TRANSPORTATION SCIENCE
Vol. 45, No. 3, August 2011, pp. 346–363
ISSN 0041-1655 | EISSN 1526-5447 | 11 | 4503 | 0346

informs®

DOI 10.1287/trsc.1100.0353
© 2011 INFORMS

Constructive Heuristics for the Multicompartment Vehicle Routing Problem with Stochastic Demands

Jorge E. Mendoza
Équipe Optimisation des Systèmes de Production et Logistiques, LISA (EA CNRS 4094),
Université Catholique de l'Ouest, 49008 Angers, France,
jorge.mendoza@uco.fr, http://www.uco.fr/~jmendoza

Bruno Castanier, Christelle Guéret
Équipe Systèmes Logistiques et de Production, IRCCyN (UMR CNRS 6597), École des Mines de Nantes,
44307 Nantes Cedex 3, France {bruno.castanier@emn.fr, christelle.gueret@emn.fr}

Andrés L. Medaglia, Nubia Velasco
Centro para la Optimización y Probabilidad Aplicada (COPA), Departamento de Ingeniería Industrial,
Universidad de los Andes, AA 4976 Bogotá, Colombia {amedagli@uniandes.edu.co,
http://wwwprof.uniandes.edu.co/~amedagli, nvelasco@uniandes.edu.co}

The vehicle routing problem with stochastic demands (VRPSD) consists of designing transportation routes of minimal expected cost to satisfy a set of customers with random demands of known probability distribution. This paper tackles a generalization of the VRPSD known as the multicompartment VRPSD (MC-VRPSD), a problem in which each customer demands several products that, because of incompatibility constraints, must be loaded in independent vehicle compartments. To solve the problem, we propose three simple and effective constructive heuristics based on a stochastic programming with recourse formulation. One of the heuristics is an extension to the multicompartment scenario of a savings-based algorithm for the VRPSD; the other two are different versions of a novel look-ahead heuristic that follows a route-first, cluster-second approach. In addition, to enhance the performance of the heuristics these are coupled with a post-optimization procedure based on the classical 2-Opt heuristic. The three algorithms were tested on instances of up to 200 customers from the MC-VRPSD and VRPSD literature. The proposed heuristics unveiled 26 and 12 new best known solutions for a set of 180 MC-VRPSD problems and a 40-instance testbed for the VRPSD, respectively.

Key words: vehicle routing; stochastic demands; stochastic programming; look-ahead heuristics; multicompartment vehicle routing
History: Received: October 2009; revision received: July 2010; accepted: October 2010. Published online in *Articles in Advance* February 18, 2011.

1. Introduction

Vehicle routing problems (VRPs) consist of designing a set of minimal-cost vehicle routes to serve the demands for goods or services of a group of geographically spread customers, satisfying a group of operational constraints. Toth and Vigo (2002), Cordeau et al. (2006), Golden, Raghavan, and Wasil (2008), and Laporte (2009) present comprehensive surveys on VRP variants and solution methods. Classical VRPs normally assume perfect knowledge of problem parameters such as customer locations and demands. However, in practice, the nature of these parameters is often stochastic, meaning that they are affected by different sources of uncertainty that should be considered during the route planning process. Consequently, in recent years a number of research efforts have been addressed to solve VRP variants, known as stochastic VRPs (SVRPs), that incorporate uncertainty in the problem parameters. Among SVRPs, probably the most studied variant is the VRP with stochastic demands (VRPSD), a problem where customer demands are modeled as random variables. For a complete overview of SVRPs the reader is referred to Gendreau, Laporte, and Séguin (1996a).

The main effect of the stochastic demands is that if their realizations (actual values) turn out to be larger than expected, the vehicle capacity might be exceeded, leading to what is commonly known as a *route failure*. Different solution frameworks can be applied to deal with uncertain demands in the VRPSD. A taxonomy of these frameworks classifies them into *dynamic* or *static* (Secomandi and Margot 2009). Routing decisions in dynamic approaches are made in multiple stages and based on demand realizations (at former stages). Depending on the remaining vehicle capacity and the set of unvisited customers, routes are *re-optimized* at each stage. On the other hand, routing decisions in static approaches, once made, remain unchanged regardless of the demand realizations. While dynamic strategies produce more accurate solutions, static strategies are preferred from the computational point of view because the problem, known to be computationally intractable, is solved only once (Laporte, Louveaux, and Van Hamme 2002). Static strategies are also particularly useful when a stable solution is sought (Bertsimas 1992) or when re-optimization during route execution is impossible due to the lack of information. Most of the research in the VRPSD has been based on static strategies.

Among the static approaches, the most widely explored in the VRPSD literature is the stochastic programming with recourse (SPR) framework. SPR solves the problem in two phases: in the first phase a set of routes is planned, and in the second phase the planned routes are executed. In case of a route failure, a problem-dependent predefined corrective action, known as *recourse*, is taken to recover the solution feasibility. In general, the route failures generate an extra cost that is known only after the second phase. Thus, the objective of the problem is to design during the first phase a set of routes that minimizes the sum of the cost of the planned routes and the expected cost of the route failures.

Exact methods for the VRPSD based on SPR include that of Laporte, Louveaux, and Van Hamme (2002), who proposed a branch-and-bound method based on the integer L-shaped algorithm for a formulation where the recourse action is a round trip to the depot that restores the vehicle capacity. Later, Christiansen and Lysgaard (2007) proposed a branch-and-price approach that proved to be competitive in tight instances of the problem, that is, when the ratio between the sum of the expected demands of all customers and the maximum number of available vehicles is close to the vehicle capacity. More recently, Rei, Gendreau, and Soriano (2007) tackled the single-route VRPSD (SVRPSD). Their method consists of using local branching (Fischetti and Lodi 2003) to generate optimality cuts on an integer L-shaped algorithm. Although successful, these approaches are limited to solve instances of up to 100 customer nodes. Consequently, industrial-scale problems are out of reach and should be targeted using heuristics (Gendreau, Laporte, and Séguin 1996a).

The pioneering SPR-based heuristic for the VRPSD is the savings algorithm of Dror and Trudeau (1986); an extension of the Clarke and Wright heuristic (CW) for a formulation where the recourse action is to perform upon failure an emergency trip from the depot to every customer remaining on the planned route. Later, Savelsbergh and Goetschalckx (1995) developed a two-phase approach to solve an SPR formulation that allows only one failure per route. In the first phase, their method extends the generalized assignment heuristic of Fisher and Jaikumar (1981) to solve the deterministic counterpart of the problem using the expected demand for each customer. This deterministic solution is improved in the second stage using post-optimization procedures specially tailored to handle the stochastic nature of the demands. More recently, Yang, Mathur, and Ballou (2000) tackled a variant of the problem with *restocking*. The principle behind the restocking policy is to decide upon the remaining capacity after completing the service of each customer and perform preventive trips to the depot to avoid future route failures. These authors introduced two constructive heuristics based on insertion procedures and a dynamic programming algorithm for tour splitting. Lately, Liu and Lai (2009) introduced a sweep algorithm for an SPR formulation where the recourse action considers cooperation between routes and the demand uncertainty is modeled using fuzzy sets. Some authors have addressed the VRPSD and its variants using metaheuristic approaches. Gendreau, Laporte, and Séguin (1996b) proposed *tabustoch*, a tabu search (TS) algorithm designed to tackle a variant of the problem where in addition to the demands customers are also stochastic, that is, they are present (or not) with a given probability. Ak and Erera (2007) tackled a stochastic routing problem that allows cooperative strategies using TS. Bianchi et al. (2006) proposed a set of algorithms for the SVRPSD with restocking, comprising simulated annealing (SA), iterated local search (ILS), ant colony optimization (ACO), evolutionary algorithms (EA), and TS.

This paper tackles a generalization of the VRPSD known as the multicompartment VRPSD (MC-VRPSD), a problem where each customer has a stochastic demand for different products that, due to incompatibility constraints, must be transported in independent vehicle compartments. The MC-VRPSD naturally arises in several practical situations. For instance, dairies often use vehicles with multiple compartments to collect milk of different types (e.g., from cows and goats) and qualities (e.g., different suckling dates); petroleum companies deliver different types of fuel to outlet retailers using multicompartment tankers; public utilities use trucks with compartments to perform selective waste collection; and food companies distribute in compartmentalized vehicles groceries that require different levels of refrigeration.

Despite the numerous practical applications, research in vehicle routing problems with multiple compartments is rather scarce, and much of the effort has concentrated on deterministic demands. Research along this line has focused mainly on two problem categories based on two different service policies commonly found in practical settings. Under the first policy, the demand for each product can be served by different routes as long it is fully served in a single visit. Some of the solution approaches for problems that fall into this category include the TS and memetic algorithm (MA) of El Fallahi, Prins, and Wolfler-Calvo (2008), the guided local search (GLS) algorithms of Muyldermans and Pang (2010a, b), the heuristic components of Derigs et al. (2010), and the multistart local search procedures recently proposed by Caramia and Guerriero (2010). On the other hand, under the second policy, the demand for every product must be served by the same route (i.e., service splitting is not allowed). Early work on problems falling into the latter category include the constructive heuristic of Van der Bruggen, Gruson, and Salomon (1995), followed by the more recent Lagrangean heuristic of Chajakis and Guignard (2003), the implicit enumeration algorithm of Ruiz, Maroto, and Alcaraz (2004), and the TS and MA of El Fallahi, Prins, and Wolfler-Calvo (2008). Likewise, the few attempts that have been made to solve the MC-VRPSD also fall into the latter service policy. Tatarakis and Minis (2009) tackled the problem of selecting the optimal restocking plan in a single-route MC-VRPSD in which the sequence of customers (route) is fixed beforehand. The authors propose a set of dynamic programming algorithms and solve to optimality problems of up to 15 customers. Mendoza et al. (2010) proposed two MAs based on an SPR formulation of the problem. The first algorithm (MA/SCS) solves the deterministic counterpart of the problem using the expected demands, while trying to avoid route failures by preserving on each compartment some spare capacity in case of high demand realizations. On the other hand, the second memetic algorithm (simply labeled MA) incorporates explicitly the stochastic nature of the demands in the route planning process. Their computational experiments showed that MA outperforms MA/SCS in terms of solution quality (accuracy), while MA/SCS proved to be more efficient in terms of computational performance (speed).

