Single vehicle routing with stochastic demands : approximate dynamic programming

Beta

Research School for Operations Management and Logistics

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

BETA publicatie	WP 425 (working paper)
ISBN	
ISSN	
NUR	804
Eindhoven	July 2013

Submitted to *Transportation Science*

manuscript (Please, provide the mansucript number!)

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 resupply 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:

$$S_t = (i_t, l_t, J_t), \quad t = \{0, 1, \ldots, N, N+1\}, \tag{1}$$

where

i_t: 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:

$$x_t = (i_{t+1}, r_t), \quad t = \{0, 1, \ldots, N\}, \tag{2}$$

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 $X_t^{\pi}(S_t)$ as the *decision function* that determines decision x_t at stage t under *policy* π, given state S_t. Each $\pi \in \Pi$ refers to a different policy and Π 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}.

$$W_{t+1} = \hat{D}_{i_{t+1}}, \quad t = \{0, 1, \ldots, N-1\}. \tag{3}$$

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 $W_{t+1} = \hat{D}_{i_{t+1}}$, the state transition function is defined as, for $t = \{0, 1, \ldots, N\}$,

$$\begin{aligned} S_{t+1} &= S^M(S_t, x_t, W_{t+1}) \\ &= \begin{cases} (i_{t+1}, Q - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 1, \\ (i_{t+1}, l_t - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 0 \text{ and } l_t \geq \hat{D}_{i_{t+1}}, \\ (i_{t+1}, l_t + Q - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 0 \text{ and } l_t < \hat{D}_{i_{t+1}}, \end{cases} \end{aligned} \tag{4}$$

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).

If the vehicle returns to the depot to replenish after serving customer t ($r_t = 1$), it arrives at the next customer i_{t+1} with full capacity Q. Therefore, the capacity after serving customer i_{t+1} becomes $Q - \hat{D}_{i_{t+1}}$. If the vehicle travels to the next customer $i_{i_{t+1}}$ without returning to the depot ($r_t = 0$), it arrives at customer i_{t+1} with capacity l_t. If l_t is sufficient to serve the realized demand $\hat{D}_{i_{t+1}}$, l_{t+1} becomes $l_t - \hat{D}_{i_{t+1}}$; otherwise, the vehicle encounters a failure and needs to replenish at the depot to satisfy demand $\hat{D}_{i_{t+1}}$, thus l_{t+1} becomes $l_t + Q - \hat{D}_{i_{t+1}}$.

Cost function

The vehicle's actual travel distance or cost depends on both the decision and realized demand at the next customer $W_{t+1} = \hat{D}_{i_{t+1}}$. 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

$$C_t(S_t, x_t) = \begin{cases} d_{i_t,0} + d_{0,i_{t+1}}, & \text{if } r_t = 1, \\ d_{i_t,i_{t+1}}, & \text{if } r_t = 0, \end{cases} \tag{6}$$

and

$$\Delta C_{t+1}(S_t, x_t, W_{t+1}) = \begin{cases} 0 & \text{if } r_t = 1, \\ 0, & \text{if } r_t = 0 \text{ and } l_t \geq \hat{D}_{i_{t+1}}, \\ d_{i_{t+1},0} + d_{0,i_{t+1}}, & \text{if } r_t = 0 \text{ and } l_t < \hat{D}_{i_{t+1}}. \end{cases} \tag{7}$$

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,

$$\begin{aligned} &\min_{\pi \in \Pi} \sum_{t=0}^{N} c_t(S_t, x_t) \\ =&\min_{\pi \in \Pi} \sum_{t=0}^{N} c_t(S_t, X_t^{\pi}(S_t)), \end{aligned} \tag{8}$$

where $x_t = X_t^{\pi}(S_t)$ is the decision made according to the decision function $X_t^{\pi}(S_t)$ under policy π, 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,

$$V_t(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \mathbb{E}[V_{t+1}(S_{t+1})]). \tag{9}$$

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 $V_{t+1}(S_{t+1}) = V_{t+1}(S^M(S_t, x_t, W_{t+1}))$ 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.

$$\bar{V}_t(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \mathbb{E}[\bar{V}_{t+1}(S_{t+1})]). \tag{10}$$

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 ρ (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,

$$\bar{V}_t^n(S_t) = (1 - \alpha_{n-1})\bar{V}_t^{n-1}(S_t^n) + \alpha_{n-1}\hat{v}_t^n, \tag{11}$$

where α_{n-1} is the stepsize.

Algorithm 4.1 A generic VFA approach with lookup table representation.

Step 0. Initialization.

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

Step 0b. Choose an initial state S_0^1.

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

Step 1. Choose a sample path ω^n.

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

Step 2a. If *exploitation*, solve

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

and let x_t^n be the solution.

If *exploration*, randomly choose a solution $x_t^n \in \mathcal{X}_t$.

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

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

Step 2c. State transition.

$$S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n)).$$

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

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

Step 4. Output the value function, $\{\bar{V}_t^{\mathbb{N}}(S_t^x)\}_{t=0}^{N-1}$.

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

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

$(\bar{\sigma}^2)^n$ denotes the estimate of the variance of the value function $\bar{V}_t^n(S_t^n)$ and $\bar{\beta}^n$ denotes the estimate of bias due to smoothing a nonstationary data series. The BAKF stepsize rule adaptively balances the estimate of the noise $(\bar{\sigma}^2)^n$ and the estimate of the bias $\bar{\beta}^n$ 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 $\bar{V}_t(S_t)$ is associated with the *pre-decision state* S_t. Solving for Equation $(*)$ 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* $S_t^x = S^{M,x}(S_t, x_t)$, 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 $\bar{V}_t^{n-1}(S_t^x)$, we can solve for $\hat{v}_t^n(S_t)$ in Step 2a with

$$\hat{v}_t^n(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x)), \tag{13}$$

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

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 $\bar{Q}_t^n(S_t, x_t)$ 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 $\bar{Q}_t^0(S_t, x_t)$ for all states S_t and decisions $x_t \in \mathcal{X}_t$.

