gorp, Marco Comuzzi</td>
<td></td>
</tr>
<tr>
<td>373</td>
<td>MyPHRMachines: Lifelong Personal Health Records in the Cloud</td>
<td>E.M. Alvarez, M.C. van der Heijden, W.H.M. Zijm</td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>Service differentiation in spare parts supply through dedicated stocks</td>
<td>Frank Karsten, Rob Basten</td>
<td></td>
</tr>
<tr>
<td>371</td>
<td>Spare parts inventory pooling: how to share the benefits</td>
<td>X.Lin, R.J.I. Basten, A.A. Kranenburg, G.J. van Houtum</td>
<td></td>
</tr>
<tr>
<td>370</td>
<td>Condition based spare parts supply</td>
<td>Martijn Mes</td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>Using Simulation to Assess the Opportunities of Dynamic Waste Collection</td>
<td>J. Arts, S.D. Flapper, K. Vernooij</td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>Aggregate overhaul and supply chain planning for rotable</td>
<td>J.T. van Essen, J.L. Hurink, W. Hartholt, B.J. van den Akker</td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>Operating Room Rescheduling</td>
<td>Kristel M.R. Hoen, Tarkan Tan, Jan C. Fransoo, Geert-Jan van Houtum</td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>Switching Transport Modes to Meet Voluntary Carbon Emission Targets</td>
<td>Elisa Alvarez, Matthieu van der Heijden</td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>On two-echelon inventory systems with Poisson demand and lost sales</td>
<td>J.T. van Essen, E.W. Hans, J.L. Hurink, A. Oversberg</td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>Minimizing the Waiting Time for Emergency Surgery</td>
<td>Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok</td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>Vehicle Routing Problem with Stochastic Travel Times Including Soft Time Windows and Service Costs</td>
<td>Erhun Özkan, Geert-Jan van Houtum, Yasemin Serin</td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>A New Approximate Evaluation Method for Two-Echelon Inventory Systems with Emergency Shipments</td>
<td>Said Dabia, El-Ghazali Talbi, Tom Van Woensel, Ton de Kok</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>Approximating Multi-Objective Time-Dependent Optimization Problems</td>
<td>Said Dabia, Stefan Röpke, Tom Van Woensel, Ton de Kok</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Analysis of an Assemble-to-Order System with Different Review Periods</td>
<td>Ahmad Al Hanbali, Matthieu van der Heijden</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Interval Availability Analysis of a Two-Echelon, Multi-Item System</td>
<td>Felipe Caro, Charles J. Corbett, Tarkan Tan, Rob Zuidwijk</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Carbon-Optimal and Carbon-Neutral Supply Chains</td>
<td>Sameh Haneyah, Henk Zijm, Marco Schutten, Peter Schuur</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Generic Planning and Control of Automated Material Handling Systems: Practical Requirements Versus Existing Theory</td>
<td>M. van der Heijden, B. Iskandar</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Last time buy decisions for products sold under warranty</td>
<td>Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Spatial concentration and location dynamics in logistics: the case of a Dutch provence</td>
<td>Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Identification of Employment Concentration Areas</td>
<td>Pieter van Gorp, Remco Dijkman</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>BOMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules: extended version</td>
<td>Frank Karsten, Marco Slikker, Geert-Jan van Houtum</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Resource pooling and cost allocation among independent service providers</td>
<td>E. Lüftenegger, S. Angelov, P. Grefen</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>A Framework for Business Innovation Directions</td>
<td>Remco Dijkman, Irene Vanderfeesten, Hajo A. Reijers</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Effect of carbon emission regulations on transport mode selection under stochastic demand</td>