To expand the existing toolbox for solving the MC-VRPSD, under the single-vehicle service scenario, this research introduces three new constructive heuristics that focus not only on accuracy and speed but also on simplicity and flexibility. The latter two design issues are commonly neglected in the academic literature but often hinder the VRP technology transfer to commercial software and real-world applications (Cordeau et al. 2002). One of the proposed heuristics is an extension to the multicompartment scenario of the savings-based approach for the VRPSD of Dror and Trudeau (1986), while the other two algorithms are variants of a novel look-ahead heuristic. The look-ahead heuristic integrates the *pilot method* (Voß, Fink, and Duin 2005) with a route-first, cluster-second procedure. The routing phase uses simple constructive heuristics for the traveling salesman problem (TSP) to produce a giant tour with a sequence of customer visits. Then, the clustering phase splits the giant tour into feasible routes for the MC-VRPSD using the dynamic programming principle. Finally, to enhance the performance of the three constructive heuristics we propose a simple post-optimization procedure based on the classical 2-Opt improvement algorithm.

The remainder of the paper is organized as follows. Section 2 formally states the problem and formulates it as an SPR program. Section 3 introduces an expression to efficiently approximate the objective function in the SPR formulation. Sections 4 and 5 outline the proposed heuristics, and §6 describes the post-optimization procedure. Section 7 presents the computational experiments conducted on a set of benchmark instances and compares the heuristics against a competing memetic algorithm proposed in the literature for the MC-VRPSD. Also in §7 we compare the heuristics with previously published results for the VRPSD. Finally, §8 concludes the paper and outlines future research perspectives.

2. Problem Formulation

Formally, the MC-VRPSD can be defined on a complete and undirected graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{0, \ldots, n\}$ is the vertex set and $\mathcal{E}$ the edge set. Vertices $v = 1, \ldots, n$ represent the customers and vertex $v = 0$ represents the depot. A distance d_e is associated to edge $e = (u, v) = (v, u) \in \mathcal{E}$, and it represents the travel cost between vertices u and v. There exists a set $\mathcal{P} = \{1, \ldots, p, \ldots, m\}$ of products that must be transported in independent compartments of fixed capacity Q_p. All vehicles are identical, and the fleet size is unlimited. For product p, customer v has an independent random demand $\xi_{v,p}$ following a known probability distribution with mean $\mu_{v,p}$ and standard deviation $\sigma_{v,p}$. The demand realizations $\vec{\xi}$ are nonnegative and less than the capacity of the corresponding compartment Q_p, yet they are known only upon the vehicle's arrival to the customer location. Each customer must be visited only once by exactly one vehicle (route), and the total length of each route cannot exceed a maximum distance L. Henceforth, without loss of generality the discussion is restricted to the case of collection routes, nonetheless the case of delivery routes is equivalent.

The MC-VRPSD formulation as a two-stage stochastic programming model follows. In the first stage, a set $\mathcal{R}$ of *a priori* or *planned* routes is designed. Each route $r \in \mathcal{R}$ is a sequence of vertices $r = (0, v_1, \ldots, v_i, \ldots, v_{n_r}, 0)$, where $v_i \in \mathcal{V}\backslash\{0\}$

and n_r is the number of customers serviced by the route. Depending on the context, we refer to the route $r \in \mathcal{R}$ also as an ordered set of arcs $r = \{(0, v_1), \ldots, (v_{i-1}, v_i), \ldots, (v_{n_r}, 0)\}$. In the second stage, each planned route is executed until a route failure occurs, that is, whenever the capacity of at least one compartment is exceeded. Upon failure, the compartment is loaded up to its capacity and the recourse action takes place. The recourse action is defined as a return trip to the depot to unload the compartments, followed by a trip back to the customer location to complete the service. After service completion, the route is resumed from that point on as originally planned. It is assumed that there is no practical way to keep track of the capacity available on each compartment, thus preventive trips to the depot (restocking) are not allowed. The second-stage solution is then the actual set of routes traveled by the vehicles. The problem is to determine in the first stage the set of planned routes $\mathcal{R}$ that minimizes the expected cost $E[C]$ of the second stage solution given by

$$E[C] = \sum_{r\in\mathcal{R}} E[l_r + G_r(\vec{\xi})], \tag{1}$$

$$= \sum_{r\in\mathcal{R}} l_r + \sum_{r\in\mathcal{R}} E[G_r(\vec{\xi})], \tag{2}$$

where l_r denotes the planned length (planned cost) and $E[G_r(\vec{\xi})]$ the expected length of the returning trips to the depot, or cost of recourse, caused by route failures for each route $r \in \mathcal{R}$. Because the total travel distance of each route is a random variable whose value is known only when the vehicle returns to the depot after completing the route, we model the distance constraint as

$$l_r + E[G_r(\vec{\xi})] \le L \quad \forall\, r \in \mathcal{R}. \tag{3}$$

3. Estimating the Total Expected Cost of a MC-VRPSD Solution

In contrast to deterministic VRPs, the overall cost expression in (2) is not straightforward to calculate. While the planned cost of route r is easily computed by the sum of the distance covered by the arcs traversed in the route ($l_r = \sum_{(u,v)\in r} d_{(u,v)}$), the computation of the cost of recourse $E[G_r(\vec{\xi})]$ is far more complex. The cost of recourse of route r is given by

$$E[G_r(\vec{\xi})] = \sum_{i=1}^{n_r} 2 \times d_{v_i,0} \times \Pr(v_i), \tag{4}$$

where $\Pr(v_i)$ is the failure probability while servicing each customer $v_i \in r$.

In the multicompartment scenario and under the selected recourse action, the failure probability while servicing a given customer v_i, depends on the location (customer) where the last failure occurred and the demands collected for each product since that last failure. As demonstrated in Appendix 8, the value of $\Pr(v_i)$ is given by

$$\Pr(v_i) = \sum_{j=0}^{i-1} \Bigg[\prod_{p\in\mathcal{P}} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p\right) - \prod_{p\in\mathcal{P}} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i} \xi_{v_k,p} \le Q_p\right)\Bigg] \times \Pr(v_j), \tag{5}$$

where v_j is the customer being serviced when the last failure occurred and $\xi'_{v_j,p}$ is customer v_j's unserviced demand for product p due to the failure. Consequently, the first product denotes the probability that the collected demand after the failure at customer v_j and up to customer v_{i-1} does not exceed the capacity of the corresponding compartment Q_p. Similarly, the second product denotes the probability that the capacities are not exceeded after servicing customer v_i.

The calculation of $\Pr(v_i)$ in (5) requires the evaluation of the probability functions of all unserviced demands (i.e., $\xi'_{v_j,p}$, $\forall\, j < i$, $p \in \mathcal{P}$), which might be impossible to perform analytically, depending on the probability function of the demands. Alternatively, one could use estimations calculated by simulation as suggested by Tan, Cheonga, and Goh (2007); Sörensen and Sevaux (2009); and Mendoza et al. (2009) for related routing problems with stochastic demands. Although effective, the simulation approach might be overly expensive, from a computational point of view, to be embedded within fast heuristics such as the ones addressed in this work. To overcome this difficulty, we use a fast approximation that can be analytically computed for a number of demand probability distributions. The approximation is based on a *take-all policy* (TAP), which assumes that even if a route failure occurs at customer v_j, the whole demand for all products is serviced before performing the return trip; that is, making $\xi'_{v_j,p} = 0$, $\forall\, p \in \mathcal{P}$ in (5). Nonetheless, after unloading the compartments at the depot, the vehicle must return to the customer being served when the failure occurred.

Extensive computational experiments show that if the capacity of the compartments is sufficiently large compared to the largest expected demand for the corresponding product, which is the case in many real-world applications, the true cost of recourse in (4) can be estimated within an average error of 3.00% by the TAP approximation. This small gap suggests that the approximation is an acceptable proxy of the overall cost and can be used to guide the search in the proposed heuristics. For a thorough discussion on the computational experiments conducted to validate the TAP approximation, the reader is referred to Mendoza et al. (2008).

4. Stochastic Clarke and Wright Heuristic

The stochastic Clarke and Wright (SCW) heuristic extends the savings-based algorithm introduced by Dror and Trudeau (1986) for the VRPSD in several ways; namely, SCW: (i) works in a multicompartment scenario, (ii) considers the round trip to the depot recourse action, and (iii) deals with the distance constraint of the routes. Similarly to the classical savings algorithm, the SCW heuristic starts from a trivial solution comprised of n round trips from the depot to each customer. Then it tries at each iteration to merge two routes by their extreme vertices (those connected to the depot), aiming to generate the largest possible *savings* in the overall cost. In the SPR formulation introduced in §2, the savings generated by the merge of two routes having customers v and u as extreme points is given by $s_{v,u} = E[C_v] + E[C_u] - E[C_{u,v}]$, where $E[C_v]$ and $E[C_u]$ are the expected costs of the routes servicing customers v and u, and $E[C_{u,v}]$ is the expected cost of the merged route that visits customer u before customer v. Because the failure probability at customer v depends on the demands collected at the preceding customers (see (5)), and as it is often the case that demands collected from the depot to a given customer v differ from those collected from customer v to the depot, the failure probability at customer v might differ upon the traveling direction, leading to different costs of recourse. Consequently, when calculating the savings generated by a merger, both traveling directions should be considered as follows:

$$s_{v,u} = E[C_v] + E[C_u] - \min\{E[C_{u,v}], E[C_{v,u}]\}. \quad (6)$$

Algorithm 1 outlines the logic of our SCW heuristic. Once the single-node routes are generated (steps 2–5), the merging process begins. At each iteration, the savings of all feasible mergers leading to routes that satisfy the distance constraint are computed and stored in the savings list $\mathcal{S}$ (steps 7–17). Then, the largest saving is selected from the list, and the corresponding merger is committed (steps 18–29). The process is repeated until the largest feasible saving has a negative value or no more feasible savings are available (steps 19–21). If the largest saving has a zero value, the corresponding merger is undertaken to save one vehicle, although the value of the overall cost does not improve. In our implementation, the savings in (6) are computed using the TAP approximation in §3.