Step 0b. Choose an initial state (S_0^1).

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

Step 1. Choose a sample path ω^n.

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

Step 2a. If *exploitation*, find

$$x_t^n = \underset{x_t \in \mathcal{X}_t}{\operatorname{argmin}}(\bar{Q}_t^{n-1}(S_t, x_t) + SD(S_t, S_t^n)),$$

If *exploration*, choose a solution $x_t^n \in \mathcal{X}_t$

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

Step 2b. Compute the next state $S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n))$.

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

Step 3a. Calculate:

$$\hat{q}_t^n = C_t(S_t^n, x_t^n) + \hat{q}_{t+1}^n$$

Step 3b. Update $\bar{Q}_t^n(S_t, x_t^n)$ using

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

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

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

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

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):

$$SD(S_t, S'_t) = \sum |j_t - j'_t| + c_1 \times |l_t - l'_t| \tag{14}$$

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 $|i_t|_{max}$).

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 $|i_t|_{max}$, 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

Step 1. Check the size of the i_t set.

Step 2. Keep the state-decision pairs in the i_t set, if $|i_t| < |i_t|_{max}$.

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,
$Q(S_t, x_t) \geq (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot)\}} Q(S_t, x_t))/|i_t|$;
and if also their number of visits is less than the average number of visits of the i_t set, that is,
$n'(S_t, x_t) \leq (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot)\}} n'(S_t, x_t))/|i_t|$.

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 $x_t = (i_{t+1}, \cdot)$, 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
$\bar{Q}_{i_{t+1}} = (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot),x_t=(i_{t+1},\cdot)\}} Q(S_t,x_t))/n(i_t,i_{t+1})$,
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:
$Q_{min}(S_t) = \min_{i_{t+1}}[\mathbb{E}[(C_t(S_t,x_t)] + \min_{x_{t+1}\in\mathcal{X}_{t+1}}(SD(S_{t+1},S'_{t+1}) + Q(S_{t+1},x_{t+1}))]$.

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.

$$\alpha_{n-1} = \frac{a}{a + n'(S_t, x_t) - 1}, \tag{15}$$

where a is a constant. Note that the stepsize α_{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 $20, 30, \ldots, 60$ 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 Secomandi 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

Secomandi N	Rollout Value	VFA Value	VFA$^+$ Value	Opt Value	VFA Imprv.*%	VFA$^+$ Imprv.*%	Opt Imprv.*%
5	5.51	5.37	5.55	5.34	2.41%	-0.75%	3.07%
6	5.40	5.28	5.29	5.21	2.24%	2.08%	3.55%
7	5.27	5.13	5.13	5.07	2.76%	2.67%	3.90%
8	5.54	5.44	5.40	5.31	1.82%	2.57%	4.12%
9	5.36	5.29	5.25	5.16	1.36%	2.10%	3.62%
10	5.44	5.35	5.33	5.26	1.59%	2.09%	3.41%
11	6.08	6.01	5.97	5.84	1.10%	1.88%	3.90%
12	6.27	6.20	6.11	5.95	1.26%	2.56%	5.10%
13	6.20	6.10	6.11	6.00	1.56%	1.42%	3.19%
14	6.21	6.14	6.08	6.01	1.22%	2.08%	3.29%
15	6.22	6.10	6.04	5.95	1.92%	2.91%	4.31%
Average	5.77	5.67	5.66	5.56	1.73%	1.97%	3.78%
16	6.64	6.61	6.51		0.40%	1.89%	
17	6.36	6.41	6.28		-0.71%	1.27%	
18	6.23	6.27	6.12		-0.54%	1.89%	
19	6.66	6.72	6.57		-0.98%	1.36%	
Average (Small size instances)	5.96	5.89	5.85		1.10%	1.87%	
20	6.86	6.73	6.61		1.81%	3.64%	
30	7.51	7.52	7.21		-0.10%	4.04%	
40	8.14		7.98			1.98%	
50	8.49		8.29			2.29%	
60	9.07		8.79			3.09%	
Average (Large size instances)	8.01		7.78			2.97%	
Grand Average	6.47		6.33			2.21%	