<td>Murat Firat, Cor Hurkens</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>An approximate approach for the joint problem of level of repair analysis and spare parts stocking</td>
<td>R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Joint optimization of level of repair analysis and spare parts stocks</td>
<td>R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td><em>Inventory control with manufacturing lead time flexibility</em></td>
<td>Ton G. de Kok</td>
<td></td>
</tr>
<tr>
<td>344</td>
<td><em>Analysis of resource pooling games via a new extension of the Erlang loss function</em></td>
<td>Frank Karsten, Marco Slikker, Geert-Jan van Houtum</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td><em>Vehicle refueling with limited resources</em></td>
<td>Murat Firat, C.A.J. Hurkens, Gerhard J. Woeginger</td>
<td></td>
</tr>
<tr>
<td>342</td>
<td><em>Optimal Inventory Policies with Non-stationary Supply Disruptions and Advance Supply Information</em></td>
<td>Bilge Atasoy, Refik Güllü, TarkanTan</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td><em>Redundancy Optimization for Critical Components in High-Availability Capital Goods</em></td>
<td>Kurtulus Baris Öner, Alan Scheller-Wolf, Geert-Jan van Houtum</td>
<td></td>
</tr>
<tr>
<td>339</td>
<td><em>Analysis of a two-echelon inventory system with two supply modes</em></td>
<td>Joachim Arts, Gudrun Kiesmüller</td>
<td></td>
</tr>
<tr>
<td>338</td>
<td><em>Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh</em></td>
<td>Murat Firat, Gerhard J. Woeginger</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td><em>Attaining stability in multi-skil workforce scheduling</em></td>
<td>Murat Firat, Cor Hurkens</td>
<td></td>
</tr>
<tr>
<td>333</td>
<td><em>An exact approach for relating recovering surgical patient workload to the master surgical schedule</em></td>
<td>P.T. Vanberkel, R.J. Boucherie, E.W. Hans, J.L. Hurink, W.A.M. van Lent, W.H. van Harten</td>
<td></td>
</tr>
<tr>
<td>332</td>
<td><em>Efficiency evaluation for pooling resources in health care</em></td>
<td>Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Nelly Litvak</td>
<td></td>
</tr>
<tr>
<td>331</td>
<td><em>The Effect of Workload Constraints in Mathematical Programming Models for Production Planning</em></td>
<td>M.M. Jansen, A.G. de Kok, I.J.B.F. Adan</td>
<td></td>
</tr>
<tr>
<td>330</td>
<td><em>Using pipeline information in a multi-echelon spare parts inventory system</em></td>
<td>Christian Howard, Ingrid Reijnen, Johan Marklund, Tarkan Tan</td>
<td></td>
</tr>
<tr>
<td>329</td>
<td><em>Reducing costs of repairable spare parts supply systems via dynamic scheduling</em></td>
<td>H.G.H. Tiemessen, G.J. van Houtum</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>A combinatorial approach to multi-skill workforce scheduling</td>
<td>Murat Firat, Cor Hurkens</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Stability in multi-skill workforce scheduling</td>
<td>Murat Firat, Cor Hurkens, Alexandre Laugier</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Near-optimal heuristics to set base stock levels in a two-echelon distribution network</td>
<td>R.J.I. Basten, G.J. van Houtum</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Inventory reduction in spare part networks by selective throughput time reduction</td>
<td>M.C. van der Heijden, E.M. Alvarez, J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>The selective use of emergency shipments for service-contract differentiation</td>
<td>E.M. Alvarez, M.C. van der Heijden, W.H. Zijm</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Preventing or escaping the suppression mechanism: intervention conditions</td>
<td>Nico Dellaert, Jully Jeunet.</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Hospital admission planning to optimize major resources utilization under uncertainty</td>
<td>R. Seguel, R. Eshuis, P. Grefen.</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Minimal Protocol Adaptors for Interacting Services</td>
<td>Tom Van Woensel, Marshall L. Fisher, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Teaching Retail Operations in Business and Engineering Schools</td>