Algorithm 1 (The stochastic Clarke and Wright heuristic)

```
 1: $\mathcal{R} \leftarrow \varnothing$
 2: for every customer $v \in \mathcal{V}\backslash 0$ do
 3:     create route $r = \{0, v, 0\}$
 4:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
 5: end for
 6: create the savings list $\mathcal{S} \leftarrow \varnothing$
 7: for every route $r \in \mathcal{R}$ do
 8:     $u \longleftarrow v_1 \in r$ and $v \longleftarrow v_{n_r} \in r$
 9:     for every route $r' \in \mathcal{R}\backslash\{r\}$ do
10:         $u' \longleftarrow v_1 \in r'$ and $v' \longleftarrow v_{n_{r'}} \in r'$
11:         calculate $s_{u,u'}, s_{u,v'}, s_{v,u'}, s_{v,v'}$
12:         if $\min\{E[C_{u,u'}], E[C_{u',u}]\} \le L$ then
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{u,u'}\}$
13:         if $\min\{E[C_{u,v'}], E[C_{v',u}]\} \le L$ then
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{u,v'}\}$
14:         if $\min\{E[C_{v,u'}], E[C_{u',v}]\} \le L$ then
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{v,u'}\}$
15:         if $\min\{E[C_{v,v'}], E[C_{v',v}]\} \le L$ then
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{v,v'}\}$
16:     end for
17: end for
18: $s_{\max} = \max_{(u,v)\in\mathcal{V}\backslash\{0\}\times\mathcal{V}\backslash\{0\}} \{s_{u,v} \mid s_{u,v} \in \mathcal{S}\}$
19: if $s_{\max} < 0$ or $\mathcal{S} = \varnothing$ then
20:     stop
21: end if
22: if $s_{\max} \ge 0$ then
23:     $(u^*, v^*) \leftarrow \arg\max_{(u,v)\in\mathcal{V}\backslash\{0\}\times\mathcal{V}\backslash\{0\}}$
            $\{s_{u,v} \mid s_{u,v} \in \mathcal{S}\}$
24:     merge routes having $u^*$ and $v^*$ as extreme
            vertices $r \leftarrow merge(r(u^*), r(v^*))$
25:     $\mathcal{R} \leftarrow \mathcal{R}\backslash\{r(u^*)\}$
26:     $\mathcal{R} \leftarrow \mathcal{R}\backslash\{r(v^*)\}$
27:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
28:     go to 6
29: end if
```

5. Look-Ahead Heuristic

The next method is a look-ahead heuristic that embeds a route-first, cluster-second (RC) approach. The RC approach for VRPs builds a single giant tour, starting and ending at the depot after visiting every customer (routing phase), then it splits the tour into routes that satisfy the particular constraints of the VRP on hand (clustering phase). In the proposed heuristic, the routing phase uses different construction algorithms for the deterministic TSP (routing procedures), while the clustering phase applies a dynamic programming algorithm specially designed to tackle the stochastic nature of demands in the multicompartment scenario (clustering procedure). One distinctive feature of our heuristic is that the routing phase is guided by a look-ahead mechanism that avoids traps often posed by the greedy choices of the TSP algorithms by evaluating the impact of local moves in the final solution of the problem. The remainder of this section outlines the building blocks of the RC approach—namely, the routing and clustering procedures—and explains how they are coupled with the look-ahead mechanism to generate the proposed heuristic.

5.1. Routing Procedures

Rei, Gendreau, and Soriano (2010) point out that in stochastic vehicle routing problems the planned cost of a solution normally dominates the cost of recourse. For instance, in the solutions by Mendoza et al. (2010) for a set of 40 MC-VRPSD instances, the planned cost represents 90% of the total cost of the solutions (on average). To build good sequences of customers that translate into good planned routes during the clustering phase, our RC approach uses simple but effective constructive heuristics for the geometric TSP, namely, the *nearest neighbor* (NN) and the *nearest insertion* (NI) heuristics (Bentley 1992). Starting by linking the depot to its closest customer, at each iteration the NN heuristic grows a directed path connecting the last added node to the closest unvisited customer, thus greedily minimizing the cost of adding one more customer to the path. Once every customer is included in the path, the last added vertex is connected to the depot generating the TSP tour. On the other hand, the NI heuristic starts directly from a partial TSP tour beginning and ending at the depot after visiting its farthest customer. The tour grows at each iteration by adding the customer whose insertion generates the smallest increment on the cost of the tour. The insertion procedure is repeated until all customers are visited. We chose NN and NI over the large set of heuristics available for the TSP based on their trade-off between implementation simplicity, accuracy, and computational performance. For an extensive study on TSP heuristics, the reader is referred to Bentley (1992).

5.2. Clustering Procedure

The TSP tours generated in the routing phase are not necessarily good or even feasible solutions for the MC-VRPSD. On one hand, a single tour visiting every customer will face a large number of route failures, thus leading to a large cost of recourse; on the other hand, a single tour will very likely violate the distance constraint. In our RC approach, the giant tours generated in the routing phase are optimally partitioned into feasible solutions for the MC-VRPSD using the *stochastic split* (s-split) procedure (Mendoza et al. 2010) as follows. From the giant tour, an auxiliary graph G' is built to find the optimal partition of the tour into feasible routes. The directed and acyclic graph $G' = (\mathcal{V}', \mathcal{A})$ is composed of the ordered vertex set $\mathcal{V}' = \{0, v_1, \ldots, v_i, \ldots, v_n\}$ and the arc set $\mathcal{A}$. Vertex 0 is an auxiliary vertex and vertices $v_1, \ldots, v_n \in \mathcal{V}'\backslash\{0\}$ in graph G' are arranged so that they follow the same order of the permutation (tour) in the routing phase. Arc $(v_i, v_{i+n_r}) \in \mathcal{A}$ represents a feasible route r with expected cost $E[C_r]$ starting and ending at the depot and traversing the sequence of customers from v_{i+1} to v_{i+n_r}. The s-split procedure finds the set of arcs (i.e., routes) along the shortest path connecting 0 and v_n in G'. Figure 1 illustrates the s-split procedure where a single tour of customers is split into two feasible routes ($L = 70$) of minimum expected cost. For simplicity, the example assumes that the problem parameters are such that independent of the route sequence, the failure probability at a given customer increases by 0.20 with respect to the failure probability of the preceding customer.

To implement the s-split procedure, our RC approach uses Algorithm 2. Based on Bellman's algorithm for directed acyclic graphs, the algorithm splits the TSP tour without generating G' explicitly. For each vertex $v_i \in \mathcal{V}'\backslash\{0\}$ the algorithm maintains two labels, namely, Z_i and B_i. Label Z_i holds the expected cost of the shortest path from 0 to v_i and label B_i holds the predecessor of v_i in the shortest path. The loops in steps 5 and 9 generate all arcs (feasible routes) in $\mathcal{A}$. The expected cost of each route is evaluated (steps 10 to 24), and its value is used to update labels Z_i and B_i (steps 25 to 31). The value of Z_n is the minimal overall expected cost of the MC-VRPSD solution and the corresponding routes can be built using the labels in B_i, starting backward from B_n.

Algorithm 2 (S-split algorithm for tour partitioning)

```
 1: set $Z_0 = 0$ and $Z_i = +\infty$, $1 \le i \le n$
 2: set $B_i = 0$, $1 \le i \le n$
 3: set $\Pr(v_0) = 1$
 4: initialize dummy variables $x = 1$ and $y = 1$
 5: for $i = 1$ to $n$ do
 6:     $l_r = 0$
 7:     $E[G_r(\vec{\xi})] = 0$
 8:     $j = i$
 9:     while $j \le n$ and $l_r + E[G_r(\vec{\xi})] \le L$ do
10:         if $j = i$ then
11:             $l_r = d_{0,v_j} + d_{v_j,0}$
12:         else
13:             $l_r = l_r - d_{v_{j-1},0} + d_{v_{j-1},v_j} + d_{v_j,0}$
14:         end if
15:         $\Pr(v_j) = 0$
16:         for $k = 0$ to $j - 2$ do
17:             $x = 1$, $y = 1$
18:             for product $p = 1$ to $m$ do
19:                 $x = x \times \Pr(\sum_{l=k+1}^{j-1} \xi_{v_l,p} \le Q_p)$
20:                 $y = y \times \Pr(\sum_{l=k+1}^{j} \xi_{v_l,p} \le Q_p)$
21:             end for
22:             $\Pr(v_j) = \Pr(v_j) + (x - y) \times \Pr(v_k)$
23:         end for
24:         $E[G_r(\vec{\xi})] = E[G_r(\vec{\xi})] + 2 \times d_{v_j,0} \times \Pr(v_j)$
25:         if $l_r + E[G_r(\vec{\xi})] \le L$ then
26:             if $Z_{i-1} + l_r + E[G_r(\vec{\xi})] \le Z_j$ then
```