* 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

	Solomon A (The Depot at the Center)							Solomon B (The Depot at the Corner)						
N	Rollout Value	VFA Value	VFA$^+$ Value	Opt Value	VFA Imprv.*%	VFA$^+$ Imprv.*%	Opt Imprv.*%	Rollout Value	VFA Value	VFA$^+$ Value	Opt Value	VFA Imprv.*%	VFA$^+$ Imprv.*%	Opt Imprv.*%
5	1.423	1.406	1.391	1.387	1.23%	2.31%	2.57%	2.811	2.799	2.791	2.779	0.43%	0.74%	1.14%
6	1.645	1.652	1.623	1.612	-0.44%	1.35%	2.01%	2.817	2.776	2.775	2.766	1.46%	1.49%	1.81%
7	1.854	1.849	1.862	1.836	0.22%	-0.44%	0.96%	2.789	2.718	2.712	2.699	2.56%	2.76%	3.22%
8	2.090	2.079	2.106	2.065	0.54%	-0.75%	1.21%	3.047	2.911	2.935	2.890	4.48%	3.70%	5.16%
9	2.290	2.292	2.285	2.257	-0.08%	0.20%	1.43%	3.146	3.031	3.060	3.007	3.63%	2.73%	4.42%
10	2.358	2.350	2.336	2.323	0.32%	0.92%	1.47%	3.086	3.054	3.038	2.996	1.04%	1.58%	2.92%
11	2.502	2.513	2.487	2.468	-0.41%	0.60%	1.38%	3.261	3.230	3.217	3.169	0.95%	1.36%	2.83%
12	2.662	2.673	2.642	2.633	-0.39%	0.77%	1.09%	3.387	3.369	3.334	3.300	0.53%	1.57%	2.55%
13	2.714	2.717	2.693	2.673	-0.10%	0.80%	1.51%	3.358	3.335	3.331	3.283	0.67%	0.79%	2.22%
14	2.905	2.921	2.892	2.874	-0.56%	0.44%	1.04%	3.605	3.558	3.590	3.489	1.31%	0.43%	3.22%
15	3.014	3.017	2.981	2.954	-0.12%	1.09%	1.99%	3.673	3.607	3.627	3.519	1.78%	1.25%	4.18%
Average	2.314	2.315	2.300	2.280	-0.05%	0.63%	1.47%	3.180	3.126	3.128	3.082	1.69%	1.64%	3.09%

* 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 algorithms 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 (≤ 19 customers) to the large size instances (≥ 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).

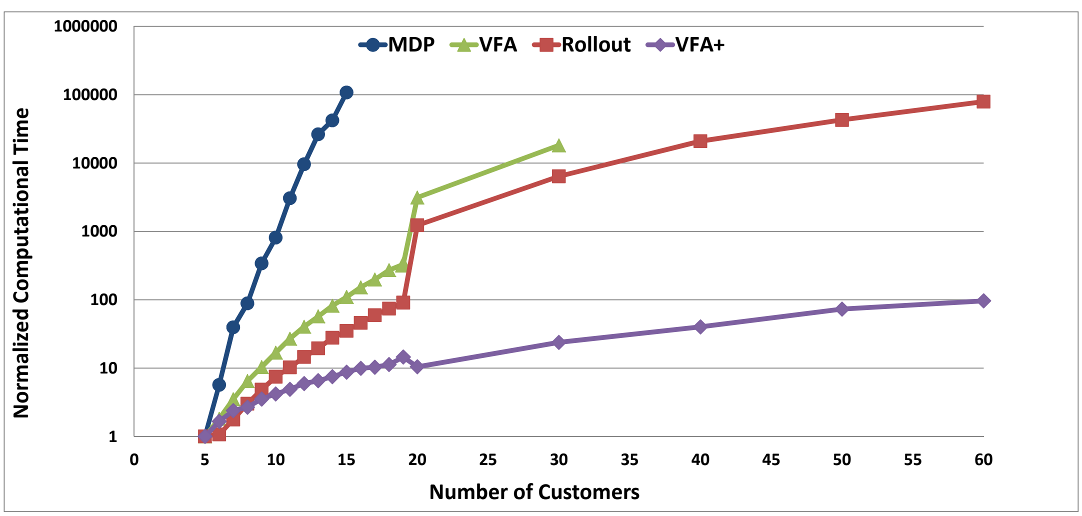

Figure 1 **Computational times for different algorithms based on the Secomandi instances**

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 combinatory 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.