<td>Lydie P.M. Smets, Geert-Jan van Houtum, Fred Langerak</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Design for Availability: Creating Value for Manufacturers and Customers</td>
<td>Pieter van Gorp, Rik Eshuis.</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Murat Firat, Cor Hurkens</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Murat Firat, Cor Hurkens, Alexandre Laugier</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>R.J.I. Basten, G.J. van Houtum</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>M.C. van der Heijden, E.M. Alvarez, J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>E.M. Alvarez, M.C. van der Heijden, W.H. Zijm</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Nico Dellaert, Jully Jeunet.</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Tom Van Woensel, Marshall L. Fisher, Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Lydie P.M. Smets, Geert-Jan van Houtum, Fred Langerak</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>Pieter van Gorp, Rik Eshuis.</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>315</td>
<td>2010</td>
<td>Getting trapped in the suppression of exploration: A simulation model</td>
<td>Bob Walrave, Kim E. van Oorschot, A. Georges L. Romme</td>
</tr>
<tr>
<td>313</td>
<td>2010</td>
<td>Tales of a So(u)rcerer: Optimal Sourcing Decisions Under Alternative Capacitated Suppliers and General Cost Structures</td>
<td>Osman Alp, Tarkan Tan</td>
</tr>
<tr>
<td>312</td>
<td>2010</td>
<td>In-store replenishment procedures for perishable inventory in a retail environment with handling costs and storage constraints</td>
<td>R.A.C.M. Broekmeulen, C.H.M. Bakx</td>
</tr>
<tr>
<td>311</td>
<td>2010</td>
<td>The state of the art of innovation-driven business models in the financial services industry</td>
<td>E. Lüftenegger, S. Angelov, E. van der Linden, P. Grefen</td>
</tr>
<tr>
<td>309</td>
<td>2010</td>
<td>Effect of carbon emission regulations on transport mode selection in supply chains</td>
<td>K.M.R. Hoen, T. Tan, J.C. Fransoo, G.J. van Houtum</td>
</tr>
<tr>
<td>308</td>
<td>2010</td>
<td>Interaction between intelligent agent strategies for real-time transportation planning</td>
<td>Martijn Mes, Matthieu van der Heijden, Peter Schuur</td>
</tr>
<tr>
<td>307</td>
<td>2010</td>
<td>Internal Slackening Scoring Methods</td>
<td>Marco Slikker, Peter Borm, René van den Brink</td>
</tr>
<tr>
<td>305</td>
<td>2010</td>
<td>Practical extensions to the level of repair analysis</td>
<td>R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten</td>
</tr>
<tr>
<td>304</td>
<td>2010</td>
<td>Ocean Container Transport: An Underestimated and Critical Link in Global Supply Chain Performance</td>
<td>Jan C. Fransoo, Chung-Yee Lee</td>
</tr>
<tr>
<td>303</td>
<td>2010</td>
<td>Capacity reservation and utilization for a manufacturer with uncertain capacity and demand</td>
<td>Y. Boulaksil; J.C. Fransoo; T. Tan</td>
</tr>
<tr>
<td>302</td>
<td>2010</td>
<td>Spare parts inventory pooling games</td>
<td>F.J.P. Karsten; M. Slikker; G.J. van Houtum</td>
</tr>
<tr>
<td>301</td>
<td>2009</td>
<td>Capacity flexibility allocation in an outsourced supply chain with reservation</td>
<td>Y. Boulaksil, M. Grunow, J.C. Fransoo</td>
</tr>
<tr>
<td>298</td>
<td>2010</td>
<td>An optimal approach for the joint problem of level of repair analysis and spare parts stocking</td>
<td>R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten</td>
</tr>
<tr>
<td>297</td>
<td>2010</td>
<td>Responding to the Lehman Wave: Sales Forecasting and Supply Management during the Credit Crisis</td>
<td>Robert Peels, Maximiliano Udenio, Jan C. Fransoo, Marcel Wolfs, Tom Hendrikx</td>
</tr>
<tr>
<td>296</td>
<td>2009</td>
<td>An exact approach for relating recovering surgical patient workload to the master surgical schedule</td>
<td>Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Wineke A.M. van Lent, Wim H. van Harten</td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>An iterative method for the simultaneous optimization of repair decisions and spare parts stocks</td>
<td>R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Fujaba hits the Wall(-e)</td>
<td>Pieter van Gorp, Ruben Jubeh, Bernhard Grusie, Anne Keller</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Business Process Model Repositories - Framework and Survey</td>