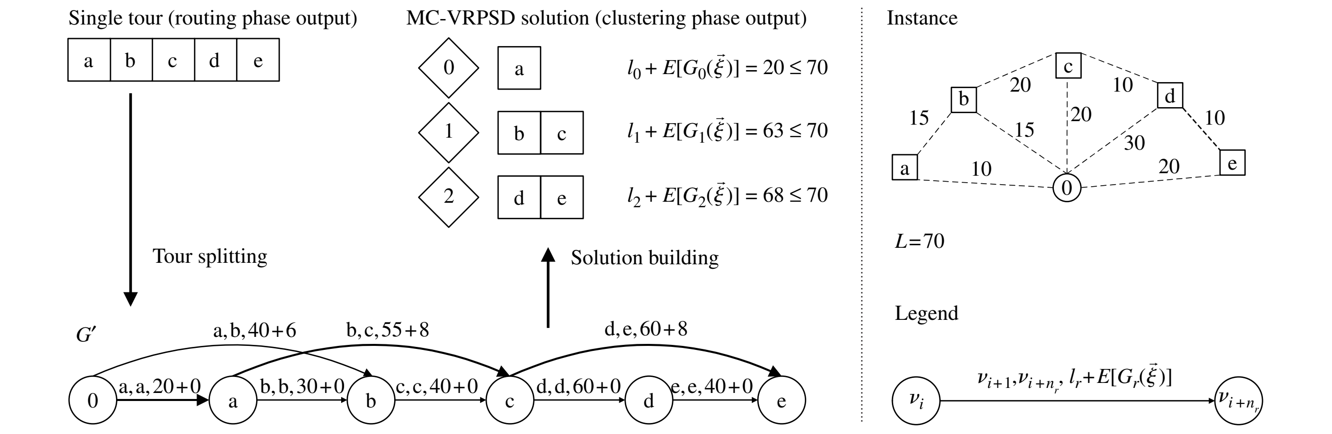
```
27:             $Z_j = Z_{i-1} + l_r + E[G_r(\vec{\xi})]$
28:             $B_j = i - 1$
29:           **end if**
30:           $j = j + 1$
31:         **end if**
32:     **end while**
33: **end for**
```

It is worth mentioning at this point that besides the original s-split procedure we tried two different versions. The first extension splits the single permutation in both directions, trying to capture the effect of the route traveling directions described in §4. The second follows the intuition of Bertsimas' cyclic heuristic (Bertsimas 1992) and generates from the single permutation n different partitions into MC-VRPSD solutions by splitting the cyclic TSP tour starting from every vertex. Based on some preliminary experimentation, we discarded both variants simply because the gain on accuracy did not pay off the loss of simplicity and speed.

5.3. Look-Ahead Mechanism

Because the s-split procedure transforms only the tours generated by NN or NI into feasible MC-VRPSD solutions without affecting the customer visiting order, it is worth highlighting that the most important decisions are made during the earlier routing phase. Moreover, as pointed out by Prins (2004), there exists at least one TSP tour that if known, s-split would translate into the optimal MC-VRPSD solution. Thus the challenge of our RC approach is to find such tour during the routing phase. However, this task, is far from trivial, especially using pure constructive heuristics such as NN and NI. Like most of the constructive heuristics for combinatorial optimization, NN and NI build the final solution (i.e., a TSP tour) by expanding a partial solution at each iteration according to a simple and intuitive decision rule. This greedy approach, however, has a major drawback: the decision rules are based on the impact of decisions on the partial solutions and not on their impact on the final output of the algorithm, often leading to myopic moves. To try to overcome this difficulty we coupled the proposed RC approach with the pilot method (Voß, Fink, and Duin 2005).

The *preferred iterative look-ahead technique* or pilot method works on top of constructive heuristics as a look-ahead mechanism that helps the subordinate


Single tour (routing phase output)
a b c d e
Tour splitting
MC-VRPSD solution (clustering phase output)
0 a $l_0 + E[G_0(\vec{\xi})] = 20 \le 70$
1 b c $l_1 + E[G_1(\vec{\xi})] = 63 \le 70$
2 d e $l_2 + E[G_2(\vec{\xi})] = 68 \le 70$
Solution building
G′
a,b,40+6
b,c,55+8
d,e,60+8
a,a,20+0
b,b,30+0
c,c,40+0
d,d,60+0
e,e,40+0
0 a b c d e
Instance
15 20 10 15 20 30 10 10 20
a b c d e 0
L=70
Legend
v_i
$v_{i+1}, v_{i+n_r}, l_r + E[G_r(\vec{\xi})]$
v_{i+n_r}

Arc (route) evaluation

Arc	Sequence	l_r	$E[G_r(\vec{\xi})]$	$E[C_r]$	$E[C_r] \le L$
a, a	0, a, 0	$10+10=20$	$2\times(0\times10)=0$	20	True
a, b	0, a, b, 0	$10+15+15=40$	$2\times(0\times10+0.2\times15)=6$	46	True
a, c	0, a, b, c, 0	$10+15+20+20=65$	$2\times(0\times10+0.2\times15+0.4\times20)=22$	87	False
a, d	0, a, b, c, d, 0	$10+15+20+10+30=85$	$2\times(0\times10+0.2\times15+0.4\times20+0.6\times30)=58$	143	False
a, e	0, a, b, c, d, e, 0	$10+15+20+10+10+20=85$	$2\times(0\times10+0.2\times15+0.4\times20+0.6\times30+0.8\times20)=90$	175	False
b, b	0, b, 0	$15+15=30$	$2\times(0\times15)=0$	30	True
b, c	0, b, c, 0	$15+20+20=55$	$2\times(0\times15+0.2\times20)=8$	63	True
b, d	0, b, c, d, 0	$15+20+10+30=75$	$2\times(0\times15+0.2\times20+0.4\times30)=32$	107	False
b, e	0, b, c, d, e, 0	$15+20+10+10+20=75$	$2\times(0\times15+0.2\times20+0.4\times30+0.6\times20)=56$	131	False
c, c	0, c, 0	$20+20=40$	$2\times(0\times20)=0$	40	True
c, d	0, c, d, 0	$20+10+30=60$	$2\times(0\times20+0.2\times30)=12$	72	False
c, e	0, c, d, e, 0	$20+10+10+20=60$	$2\times(0\times20+0.2\times30+0.4\times20)=28$	88	False
d, d	0, d, 0	$30+30=60$	$2\times(0\times30)=0$	50	True
d, e	0, d, e, 0	$30+10+20=60$	$2\times(0\times30+0.2\times20)=8$	68	True
e, e	0, e, 0	$20+20=40$	$2\times(0\times20)=0$	40	True

Figure 1 S-Split Procedure for Clustering

procedures to avoid myopic moves. The basic idea of the pilot method is to iteratively build the solution to a problem (master solution) by performing K different local moves (instead of only one) at each iteration, and then evaluating their impact in the final output by using the underlined heuristic to complete the new K partial solutions (pilot solutions) and their merit functions. Then, the local move leading to the best pilot solution is performed in the master solution and a new iteration begins. This process is repeated until the master solution is fully built. The pilot method has been successfully applied to a number of combinatorial optimization problems, including Steiner tree problems (Duin and Voß 1999), the general ring network design problem (Fink, Schneidereit, and Voß 2000), and the flow shop scheduling problem (Fink and Voß 2003). However, to the best of our knowledge this is the first time that a pilot-based approach is applied to a VRP.

In the proposed look-ahead heuristic, the pilot method works on top of the RC approach as shown in Figure 2. At every iteration, the pilot method makes K copies of the master solution, that is, the partial TSP tour being built by the routing procedure (NN or NI), and stores them as partial pilot solutions. Then the pilot method applies the K best local moves on the partial pilot solutions according to the decision rule of the routing procedure, that is, the K nearest neighbors for NN and the K nearest insertions for NI. Using the partial pilot solution as the starting point, the pilot method makes a callback to the routing procedure to complete the generation of the TSP tour and then invokes s-split to perform the clustering phase. Next, the partial pilot solution leading to the best MC-VRPSD solution becomes the new master solution and a new iteration begins. Pilot iterations are carried out until every customer is included in the master solution, then the s-split procedure performs the clustering phase, and the resulting MC-VRPSD solution is the one reported by the look-ahead heuristic. Because the use of NN and NI as routing procedures leads to two different versions of our heuristic, henceforth we will refer to each version using the notation Pilot(RP), where $\text{RP} \in \{\text{NN}, \text{NI}\}$ is the routing procedure used for the routing phase.

Even though NN and NI were the only two heuristics tried on the routing procedure, any of the constructive heuristics for the TSP discussed by Bentley (1992) is also suitable to be embedded into our look-ahead heuristic. Furthermore, s-split inherits from the original Prins' proposition its capability to be easily extended to handle richer side constraints (Prins 2004). Indeed, a number of split procedures for vehicle routing reported in the literature (Villegas et al. 2008; Prins, Labadi, and Reghioui 2009) can replace s-split in the clustering phase, leaving the other building blocks of the look-ahead heuristic untouched. These facts highlight the flexibility and wide potential of the proposed look-ahead heuristic.

6. Stochastic 2-Opt

Constructive heuristics for combinatorial optimization, particulary those for VRPs, build feasible solutions trying to control the objective function without exploring deeply the vast solution space.