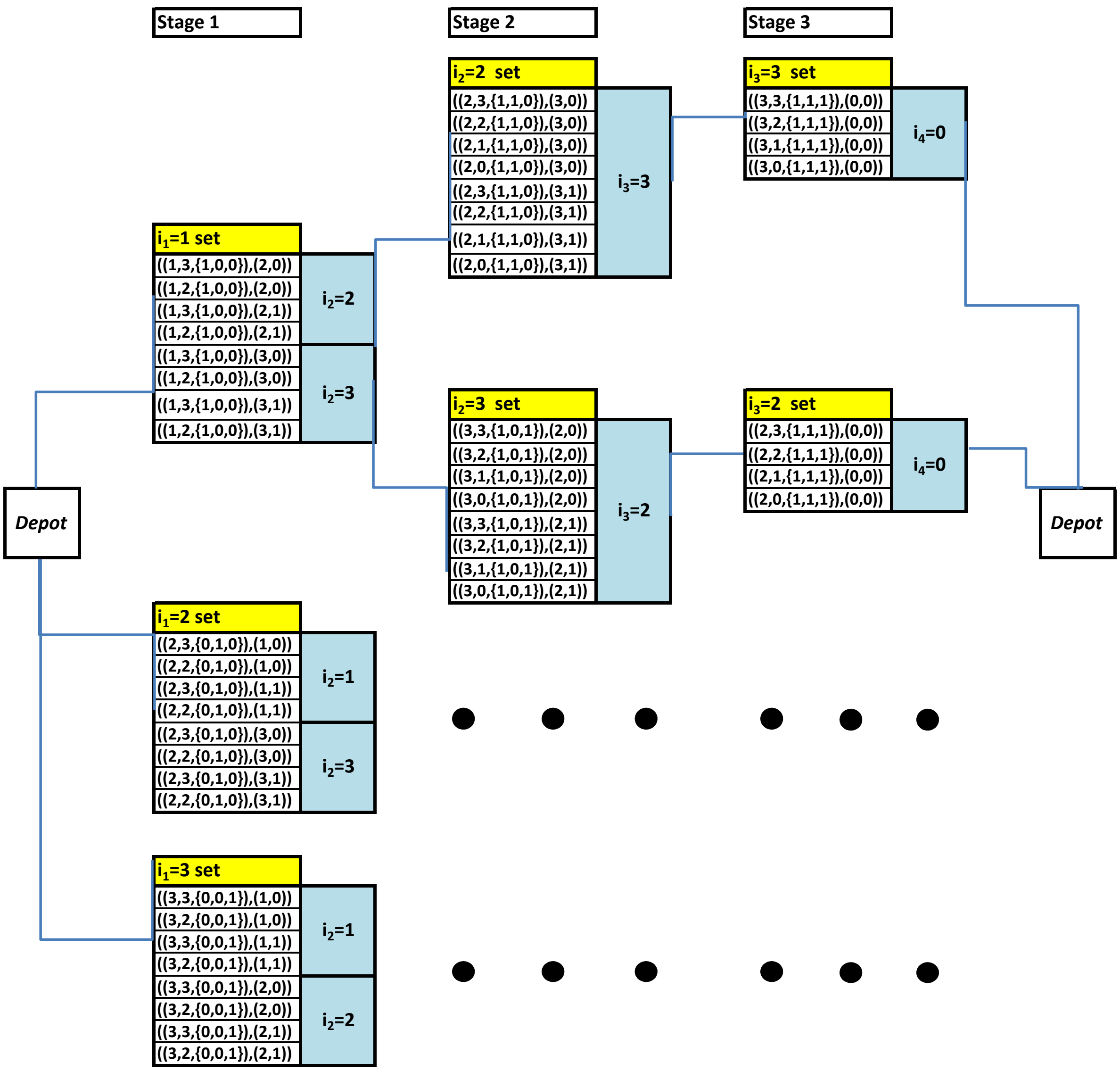

Figure 2 **The illustration for state-decision pairs and i_t sets**

References

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

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

Bertsekas, D. P. 2012. *Dynamic programming and optimal control: Approximate dynamic programming*, vol. 2. 4th ed. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D. P., J. N. Tsitsiklis. 1995. Neuro-dynamic programming: an overview. *Proceedings of the 34th IEEE Conference on Decision and Control*. 560–564.

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

Dror, M., G. Laporte, F. V. Louveaux. 1993. Vehicle routing with stochastic demands and restricted failures. *Mathematical Methods of Operations Research* **37**(3) 273–283.

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

Erera, A. L., J. C. Morales, M. Savelsbergh. 2010. The vehicle routing problem with stochastic demand and duration constraints. *Transportation Science* **44**(4) 474–492.

Fan, J., X. Wang, H. Ning. 2006. A multiple vehicles routing problem algorithm with stochastic demand. *The Sixth World Congress on Intelligent Control and Automation*. 1688–1692.

Fisher, M. L., R. Jaikumar. 1981. A generalized assignment heuristic for vehicle routing. *Networks* **11**(2) 109–124.

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

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

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

George, A., W. B. Powell. 2006. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning* **65** 167–198.

Godfrey, G. A., W. B. Powell. 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science* **36**(1) 21–39.

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

Kenyon, A. S., D. P. Morton. 2003. Stochastic vehicle routing with random travel times. *Transportation Science* **37**(1) 69–82.

Laporte, G. 2007. What you should know about the vehicle routing problem. *Naval Research Logistics* **54**(8) 811–819.

Laporte, G., F. Louveaux, H. Mercure. 1992. The vehicle routing problem with stochastic travel times. *Transportation Science* **26**(3) 161–170.

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

Minis, I., A. Tatarakis. 2011. Stochastic single vehicle routing problem with delivery and pick up anda predefined customer sequence. *European Journal of Operational Research* **213**(1) 37–51.

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

Pandelis, D. G., E. G. Kyriakidis, T. D. Dimitrakos. 2012. Single vehicle routing problems with a predefined customer sequence, compartmentalized load and stochastic demands. *European Journal of Operational Research* **217**(2) 324–332.

Pillac, V., M. Gendreau, C. Guéret, A. L. Medaglia. 2012. A review of dynamic vehicle routing problems. *European Journal of Operational Research* **225**(1) 1–11.

Powell, W. B. 2007. *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley and Sons, Inc., New York.

Powell, W. B. 2011. *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley and Sons, Inc., New York.

Powell, W. B., J. A. Shapiro, H. P. Simo. 2002. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science* **36**(2) 231–249.

Powell, W. B., H. P. Simao, B. Bouzaiene-Ayari. 2012. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal of Transportation and Logistics* **1**(3) 237–284.

Powell, W. B., B. Van Roy. 2004. Approximate dynamic programming for high dimensional resource allocation problems. *Handbook of learning and approximate dynamic programming* 261–280.

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

Simão, H. P, J. Day, A. P. George, T. Gifford, J. Nienow, W. B. Powell. 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science* **43**(2) 178–197.

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

Sutton, R., A. Barto. 1998. *Reinforcement learning*. The MIT Press, Cambridge, MA.

Tatarakis, A., I. Minis. 2009. Stochastic single vehicle routing with a predefined customer sequence and multiple depot returns. *European Journal of Operational Research* **197**(2) 557–571.