<td>Zhiqiang Yan, Remco Dijkman, Paul Grefen</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Efficient Optimization of the Dual-Index Policy Using Markov Chains</td>
<td>Joachim Arts, Marcel van Vuuren, Gudrun Kiesmuller</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Hierarchical Knowledge-Gradient for Sequential Sampling</td>
<td>Martijn R.K. Mes; Warren B. Powell; Peter I. Frazier</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Analyzing combined vehicle routing and break scheduling from a distributed decision making perspective</td>
<td>C.M. Meyer; A.L. Kok; H. Kopfer; J.M.J. Schutten</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Anticipation of lead time performance in Supply Chain Operations Planning</td>
<td>Michiel Jansen; Ton G. de Kok; Jan C. Fransoo</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Inventory Models with Lateral Transshipments: A Review</td>
<td>Colin Paterson; Gudrun Kiesmuller; Ruud Teunter; Kevin Glazebro</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Efficiency evaluation for pooling resources in health care</td>
<td>P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>A Survey of Health Care Models that Encompass Multiple Departments</td>
<td>P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Supporting Process Control in Business Collaborations</td>
<td>S. Angelov; K. Vidyasankar; J. Vonk; P. Grefen</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Inventory Control with Partial Batch Ordering</td>
<td>O. Alp; W.T. Huh; T. Tan</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Translating Safe Petri Nets to Statecharts in a Structure-Preserving Way</td>
<td>R. Eshuis</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>The link between product data model and process model</td>
<td>J.J.C.L. Vogelaar; H.A. Reijers</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Inventory planning for spare parts networks with delivery time requirements</td>
<td>I.C. Reijnen; T. Tan; G.J. van Houtum</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Co-Evolution of Demand and Supply under Competition</td>
<td>B. Vermeulen; A.G. de Kok</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Toward Meso-level Product-Market Network Indices for Strategic Product Selection and (Re)Design Guidelines over the Product Life-Cycle</td>
<td>B. Vermeulen, A.G. de Kok</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>An Efficient Method to Construct Minimal Protocol Adaptors</td>
<td>R. Seguel, R. Eshuis, P. Grefen</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Coordinating Supply Chains: a Bilevel Programming Approach</td>
<td>Ton G. de Kok, Gabriella Muratore</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Inventory redistribution for fashion products under</td>
<td>G.P. Kiesmuller, S. Minner</td>
<td></td>
</tr>
<tr>
<td>Paper Number</td>
<td>Year</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>274 2009</td>
<td></td>
<td><strong>Comparing Markov chains: Combining aggregation and precedence relations applied to sets of states</strong></td>
<td>A. Busic, I.M.H. Vliegen, A. Scheller-Wolf</td>
</tr>
<tr>
<td>273 2009</td>
<td></td>
<td><strong>Separate tools or tool kits: an exploratory study of engineers' preferences</strong></td>
<td>I.M.H. Vliegen, P.A.M. Kleingeld, G.J. van Houtum</td>
</tr>
<tr>
<td>272 2009</td>
<td></td>
<td><strong>An Exact Solution Procedure for Multi-Item Two-Echelon Spare Parts Inventory Control Problem with Batch Ordering</strong></td>
<td>Engin Topan, Z. Pelin Bayindir, Tarkan Tan</td>
</tr>
<tr>
<td>269 2009</td>
<td></td>
<td><strong>Similarity of Business Process Models: Metics and Evaluation</strong></td>
<td>Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Kaarik, Jan Mendling</td>
</tr>
<tr>
<td>266 2009</td>
<td></td>
<td><strong>Restricted dynamic programming: a flexible framework for solving realistic VRPs</strong></td>
<td>J. Gromicho; J.J. van Hoorn; A.L. Kok; J.M.J. Schutten;</td>
</tr>
</tbody>
</table>

Working Papers published before 2009 see: [http://beta.ieis.tue.nl](http://beta.ieis.tue.nl)