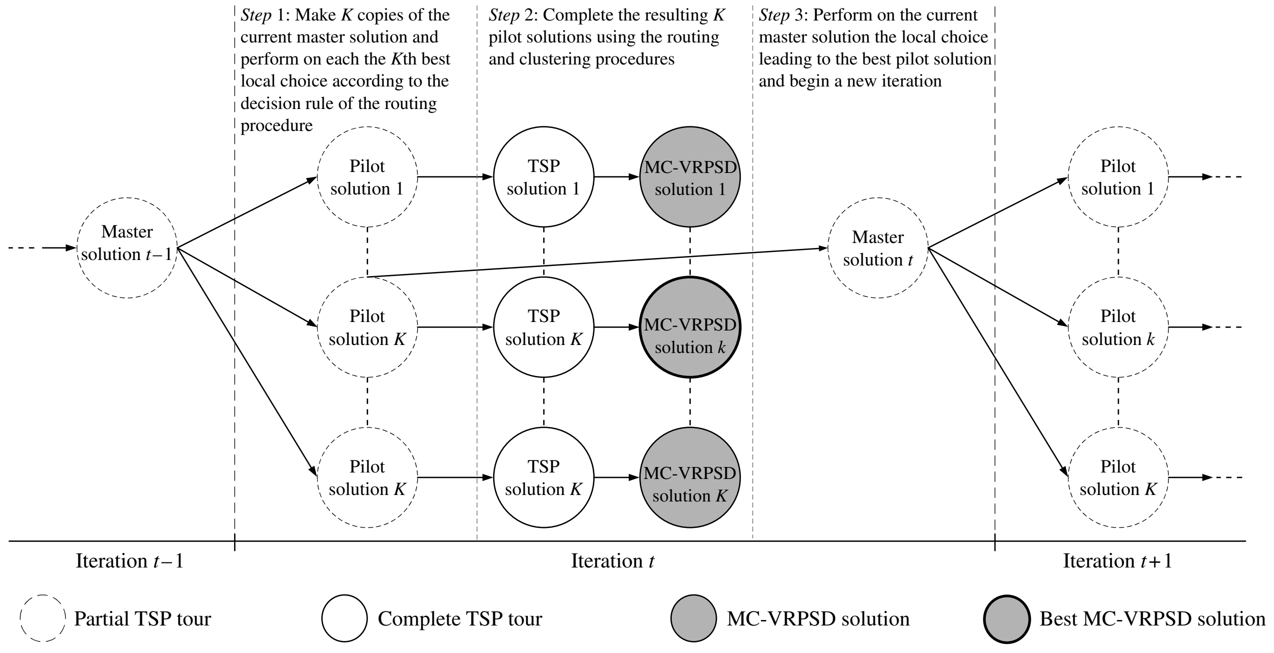

Step 1: Make K copies of the current master solution and perform on each the Kth best local choice according to the decision rule of the routing procedure
Step 2: Complete the resulting K pilot solutions using the routing and clustering procedures
Step 3: Perform on the current master solution the local choice leading to the best pilot solution and begin a new iteration
Master solution $t-1$
Pilot solution 1
Pilot solution K
Pilot solution K
TSP solution 1
TSP solution K
TSP solution K
MC-VRPSD solution 1
MC-VRPSD solution k
MC-VRPSD solution K
Master solution t
Pilot solution 1
Pilot solution k
Pilot solution K
Iteration $t-1$
Iteration t
Iteration $t+1$
Partial TSP tour
Complete TSP tour
MC-VRPSD solution
Best MC-VRPSD solution

Figure 2 Pilot Method on Top of the RC Procedure

Consequently, the use of constructive algorithms only is not advisable to generate competitive solutions for complex problems such as the MC-VRPSD and should be complemented with *post-optimization* or *improvement* methods. Therefore, to enhance the performance of the three proposed heuristics, we implemented a post-optimization procedure based on the well-known 2-Opt improvement heuristic for the TSP.

The 2-Opt heuristic evaluates all possible arc exchanges in a route r by deleting two nonadjacent arcs (u, v) and (u', v') and reconnecting the remaining route segments inserting arcs (u, u') and (v, v'). If an improving move (arc exchange) is found—that is, the resulting route is feasible and has a cost lower than the original route—the move is committed and the procedure restarted. The whole operation is repeated until no more improving arc exchanges are found. For further details on the 2-Opt heuristic, the reader is referred to Hansen and Mladenović (2006). However, applying 2-Opt to MC-VRPSD solutions is not straightforward because each arc exchange reverses the visiting order of a subsequence of customers, thus affecting the failure probability of all the customers in the tour. Consequently, the impact of a move on the cost and the feasibility of a solution can only be evaluated after performing the computationally expensive reestimation of the expected cost of the route. To address this issue, we designed a special procedure that avoids expensive evaluations of unpromising 2-Opt moves yet still explores a sufficiently large number of exchanges.

The proposed post-optimization procedure, henceforth called stochastic 2-Opt (S2-Opt), is described in Algorithm 3. At every iteration, routes r and r' from the MC-VRPSD solution ($\mathcal{R}$) are merged into a single tour, and all possible arc exchanges in the resulting route are explored (loop in 5). To avoid excessive computations, every exchange is first evaluated solely on the basis of the deterministic part of the overall cost, which can be done in constant time as in the classical 2-Opt procedure. If the exchange is promising, meaning that the planned cost of the MC-VRPSD solution improves (step 7), the move is then evaluated in terms of the whole overall cost using s-split (step 8). The tour partitioning process generates a temporary set of feasible routes $\mathcal{T}$ serving the customers originally visited by r and r'. If the expected cost of the new set $\mathcal{T}$ is lower than the sum of those of the two original routes r and r' (step 9), the move improves the current MC-VRPSD solution because the expected cost of the other routes (i.e., $\mathcal{R}\backslash\{r, r'\}$) remains unchanged. In such case, we deem the move successful, and the MC-VRPSD solution is updated (step 10). Finally, the procedure is restarted from the top following a *first improvement* configuration. The S2-Opt procedure repeats until it cannot find any more improvements.

Algorithm 3 (Stochastic 2-Opt procedure for post-optimization)

```
 1: for every route $r \in \mathcal{R}$ do
 2:     for every route $r' \in \mathcal{R}\backslash r$ do
 3:        concatenate copies of $r$ and $r'$ to generate $r''$
 4:        let $\mathcal{W}$ be the set of all possible 2-Opt
              exchanges in $r''$
 5:        for every exchange $w \in \mathcal{W}$ do
 6:           make a copy of $r''$ and perform $w$
                 to generate $r'''$
 7:           if $l_{r'''} < l_{r''}$ then
 8:              call s-split and partition $r'''$ to
                    generate $\mathcal{T}$
 9:              if $E[C(\mathcal{T})] < E[C_r] + E[C_{r'}]$ then
10:                 $\mathcal{R} \leftarrow \mathcal{T} \cup \mathcal{R}\backslash\{r, r'\}$
11:                 go to 1
12:              end if
13:           end if
14:        end for
15:     end for
16: end for
```

7. Computational Experiments

We implemented the proposed heuristics in Java, compiled the code using the Java 1.6 compiler embedded in the Eclipse Java Development Tools (JDT) build I20080617-2000, and ran all the experiments on the Java Runtime Environment (JRE) build 1.6.0_11-b03. Our implementation uses stochastic simulation in Java (SSJ) by L'Ecuyer, Meliani, and Vaucher (2002) for the calculation of probabilities. On its current version, our code supports normal and Poisson distributions for the demands; nonetheless, it can be easily extended to handle other probability distributions having the *cumulative property*. This property states that the sum of two or more independent and Ψ distributed random variables is also Ψ distributed, as it is the case for the normal, Poisson, and Gamma distributions (Christiansen and Lysgaard 2007; Law and Kelton 2000). The remainder of this section details the computational experiments carried out to assess the proposed heuristics on instances from the MC-VRPSD and VRPSD literature.

7.1. MC-VRPSD Instances

To evaluate the power of the proposed heuristics to solve the MC-VRPSD, we used the two sets of randomly generated instances proposed by Mendoza et al. (2010), namely, *tuning set* and *test set*. The former is used to fine tune the value of K in the pilot-based heuristics, while the latter is used to compare the performance of the three algorithms. For completeness, the two sets of instances are briefly described below.

The tuning set consists of 180 instances generated with the following characteristics. First, 50, 100, and 200 customers and the depot were randomly distributed over a 100×100 Euclidean space. Each client v has a demand for three different products ($p = 1, 2, 3$) following a normal distribution $N(\mu_{v,p}, \sigma_{v,p})$, where $\mu_{v,p}$ takes the value of 10 or 30 and $\sigma_{v,p}$ is set such that the coefficient of variation $CV = \sigma_{v,p}/\mu_{v,p}$ is either 0.1 or 0.3. The capacity of the compartments was fixed by defining a *tightness ratio* $(\sum_{v \in \mathcal{V}\setminus\{0\}} \mu_{v,p})/Q_p$ equal to 10 (Bianchi et al. 2006). Finally, the maximum distance per route was set to $L = \beta \times \max_{v \in \mathcal{V}\setminus\{0\}} d_{0,v}$, where β is uniformly distributed between 3 and 4. Thirty instances were generated for each combination of number of customers and coefficient of variation. Similarly, the test set comprises 180 different instances generated with the same parameters, except for the mean values of the demands $\mu_{v,p}$, which were set using a uniform distribution $U[10, 100]$.

7.2. Tuning Parameter K in the Look-Ahead Heuristics

While SCW is a parameterless heuristic, the pilot-based approaches use parameter K to decide upon the number of pilot solutions to explore at each iteration. A common strategy in the pilot-method literature is to set $K = n$, but in our case it translates into one pilot solution per each customer yet to be visited. Thus this strategy leads to unacceptable execution times for the MC-VRPSD. Voß, Fink, and Duin (2005) overcome this difficulty by limiting what they call the *evaluation depth*, that is, to set $K = n$, but to stop the pilot method when the master solution reaches a predefined level of completeness. At this point they continue the construction of the solution using only the subordinate heuristic. Alternatively, we decided not to limit the evaluation depth but to fine tune the value of K looking after a good trade-off between accuracy and speed.