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

Working Papers Beta 2009 - 2013

nr.	Year	Title	Author(s)
425	2013	Single Vehicle Routing with Stochastic Demands: Approximate Dynamic Programming	C. Zhang, N.P. Dellaert, L. Zhao, T. Van Woensel, D. Sever
424	2013	Influence of Spillback Effect on Dynamic Shortest Path Problems with Travel-Time-Dependent Network Disruptions	Derya Sever, Nico Dellaert, Tom Van Woensel, Ton de Kok
423	2013	Dynamic Shortest Path Problem with Travel-Time-Dependent Stochastic Disruptions: Hybrid Approximate Dynamic Programming Algorithms with a Clustering Approach	Derya Sever, Lei Zhao, Nico Dellaert, Tom Van Woensel, Ton de Kok
422	2013	System-oriented inventory models for spare parts	R.J.I. Basten, G.J. van Houtum
421	2013	Lost Sales Inventory Models with Batch Ordering And Handling Costs	T. Van Woensel, N. Erkip, A. Curseu, J.C. Fransoo
420	2013	Response speed and the bullwhip	Maximiliano Udenio, Jan C. Fransoo, Eleni Vatamidou, Nico Dellaert
419	2013	Anticipatory Routing of Police Helicopters	Rick van Urk, Martijn R.K. Mes, Erwin W. Hans
418	2013	Supply Chain Finance. A conceptual framework to advance research	Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo
417	2013	Improving the Performance of Sorter Systems By Scheduling Inbound Containers	S.W.A. Haneyah, J.M.J. Schutten, K. Fikse
416	2013	Regional logistics land allocation policies: Stimulating spatial concentration of logistics firms	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
415	2013	The development of measures of process harmonization	Heidi L. Romero, Remco M. Dijkman, Paul W.P.J. Grefen, Arjan van Weele
414	2013	*BASE/X.* Business Agility through Cross-Organizational Service Engineering	Paul Grefen, Egon Lüftenegger, Eric van der Linden, Caren Weisleder

413	2013	The Time-Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times	Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok
412	2013	Clearing the Sky - Understanding SLA Elements in Cloud Computing	Marco Comuzzi, Guus Jacobs, Paul Grefen
411	2013	Approximations for the waiting time distribution In an *M/G/c* priority queue	A. Al Hanbali, E.M. Alvarez, M.C. van der van der Heijden
410	2013	To co-locate or not? Location decisions and logistics concentration areas	Frank P. van den Heuvel, Karel H. van Donselaar, Rob A.C.M. Broekmeulen, Jan C. Fransoo, Peter W. de Langen
409	2013	The Time-Dependent Pollution-Routing Problem	Anna Franceschetti, Dorothée Honhon,Tom van Woensel, Tolga Bektas, GilbertLaporte.
408	2013	Scheduling the scheduling task: A time Management perspective on scheduling	J.A. Larco, V. Wiers, J. Fransoo
407	2013	Clustering Clinical Departments for Wards to Achieve a Prespecified Blocking Probability	J. Theresia van Essen, Mark van Houdenhoven, Johann L. Hurink
406	2013	MyPHRMachines: Personal Health Desktops in the Cloud	Pieter Van Gorp, Marco Comuzzi
405	2013	Maximising the Value of Supply Chain Finance	Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo
404	2013	Reaching 50 million nanostores: retail distribution in emerging megacities	Edgar E. Blanco, Jan C. Fransoo
403	2013	A Vehicle Routing Problem with Flexible Time Windows	Duygu Tas, Ola Jabali, Tom van Woensel
402	2013	The Service Dominant Business Model: A Service Focused Conceptualization	Egon Lüftenegger, Marco Comuzzi, Paul Grefen, Caren Weisleder
401	2012	Relationship between freight accessibility and Logistics employment in US counties	Frank P. van den Heuvel, Liliana Rivera,Karel H. van Donselaar, Ad de Jong,Yossi Sheffi, Peter W. de Langen, Jan C.Fransoo

400	2012	A Condition-Based Maintenance Policy for Multi-Component Systems with a High Maintenance Setup Cost	Qiushi Zhu, Hao Peng, Geert-Jan van Houtum
399	2012	A flexible iterative improvement heuristic to Support creation of feasible shift rosters in Self-rostering	E. van der Veen, J.L. Hurink, J.M.J. Schutten, S.T. Uijland
398	2012	Scheduled Service Network Design with Synchronization and Transshipment Constraints For Intermodal Container Transportation Networks	K. Sharypova, T.G. Crainic, T. van Woensel, J.C. Fransoo
397	2012	Destocking, the bullwhip effect, and the credit Crisis: empirical modeling of supply chain Dynamics	Maximiliano Udenio, Jan C. Fransoo, Robert Peels
396	2012	Vehicle routing with restricted loading capacities	J. Gromicho, J.J. van Hoorn, A.L. Kok J.M.J. Schutten
395	2012	Service differentiation through selective lateral transshipments	E.M. Alvarez, M.C. van der Heijden, I.M.H. Vliegen, W.H.M. Zijm
394	2012	A Generalized Simulation Model of an Integrated Emergency Post	Martijn Mes, Manon Bruens
393	2012	Business Process Technology and the Cloud: Defining a Business Process Cloud Platform	Vasil Stoitsev, Paul Grefen
392	2012	Vehicle Routing with Soft Time Windows and Stochastic Travel Times: A Column Generation And Branch-and-Price Solution Approach	D. Tas, M. Gendreau, N. Dellaert, T. van Woensel, A.G. de Kok
391	2012	Improve OR-Schedule to Reduce Number of Required Beds	J.T. v. Essen, J.M. Bosch, E.W. Hans, M. v. Houdenhoven, J.L. Hurink
390	2012	How does development lead time affect performance over the ramp-up lifecycle?	Andres Pufall, Jan C. Fransoo, Ad de Jong
389	2012	Evidence from the consumer electronics industry	Andreas Pufall, Jan C. Fransoo, Ad de Jong, Ton de Kok

388	2012	The Impact of Product Complexity on Ramp-Up Performance	Frank P.v.d. Heuvel, Peter W.de Langen, Karel H. v. Donselaar, Jan C. Fransoo
387	2012	Co-location synergies: specialized versus diverse logistics concentration areas	Frank P.v.d. Heuvel, Peter W.de Langen, Karel H. v.Donselaar, Jan C. Fransoo
386	2012	Proximity matters: Synergies through co-location of logistics establishments	Frank P. v.d.Heuvel, Peter W.de Langen, Karel H.v. Donselaar, Jan C. Fransoo
385	2012	Spatial concentration and location dynamics in logistics:the case of a Dutch province	Zhiqiang Yan, Remco Dijkman, Paul Grefen
384	2012	FNet: An Index for Advanced Business Process Querying	W.R. Dalinghaus, P.M.E. Van Gorp
383	2012	Defining Various Pathway Terms	Egon Lüftenegger, Paul Grefen, Caren Weisleder
382	2012	The Service Dominant Strategy Canvas: Defining and Visualizing a Service Dominant Strategy through the Traditional Strategic Lens	Stefano Fazi, Tom van Woensel, Jan C. Fransoo
381	2012	A Stochastic Variable Size Bin Packing Problem With Time Constraints	K. Sharypova, T. van Woensel, J.C. Fransoo
380	2012	Coordination and Analysis of Barge Container Hinterland Networks	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
379	2012	Proximity matters: Synergies through co-location of logistics establishments	Heidi Romero, Remco Dijkman, Paul Grefen, Arjan van Weele
378	2012	A literature review in process harmonization: a conceptual framework	S.W.A. Haneya, J.M.J. Schutten, P.C. Schuur, W.H.M. Zijm
377	2012	A Generic Material Flow Control Model for Two Different Industries	H.G.H. Tiemessen, M. Fleischmann, G.J. van Houtum, J.A.E.E. van Nunen, E. Pratsini
375	2012	Improving the performance of sorter systems by scheduling inbound containers	Albert Douma, Martijn Mes

374	2012	Strategies for dynamic appointment making by container terminals	Pieter van Gorp, Marco Comuzzi
373	2012	MyPHRMachines: Lifelong Personal Health Records in the Cloud	E.M. Alvarez, M.C. van der Heijden, W.H.M. Zijm
372	2012	Service differentiation in spare parts supply through dedicated stocks	Frank Karsten, Rob Basten
371	2012	Spare parts inventory pooling: how to share the benefits	X.Lin, R.J.I. Basten, A.A. Kranenburg, G.J. van Houtum
370	2012	Condition based spare parts supply	Martijn Mes
369	2012	Using Simulation to Assess the Opportunities of Dynamic Waste Collection	J. Arts, S.D. Flapper, K. Vernooij
368	2011	Aggregate overhaul and supply chain planning for rotables	J.T. van Essen, J.L. Hurink, W. Hartholt, B.J. van den Akker
367	2011	Operating Room Rescheduling	Kristel M.R. Hoen, Tarkan Tan, Jan C. Fransoo, Geert-Jan van Houtum
366	2011	Switching Transport Modes to Meet Voluntary Carbon Emission Targets	Elisa Alvarez, Matthieu van der Heijden
365	2011	On two-echelon inventory systems with Poisson demand and lost sales	J.T. van Essen, E.W. Hans, J.L. Hurink, A. Oversberg
364	2011	Minimizing the Waiting Time for Emergency Surgery	Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok
363	2011	Vehicle Routing Problem with Stochastic Travel Times Including Soft Time Windows and Service Costs	Erhun Özkan, Geert-Jan van Houtum, Yasemin Serin
362	2011	A New Approximate Evaluation Method for Two-Echelon Inventory Systems with Emergency Shipments	Said Dabia, El-Ghazali Talbi, Tom Van Woensel, Ton de Kok
361	2011	Approximating Multi-Objective Time-Dependent Optimization Problems	Said Dabia, Stefan Röpke, Tom Van Woensel, Ton de Kok

360	2011	Branch and Cut and Price for the Time Dependent Vehicle Routing Problem with Time Window	A.G. Karaarslan, G.P. Kiesmüller, A.G. de Kok
359	2011	Analysis of an Assemble-to-Order System with Different Review Periods	Ahmad Al Hanbali, Matthieu van der Heijden
358	2011	Interval Availability Analysis of a Two-Echelon, Multi-Item System	Felipe Caro, Charles J. Corbett, Tarkan Tan, Rob Zuidwijk
357	2011	Carbon-Optimal and Carbon-Neutral Supply Chains	Sameh Haneyah, Henk Zijm, Marco Schutten, Peter Schuur
356	2011	Generic Planning and Control of Automated Material Handling Systems: Practical Requirements Versus Existing Theory	M. van der Heijden, B. Iskandar
355	2011	Last time buy decisions for products sold under warranty	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
354	2011	Spatial concentration and location dynamics in logistics: the case of a Dutch provence	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
353	2011	Identification of Employment Concentration Areas	Pieter van Gorp, Remco Dijkman
352	2011	BOMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules: extended version	Frank Karsten, Marco Slikker, Geert-Jan van Houtum
351	2011	Resource pooling and cost allocation among independent service providers	E. Lüftenegger, S. Angelov, P. Grefen
350	2011	A Framework for Business Innovation Directions	Remco Dijkman, Irene Vanderfeesten, Hajo A. Reijers
349	2011	The Road to a Business Process Architecture: An Overview of Approaches and their Use	K.M.R. Hoen, T. Tan, J.C. Fransoo G.J. van Houtum
348	2011	Effect of carbon emission regulations on transport mode selection under stochastic demand	Murat Firat, Cor Hurkens
347	2011	An improved MIP-based combinatorial approach for a multi-skill workforce scheduling problem	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
346	2011	An approximate approach for the joint problem of level of repair analysis and spare parts stocking	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
		Joint optimization of level of repair analysis and spare parts stocks	