To fine tune the value of K we conducted two experiments on the 180 instances in the tuning set. In the first experiment we conducted Pilot(NN) and Pilot(NI) runs setting $K = 1, 2, \ldots, 10$. The intuition behind the selection of these low values is that the decision rules of NN and NI are good in principle, so if better decisions can be made, they should lie in the vicinity of the first decision. However, to assess the benefits of significant increments in the number of pilot solutions, in the second experiment we tested larger values of K ($K = 15, 20, 25, 30$).

Figures 3(a) and 3(b) summarize the results of the 2,520 (=14 × 180) runs conducted for each heuristic in light of two performance metrics. The first metric is the average gap (over the 180 instances) with respect to the solution provided by the SCW heuristic, given by

$$\Delta_{\text{Pilot(RP)}\,|\,\text{SCW}} = \left(\frac{E[C]_{\text{Pilot(RP)}} - E[C]_{\text{SCW}}}{E[C]_{\text{SCW}}} \right) \times 100\%, \quad (7)$$

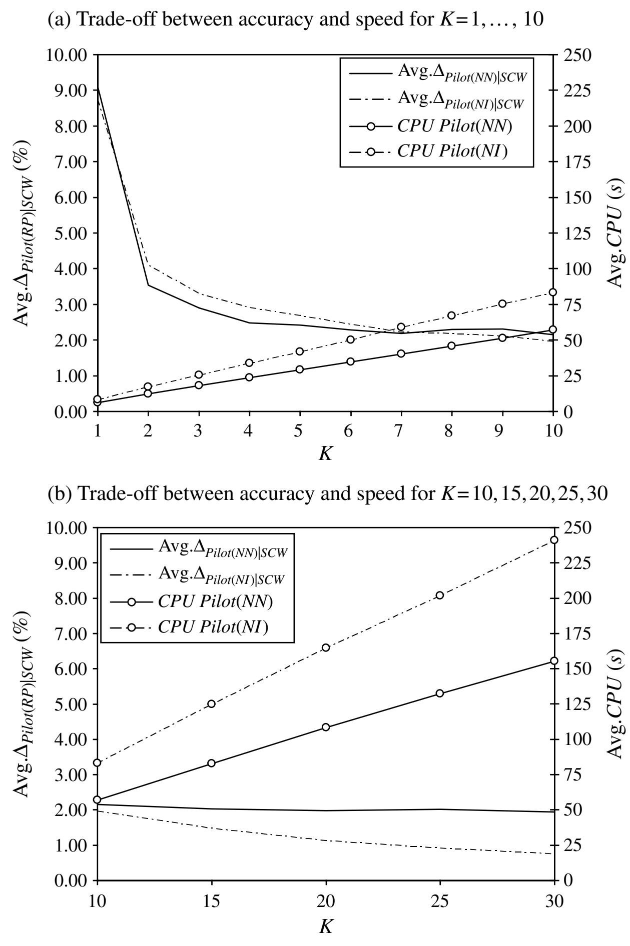


Figure 3 Tuning the Value of K on the Proposed Pilot-Based Heuristics

where $E[C]_{\text{Pilot(RP)}}$ and $E[C]_{\text{SCW}}$ are the total overall cost on a given instance for the corresponding pilot-based method and SCW, respectively. Henceforth, we use Equation (7) to calculate the gap $\Delta_{\text{Alg1}|\text{Alg2}}$ between two algorithms Alg1 and Alg2. The second performance metric is the average execution time on a 3-GHz Intel Xeon X5450 CPU with 8Gb of RAM running Windows Vista Ultimate 64 bits.

The results plotted in Figure 3(a) show the significant contribution of the pilot method to the accuracy of the heuristics. When the look-ahead mechanism is turned off (i.e., $K = 1$), the simple RC procedures perform poorly compared to the SCW heuristic, with average gaps of 9.10% and 8.73% for Pilot(NN) and Pilot(NI), respectively. However, exploring just one extra pilot solution at each iteration—that is, setting $K = 2$—reduces the gap to 3.54% for Pilot(NN) and 4.10% for Pilot(NI) with a minor increment in the average CPU time of 6 and 9 seconds, respectively. As expected, the accuracy of the pilot-based heuristics increases with the value of K, but from $K = 7$ the marginal gap flattens while the execution time continues to grow linearly. For instance, raising K from 1 to 4 reduces the average gap by 6.61%

(=9.10% − 2.49%) for Pilot(NN) and 5.82% (=8.73% − 2.91%) for Pilot(NI), whereas increasing the same three levels from 7 to 10 reduces the gap by only 0.04% and 0.26%, respectively. On the other hand, for each marginal unitary change in K over the range from 1 to 10, the average execution time increases around 5.7 seconds for Pilot(NN) and 8.3 seconds for Pilot(NI). In conclusion, the first experiment suggests that the most interesting trade-offs between accuracy and speed are found with relatively small values of K (i.e., $4 \le K \le 7$). Furthermore, the results of the second experiment support this conclusion. In Figure 3(b) the gap curve for Pilot(NN) is virtually flat for values of K between 10 and 30. On the contrary, the accuracy of Pilot(NI) slightly increases with the value of K over the same interval but at a high computational price: the average gap reduces by 1.21% (=1.97% − 0.76%) at a threefold increase in CPU time. In summary and based on these observations, we set the value of K to 6 for Pilot(NN) and Pilot(NI).

7.3. Results

To assess the performance of the proposed heuristics, we tested SCW, Pilot(NN) and Pilot(NI) on the test set and compared the results with those reported by the MA of Mendoza et al. (2010). For each of the 180 instances we ran the three heuristics with and without S2-Opt and the pilot-based heuristics with $K = 6$ (see §7.2). Table 1 summarizes the results through five performance metrics for each instance group: the number of best known solutions (BKSs) found by each approach; the mean gap with respect to the BKS ($\Delta_{\text{Alg}|\text{BKS}}$); the standard deviation of $\Delta_{\text{Alg}|\text{BKS}}$; the third and fourth quartiles for $\Delta_{\text{Alg}|\text{BKS}}$; and the

Table 1 Performance Metrics for the Test Set

		50 customers		100 customers		200 customers		
Metric	Algorithm	C.V. = 0.1	C.V. = 0.3	C.V. = 0.1	C.V. = 0.3	C.V. = 0.1	C.V. = 0.3	Total/Avg.
Number of BKS	SCW	0	0	0	0	0	0	0
	Pilot(NN)	0	0	0	0	0	0	0
	Pilot(NI)	0	0	0	0	0	0	0
	SCW + S2-Opt	0	1	0	0	0	2	3
	Pilot(NN) + S2-Opt	0	0	1	1	6	2	10
	Pilot(NI) + S2-Opt	1	0	4	3	4	1	13
	MA	29	29	25	26	20	25	154
Avg. $\Delta_{\text{Alg}\,|\,\text{BKS}}$	SCW	7.60	7.66	2.97	3.53	2.54	2.38	4.45
	Pilot(NN)	6.50	5.67	7.33	5.49	7.40	8.08	6.75
	Pilot(NI)	6.39	5.91	7.34	5.97	7.74	8.13	6.91
	SCW + S2-Opt	6.18	5.41	2.29	2.57	1.60	1.48	3.26
	Pilot(NN) + S2-Opt	3.65	3.07	2.88	2.63	1.60	2.58	2.73
	Pilot(NI) + S2-Opt	3.52	3.54	2.99	2.80	1.92	3.22	3.00
	MA	0.05	0.00	0.15	0.11	0.44	0.07	0.14
Std. dev. $\Delta_{\text{Alg}|\text{BKS}}$	SCW	2.94	2.84	1.31	1.48	1.25	1.66	3.04
	Pilot(NN)	2.05	1.97	1.99	1.57	1.46	1.90	2.05
	Pilot(NI)	2.42	1.66	2.10	1.82	1.81	1.44	2.07
	SCW + S2-Opt	2.60	2.63	1.23	1.40	1.20	1.48	2.61
	Pilot(NN) + S2-Opt	2.06	1.41	1.84	1.64	1.35	1.81	1.79
	Pilot(NI) + S2-Opt	2.03	1.43	1.76	1.77	1.83	1.82	1.84
	MA	0.29	0.00	0.45	0.43	0.78	0.18	0.45
$\Delta_{\text{Alg}|\text{BKS}}$ [3th, 4th] quartiles	SCW	[9.34, 15.44]	[9.78, 12.87]	[3.86, 6.43]	[4.55, 7.02]	[3.32, 5.88]	[2.68, 8.14]	—
	Pilot(NN)	[8.15, 10.01]	[6.47, 11.90]	[8.51, 11.38]	[6.28, 9.73]	[8.37, 10.92]	[8.90, 14.01]	—
	Pilot(NI)	[7.81, 11.90]	[6.38, 13.08]	[8.23, 14.00]	[6.94, 10.50]	[8.95, 11.28]	[8.98, 11.52]	—
	SCW + S2-Opt	[8.29, 11.75]	[6.65, 12.19]	[2.95, 4.98]	[3.18, 6.78]	[2.24, 5.44]	[1.70, 6.50]	—
	Pilot(NN) + S2-Opt	[4.77, 8.27]	[3.97, 7.10]	[3.30, 8.49]	[3.46, 6.08]	[2.60, 4.78]	[3.87, 5.69]	—
	Pilot(NI) + S2-Opt	[4.61, 8.09]	[4.23, 6.39]	[4.42, 6.25]	[4.53, 5.51]	[3.21, 6.83]	[4.53, 6.10]	—
	MA	[0.00, 1.57]	[0.00, 0.00]	[0.00, 2.04]	[0.00, 2.28]	[0.85, 2.69]	[0.00, 0.77]	—
Avg. CPU	SCW	<1	<1	8	8	70	77	27
	Pilot(NN)	<1	<1	8	8	96	96	35
	Pilot(NI)	<1	<1	11	11	137	138	50
	SCW + S2-Opt	<1	<1	8	8	73	80	28
	Pilot(NN) + S2-Opt	<1	<1	9	9	103	103	37
	Pilot(NI) + S2-Opt	<1	<1	12	12	144	142	52
	MA*	115	119	503	504	2,554	2,882	1,113

Notes. Number of BKS: number of best known solutions found. Avg. $\Delta_{\text{Alg}|\text{BKS}}$ (%): average gap with respect to the best known solution. Std. Dev. $\Delta_{\text{Alg}|\text{BKS}}$: standard deviation of the gap with respect to the best known solution. $\Delta_{\text{Alg}|\text{BKS}}$ [3th, 4th] quartiles: third and fourth quartiles for the gap with respect to the best known solution. Avg. CPU (s): average execution time.

*On a PC with a 2.4-GHz Core 2 Duo processor, 4 GB of RAM, and running Windows XP Professional.

execution time (in seconds) on a 3 GHz Intel Xeon X5450 CPU with 8 Gb of RAM running Windows Vista Ultimate 64 bits.

The results in Table 1 show that the post-optimization procedure S2-Opt has a significant effect on the performance of the three proposed heuristics. By applying the S2-Opt procedure, the average gap with respect to the BKS was reduced by nearly 25% (from 4.45% to 3.26%) for SCW, and close to 60% for Pilot(NN) (from 6.75% to 2.73%) and Pilot(NI) (from 6.91% to 3.00%). A close look at these results raises an interesting point about the computational cost of achieving these improvements. In the worst case, that is, Pilot(NN) running on the large 200-customer instances, the largest increment in the average execution time due to S2-Opt is just 7 seconds, a marginal computing cost of less than 7.3% $(=((103-96)/96) \times 100\%)$. In conclusion, the S2-Opt procedure achieves its goal of enhancing the accuracy of the heuristics with a low impact on their simplicity and speed. Consequently, in the remainder of the paper we will focus on the results obtained by the proposed algorithms enhanced by S2-Opt, meaning that whenever there is a reference to one of the heuristics, it is implied that S2-Opt was performed, unless otherwise stated.

The results indicate that the proposed heuristics are able to produce competitive solutions for the MC-VRPSD with a reduced computational effort. A look at the number of BKSs achieved by each approach reveals that the proposed heuristics set 26 new best known solutions for the 180-instance test set. Of these 26 new BKSs, 3 (11.5%) were contributed by SCW, 10 (38.5%) by Pilot(NN), and the remaining 13 (50.0%) were found by Pilot(NI). Moreover, the average gap with respect to the BKS in the testbed is only 3.26%, 2.73%, and 3.00% for SCW, Pilot(NN), and Pilot(NI), respectively. These results are encouraging due to the moderate computational effort required by the proposed heuristics. While the MA of Mendoza et al. (2010) invests between 1.5 minutes (50 customers, $CV = 0.1$) and 48 minutes (200 customers, $CV = 0.3$) to generate solutions with average gaps between 0.00% and 0.44% (high quality), the proposed heuristics require only from 1 to 144 seconds to obtain solutions with average gaps ranging from 1.48% to 6.18%. After analyzing the performance of the heuristics by instance size, we found that while in the small 50-customer problems the gap between the proposed approaches and the BKSs is on average 4.23% $(=(6.18\%+5.41\%+3.65\%+3.07\%+3.52\%+3.54\%)/6)$, the average gap in the 200-customer instances reduces to 2.07% $(=(1.60\%+1.48\%+1.60\%+2.58\%+1.92\%+3.22\%)/6)$. This observation tips the balance in favor of the proposed approaches to solve the MC-VRPSD in practice, where both fast and accurate solutions are needed.

After comparing the results from the proposed heuristics we conclude that the three approaches are able to produce solutions of similar quality. A close look at the third and fourth quartiles for the gap reported in Table 1 shows that for only the 50-customer instances there seems to be a difference in the accuracy of SCW, Pilot(NN), and Pilot(NI). In those instances, the pilot-based approaches clearly dominate the SCW heuristic. To illustrate, for the 50-customer instances with $CV = 0.1$, Pilot(NN) and Pilot(NI) achieve average gaps of 3.65% and 3.52%, that are clearly superior to the larger average gap of 6.18% by SCW. In addition, the worst quality gap (fourth quartile) for Pilot(NN) and Pilot(NI)—namely, 8.27% and 8.09%, respectively—falls below the third quartile for SCW (8.29%), providing evidence of the superiority of the pilot-based approaches over SCW. On the other hand, the SCW heuristic proved to be faster than the pilot-based approaches. Although for the 50-customer instances the three algorithms report similar execution times (less than 1 second), the better computing scalability of SCW becomes evident with the larger instance sizes.

7.4. VRPSD Instances

Because the VRPSD is a special case of the MC-VRPSD where $m = 1$, we tested further the effectiveness of the proposed approaches on the set of benchmark problems proposed by Christiansen and Lysgaard (2007). The set, henceforth called VRPSD instances, comprises 40 problems ranging from 16 to 60 customers adapted from the classical instances for the capacitated VRP by Augerat (1995) and Christofides and Eilon (1969). Christiansen and Lysgaard (2007) assumed for each customer v a Poisson distributed demand with mean μ_v, equal to the deterministic demand in the original instance, and kept the remaining data unchanged (i.e., vehicle capacity and vertex locations). Based on the better performance of the pilot-based heuristics over SCW shown in §7.3 on similarly sized problems (50 customers), we decided to focus solely on pilot-based heuristics for the VRPSD.

To solve the VRPSD instances, we slightly modified our pilot-based heuristics. On one hand, in the single-product scenario the exact value of the expected cost of a route $E[C_r]$ is analytically tractable (Laporte, Louveaux, and Van Hamme 2002). Therefore, we modified steps 10–24 in Algorithm 2 to allow the s-split procedure to evaluate the arcs in $\mathcal{A}$ (feasible routes) using the exact value of the expected cost instead of the proxy provided by the TAP approximation. On the other hand, previously published results for the VRPSD instances consider a formulation with an additional constraint that forbids the total expected demand served by a route to exceed the vehicle's capacity. To compare against these results, we took

Table 2 Performance Metrics for the VRPSD Instances with Proven Optimal Solution

	Christiansen and Lysgaard (2007)			Pilot (NN)			Pilot (NI)		
Instance	OPT(=LB)	CPU*	CPU**	$E[C]$	CPU	Δ_{OPT}	$E[C]$	CPU	Δ_{OPT}
P-16-8	**512.82**	0.00	0.00	515.00	0.03	0.43	516.47	0.06	0.71
P-22-8	**681.06**	0.00	0.00	701.14	0.06	2.95	698.69	0.13	2.59
E-22-4	**411.57**	1.00	0.33	419.15	0.16	1.84	428.28	0.20	4.06
P-23-8	**619.52**	1.00	0.33	635.80	0.08	2.63	630.80	0.13	1.82
A-39-5	**869.18**	3.00	0.99	897.98	3.02	3.31	902.78	10.80	3.87
P-21-2	**218.96**	5.00	1.65	240.38	0.45	9.78	228.21	0.34	4.22
A-33-5	**704.20**	8.00	2.64	747.34	0.44	6.13	726.21	0.61	3.13
P-40-5	**472.50**	9.00	2.97	506.55	1.42	7.21	504.89	2.22	6.85
A-33-6	**793.90**	49.00	16.17	846.74	0.36	6.66	801.15	0.58	0.91
E-33-4	**850.27**	86.00	28.38	925.54	1.27	8.85	866.27	10.58	1.88
P-19-2	**224.06**	153.00	50.49	**224.06**	0.23	0.00	**224.06**	0.23	0.00
P-22-2	**231.26**	219.00	72.27	273.38	0.30	18.22	237.12	0.38	2.53
A-39-6	**876.60**	279.00	92.07	897.95	1.06	2.44	889.98	1.50	1.53
A-32-5	**853.60**	282.00	93.06	889.74	0.44	4.23	863.47	0.80	1.16
P-20-2	**233.05**	352.00	116.16	237.06	0.33	1.72	234.26	0.28	0.52
P-51-10	**809.70**	430.00	141.90	871.87	0.81	7.68	831.84	2.14	2.73
P-55-15	**1,068.05**	792.00	261.36	1,106.66	0.75	3.61	1,097.81	1.67	2.79
A-45-7	**1,264.83**	882.00	291.06	1,341.87	10.38	6.09	1,318.44	11.08	4.24
P-60-15	**1,085.49**	1,348.00	444.84	1,111.86	1.30	2.43	1,113.01	2.27	2.54
Avg.		257.84	85.09		1.20	5.06		2.42	2.53
Min.		0.00	0.00		0.03	0.00		0.06	0.00
Max.		1,348.00	444.84		10.38	18.22		11.08	6.85
<3%						8/19			13/19
<5%						11/19			18/19

Notes. OPT(=LB): optimal solution reported by Christiansen and Lysgaard (2007). CPU* (s): execution time on a PC with a 1.5 GHz Pentium Centrino processor and 480 MB of RAM (operational system not reported). CPU** (s): scaled execution time. $E[C]$: total expected cost of the solution. CPU (s): execution time on a PC with a 2.4 GHz Core 2 Duo processor, 4 GB of RAM, and running Windows XP. Δ_{OPT} (%): optimality gap.

advantage of the flexibility of s-split and adapted Pilot(NN) and Pilot(NI) to handle this new side constraint. As in Laporte, Louveaux, and Van Hamme (2002) and Christiansen and Lysgaard (2007) the maximum expected-demand constraint can be modeled as

$$\sum_{v\in r} E[\xi_v] \leq Q \quad \forall r \in \mathcal{R}. \tag{8}$$

Therefore, just by adding the condition $\sum_{v\in r}\mu_v \leq Q$ in steps 9 and 25 of Algorithm 2, s-split and consequently the pilot-based heuristics are easily adapted to handle the additional constraint.