345	2011	Inventory control with manufacturing lead time flexibility	Ton G. de Kok
344	2011	Analysis of resource pooling games via a new extenstion of the Erlang loss function	Frank Karsten, Marco Slikker, Geert-Jan van Houtum
343	2011	Vehicle refueling with limited resources	Murat Firat, C.A.J. Hurkens, Gerhard J. Woeginger
342	2011	Optimal Inventory Policies with Non-stationary Supply Disruptions and Advance Supply Information	Bilge Atasoy, Refik Güllü, TarkanTan
341	2010	Redundancy Optimization for Critical Components in High-Availability Capital Goods	Kurtulus Baris Öner, Alan Scheller-Wolf Geert-Jan van Houtum
339	2010	Analysis of a two-echelon inventory system with two supply modes	Joachim Arts, Gudrun Kiesmüller
338	2010	Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh	Murat Firat, Gerhard J. Woeginger
335	2010	Attaining stability in multi-skill workforce scheduling	Murat Firat, Cor Hurkens
334	2010	Flexible Heuristics Miner (FHM)	A.J.M.M. Weijters, J.T.S. Ribeiro
333	2010	An exact approach for relating recovering surgical patient workload to the master surgical schedule	P.T. Vanberkel, R.J. Boucherie, E.W. Hans, J.L. Hurink, W.A.M. van Lent, W.H. van Harten
332	2010	Efficiency evaluation for pooling resources in health care	Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Nelly Litvak
331	2010	The Effect of Workload Constraints in Mathematical Programming Models for Production Planning	M.M. Jansen, A.G. de Kok, I.J.B.F. Adan
330	2010	Using pipeline information in a multi-echelon spare parts inventory system	Christian Howard, Ingrid Reijnen, Johan Marklund, Tarkan Tan
329	2010	Reducing costs of repairable spare parts supply systems via dynamic scheduling	H.G.H. Tiemessen, G.J. van Houtum

328	2010	Identification of Employment Concentration and Specialization Areas: Theory and Application	F.P. van den Heuvel, P.W. de Langen, K.H. van Donselaar, J.C. Fransoo
327	2010	A combinatorial approach to multi-skill workforce scheduling	Murat Firat, Cor Hurkens
326	2010	Stability in multi-skill workforce scheduling	Murat Firat, Cor Hurkens, Alexandre Laugier
325	2010	Maintenance spare parts planning and control: A framework for control and agenda for future research	M.A. Driessen, J.J. Arts, G.J. v. Houtum, W.D. Rustenburg, B. Huisman
324	2010	Near-optimal heuristics to set base stock levels in a two-echelon distribution network	R.J.I. Basten, G.J. van Houtum
323	2010	Inventory reduction in spare part networks by selective throughput time reduction	M.C. van der Heijden, E.M. Alvarez, J.M.J. Schutten
322	2010	The selective use of emergency shipments for service-contract differentiation	E.M. Alvarez, M.C. van der Heijden, W.H. Zijm
321	2010	Heuristics for Multi-Item Two-Echelon Spare Parts Inventory Control Problem with Batch Ordering in the Central Warehouse	B. Walrave, K. v. Oorschot, A.G.L. Romme
320	2010	Preventing or escaping the suppression mechanism: intervention conditions	Nico Dellaert, Jully Jeunet.
319	2010	Hospital admission planning to optimize major resources utilization under uncertainty	R. Seguel, R. Eshuis, P. Grefen.
318	2010	Minimal Protocol Adaptors for Interacting Services	Tom Van Woensel, Marshall L. Fisher, Jan C. Fransoo.
317	2010	Teaching Retail Operations in Business and Engineering Schools	Lydie P.M. Smets, Geert-Jan van Houtum, Fred Langerak.
316	2010	Design for Availability: Creating Value for Manufacturers and Customers	Pieter van Gorp, Rik Eshuis.
		Transforming Process Models: executable rewrite	

		rules versus a formalized Java program	
315	2010	Getting trapped in the suppression of exploration: A simulation model	Bob Walrave, Kim E. van Oorschot, A. Georges L. Romme
314	2010	A Dynamic Programming Approach to Multi-Objective Time-Dependent Capacitated Single Vehicle Routing Problems with Time Windows	S. Dabia, T. van Woensel, A.G. de Kok
313	2010		
312	2010	Tales of a So(u)rcerer: Optimal Sourcing Decisions Under Alternative Capacitated Suppliers and General Cost Structures	Osman Alp, Tarkan Tan
311	2010	In-store replenishment procedures for perishable inventory in a retail environment with handling costs and storage constraints	R.A.C.M. Broekmeulen, C.H.M. Bakx
310	2010	The state of the art of innovation-driven business models in the financial services industry	E. Lüftenegger, S. Angelov, E. van der Linden, P. Grefen
309	2010	Design of Complex Architectures Using a Three Dimension Approach: the CrossWork Case	R. Seguel, P. Grefen, R. Eshuis
308	2010	Effect of carbon emission regulations on transport mode selection in supply chains	K.M.R. Hoen, T. Tan, J.C. Fransoo, G.J. van Houtum
307	2010	Interaction between intelligent agent strategies for real-time transportation planning	Martijn Mes, Matthieu van der Heijden, Peter Schuur
306	2010	Internal Slackening Scoring Methods	Marco Slikker, Peter Borm, René van den Brink
305	2010	Vehicle Routing with Traffic Congestion and Drivers' Driving and Working Rules	A.L. Kok, E.W. Hans, J.M.J. Schutten, W.H.M. Zijm
304	2010	Practical extensions to the level of repair analysis	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
303	2010	Ocean Container Transport: An Underestimated and Critical Link in Global Supply Chain Performance	Jan C. Fransoo, Chung-Yee Lee
302	2010	Capacity reservation and utilization for a manufacturer with uncertain capacity and demand	Y. Boulaksil; J.C. Fransoo; T. Tan
300	2009	Spare parts inventory pooling games	F.J.P. Karsten; M. Slikker; G.J. van Houtum
299	2009	Capacity flexibility allocation in an outsourced supply chain with reservation	Y. Boulaksil, M. Grunow, J.C. Fransoo
298	2010	An optimal approach for the joint problem of level of repair analysis and spare parts stocking	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
297	2009	Responding to the Lehman Wave: Sales Forecasting and Supply Management during the Credit Crisis	Robert Peels, Maximiliano Udenio, Jan C. Fransoo, Marcel Wolfs, Tom Hendrikx
296	2009	An exact approach for relating recovering surgical patient workload to the master surgical schedule	Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Wineke A.M. van Lent, Wim H. van Harten

295	2009	An iterative method for the simultaneous optimization of repair decisions and spare parts stocks	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
294	2009	Fujaba hits the Wall(-e)	Pieter van Gorp, Ruben Jubeh, Bernhard Grusie, Anne Keller
293	2009	Implementation of a Healthcare Process in Four Different Workflow Systems	R.S. Mans, W.M.P. van der Aalst, N.C. Russell, P.J.M. Bakker
292	2009	Business Process Model Repositories - Framework and Survey	Zhiqiang Yan, Remco Dijkman, Paul Grefen
291	2009	Efficient Optimization of the Dual-Index Policy Using Markov Chains	Joachim Arts, Marcel van Vuuren, Gudrun Kiesmuller
290	2009	Hierarchical Knowledge-Gradient for Sequential Sampling	Martijn R.K. Mes; Warren B. Powell; Peter I. Frazier
289	2009	Analyzing combined vehicle routing and break scheduling from a distributed decision making perspective	C.M. Meyer; A.L. Kok; H. Kopfer; J.M.J. Schutten
288	2009	Anticipation of lead time performance in Supply Chain Operations Planning	Michiel Jansen; Ton G. de Kok; Jan C. Fransoo
287	2009	Inventory Models with Lateral Transshipments: A Review	Colin Paterson; Gudrun Kiesmuller; Ruud Teunter; Kevin Glazebrook
286	2009	Efficiency evaluation for pooling resources in health care	P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak
285	2009	A Survey of Health Care Models that Encompass Multiple Departments	P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak
284	2009	Supporting Process Control in Business Collaborations	S. Angelov; K. Vidyasankar; J. Vonk; P. Grefen
283	2009	Inventory Control with Partial Batch Ordering	O. Alp; W.T. Huh; T. Tan
282	2009	Translating Safe Petri Nets to Statecharts in a Structure-Preserving Way	R. Eshuis
281	2009	The link between product data model and process model	J.J.C.L. Vogelaar; H.A. Reijers
280	2009	Inventory planning for spare parts networks with delivery time requirements	I.C. Reijnen; T. Tan; G.J. van Houtum
279	2009	Co-Evolution of Demand and Supply under Competition	B. Vermeulen; A.G. de Kok
278	2010	Toward Meso-level Product-Market Network Indices for Strategic Product Selection and (Re)Design Guidelines over the Product Life-Cycle	B. Vermeulen, A.G. de Kok
277	2009	An Efficient Method to Construct Minimal Protocol Adaptors	R. Seguel, R. Eshuis, P. Grefen
276	2009	Coordinating Supply Chains: a Bilevel Programming Approach	Ton G. de Kok, Gabriella Muratore
275	2009	Inventory redistribution for fashion products under	G.P. Kiesmuller, S. Minner

		demand parameter update	
274	2009	Comparing Markov chains: Combining aggregation and precedence relations applied to sets of states	A. Busic, I.M.H. Vliegen, A. Scheller-Wolf
273	2009	Separate tools or tool kits: an exploratory study of engineers' preferences	I.M.H. Vliegen, P.A.M. Kleingeld, G.J. van Houtum
272	2009	An Exact Solution Procedure for Multi-Item Two-Echelon Spare Parts Inventory Control Problem with Batch Ordering	Engin Topan, Z. Pelin Bayindir, Tarkan Tan
271	2009	Distributed Decision Making in Combined Vehicle Routing and Break Scheduling	C.M. Meyer, H. Kopfer, A.L. Kok, M. Schutten
270	2009	Dynamic Programming Algorithm for the Vehicle Routing Problem with Time Windows and EC Social Legislation	A.L. Kok, C.M. Meyer, H. Kopfer, J.M.J. Schutten
269	2009	Similarity of Business Process Models: Metics and Evaluation	Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Kaarik, Jan Mendling
267	2009	Vehicle routing under time-dependent travel times: the impact of congestion avoidance	A.L. Kok, E.W. Hans, J.M.J. Schutten
266	2009	Restricted dynamic programming: a flexible framework for solving realistic VRPs	J. Gromicho; J.J. van Hoorn; A.L. Kok; J.M.J. Schutten;

Working Papers published before 2009 see: http://beta.ieis.tue.nl