We applied Pilot(NN) and Pilot(NI) with the standard configuration (i.e., $K = 6$) on the VRPSD instances and compared the results with those reported by Christiansen and Lysgaard (2007). Table 2 summarizes the results on a subset of 19 instances (out of the 40) for which Christiansen and Lysgaard (2007) reported optimal solutions. Column 1 presents the instance following the template name S-n-V, where $S \in \{A, P, E\}$ is the source (A and P for Augerat sets A and P, and E for Christofides and Eilon), n is the number of nodes, and V the minimum number of vehicles required; column 2 reports the total expected cost of the optimal solution (OPT); column 3 shows the CPU time in seconds originally reported by the authors; column 4 reports the CPU time scaled to our machine using the benchmarks of Dongarra (2009); and columns 5 to 10 summarize the results for the proposed methods Pilot(NN) and Pilot(NI) in light of the total expected cost of the solution ($E[C]$), the CPU time in seconds, and the gap with respect to the optimal solution (Δ_{OPT}). Similarly, Table 3 presents the results on the 21 instances without proven optimum. For these instances, we assess the quality of our approaches by calculating the gaps with respect to the lower bound (LB) and the best integer solution (BCL) reported by Christiansen and Lysgaard (2007). The latter is given either by (i) the best solution returned by their algorithm under a 1,200-second time limit (396 seconds in scaled CPU) or (ii) the expected cost of the optimal solution for the deterministic instance. To the best of our knowledge, the BCLs are the best known solutions for these 21 VRPSD instances.

Similarly to the results obtained on the test set for the MC-VRPSD (see §7.3), our experiments on the VRPSD show that our pilot-based heuristics are able to generate good quality solutions in short CPU times. On the instances solved to optimality by Christiansen and Lysgaard (2007), Pilot(NN) and Pilot(NI) reported average optimality gaps of 5.06% and 2.53% in short average execution times of 1.20 and 2.42 seconds, respectively. Moreover, the data show that in 58% (11/19) of the instances, Pilot(NN)

Table 3 Performance Metrics for the VRPSD Instances Without Proven Optimal Solution

	Christiansen and Lysgaard (2007)		Pilot(NN)				Pilot(NI)			
Instance	LB	BCL	$E[C]$	CPU	Δ_{LB}	Δ_{BCL}	$E[C]$	CPU	Δ_{LB}	Δ_{BCL}
A-34-5	825.26	827.87	**826.87**	0.52	0.19	−0.12	849.83	0.78	2.98	2.65
A-36-5	852.09	907.55	897.04	0.89	5.28	−1.16	**892.54**	1.11	4.75	−1.65
A-37-5	707.54	**708.34**	753.26	1.38	6.46	6.34	727.48	1.38	2.82	2.70
A-37-6	1,030.44	**1,030.75**	1,051.80	0.80	2.07	2.04	1,048.70	0.98	1.77	1.74
A-38-5	761.12	778.09	**777.82**	10.30	2.19	−0.04	782.87	10.69	2.86	0.61
A-44-6	1,021.29	**1,025.48**	1,065.12	10.41	4.29	3.87	1,065.58	11.05	4.34	3.91
A-45-6	1,006.88	1,096.19	1,050.49	1.88	4.33	−4.17	**1,050.49**	1.66	4.33	−4.17
A-46-7	999.87	**1,002.41**	1,008.75	3.02	0.89	0.63	1,031.57	2.61	3.17	2.91
A-48-7	1,180.22	**1,248.27**	1,279.56	1.42	8.42	2.51	1,256.85	2.55	6.49	0.69
A-53-7	1,109.34	1,180.10	1,190.52	10.69	7.32	0.88	**1,151.23**	11.89	3.78	−2.45
A-54-7	1,279.93	1,342.87	**1,334.63**	4.28	4.27	−0.61	1,338.39	4.80	4.57	−0.33
A-55-9	1,173.56	1,264.18	1,227.57	10.56	4.60	−2.90	**1,202.94**	11.77	2.50	−4.84
A-60-9	1,503.65	1,608.40	**1,547.27**	4.34	2.90	−3.80	1,568.74	12.83	4.33	−2.47
E-51-5	544.63	**553.26**	588.63	3.63	8.08	6.39	594.58	12.16	9.17	7.47
P-45-5	527.77	**546.05**	569.82	2.25	7.97	4.35	562.53	11.34	6.59	3.02
P-50-10	757.44	792.20	783.93	1.03	3.50	−1.04	**783.10**	1.69	3.39	−1.15
P-50-7	581.19	**606.41**	613.14	10.56	5.50	1.11	626.63	2.75	7.82	3.33
P-50-8	666.90	724.69	**690.98**	0.98	3.61	−4.65	702.97	2.19	5.41	−3.00
P-55-10	741.25	797.21	**767.90**	1.77	3.60	−3.68	768.09	11.78	3.62	−3.65
P-55-7	585.47	**616.44**	630.39	5.00	7.67	2.26	620.60	12.33	6.00	0.68
P-60-10	799.58	831.24	**826.42**	3.13	3.36	−0.58	835.97	12.48	4.55	0.57
Avg.				4.23	4.60	0.36		6.71	4.54	0.31
Min.				0.52	0.19	−4.65		0.78	1.77	−4.84
Max.				10.69	8.42	6.39		12.83	9.17	7.47
<3%					5/21	17/21			5/21	17/21
<5%					13/21	19/21			15/21	20/21

Notes. LB: lower bound reported by Christiansen and Lysgaard (2007). BCL: best integer solution reported by Christiansen and Lysgaard (2007). $E[C]$: total expected cost of the solution. CPU (s): execution time on a PC with a 2.4-GHz Core 2 Duo processor, 4 GB of RAM, and running Windows XP. Δ_{LB} (%): Gap with respect to LB. Δ_{BCL} (%): Gap with respect to BCL.

delivers a solution within 5% of the optimal cost. Pilot(NI) reports even better results, achieving gaps lower than 5% on 95% (18/19) of the instances. On the instances without proven optimum, the comparison with respect to the best integer solutions reported by Christiansen and Lysgaard (2007) shows that our look-ahead approaches set 12 new best known solutions (bold numbers in column $E[C]$) and reported small average gaps with respect to the previously best known solutions of 0.36% and 0.31% for Pilot(NN) and Pilot(NI), respectively. In addition, the data confirm that our pilot-based heuristics produce integer solutions for the instances without provably optimal solutions within an average gap of just 4.5% with respect to the lower bounds reported by Christiansen and Lysgaard (2007).

Even though the parameter tuning study (see §7.2) suggested that $K = 6$ offers an interesting trade-off between accuracy and speed for our pilot-based heuristics, it also showed that higher values of K might lead to better solutions if one is willing to pay the extra computational burden. Based on the latter observation, we selected Pilot(NI), our best performing approach, and conducted an additional experiment setting $K = \min\{20, n\}$. The results reported in Table 4 show that under the new configuration for K, Pilot(NI) can further improve its results on the VRPSD instances. For the instances with provably optimal solutions, the average optimality gap reduces by 0.57% (=2.53% − 1.96%) with respect to the results obtained under the default configuration. Even better improvements are found on the instances without provably optimal solutions, where the average gap with respect to the lower bounds reduces by 1% (=4.54% − 3.53%). In addition, for the instances without provably optimal solutions, Pilot(NI) with $K = \min\{20, n\}$ improves 11 of the solutions reported by Christiansen and Lysgaard (2007)—negative values in column Δ_{BCL}—and unveils 6 new best known solutions for the subset.

In summary, the results obtained both on the MC-VRPSD and VRPSD instances are remarkable taking into account the simplicity of the pilot-based heuristics, the good quality of the solutions, and the low computational requirements.

8. Concluding Remarks and Future Research

This paper proposes simple and effective constructive heuristics for the MC-VRPSD. The problem consists

Table 4 **Performance of Pilot(NI) with $K = \min\{20, n\}$ on the VRPSD Instances**

Instances with proven optimal solution				Instances without proven optimal solution				
Instance	$E[C]$	CPU	Δ_{OPT}	Instance	$E[C]$	CPU	Δ_{LB}	Δ_{BCL}
P-16-8	515.00	0.09	0.43	A-34-5	**826.87**	1.53	0.19	−0.12
P-22-8	700.08	0.23	2.79	A-36-5	**890.13**	2.27	4.46	−1.92
E-22-4	419.15	0.39	1.84	A-37-5	724.62	2.75	2.41	2.30
P-23-8	628.66	0.28	1.47	A-37-6	1,056.76	1.83	2.55	2.52
A-39-5	895.24	2.78	3.00	A-38-5	782.17	17.75	2.77	0.52
P-21-2	228.21	0.63	4.22	A-44-6	1,054.32	18.7	3.23	2.81
A-33-5	716.71	1.34	1.78	A-45-6	1,050.49	3.25	4.33	−4.17
P-40-5	501.78	3.41	6.20	A-46-7	1,006.49	3.77	0.66	0.41
A-33-6	801.15	1.30	0.91	A-48-7	1,246.76	3.98	5.64	−0.12
E-33-4	862.71	1.92	1.46	A-53-7	1,159.61	5.8	4.53	−1.74
P-19-2	**224.06**	0.44	0.00	A-54-7	**1,311.12**	7.66	2.44	−2.36
P-22-2	237.12	0.70	2.53	A-55-9	**1,198.96**	20.55	2.16	−5.16
A-39-6	879.52	2.41	0.33	A-60-9	1,555.63	23.23	3.46	−3.28
A-32-5	859.62	1.45	0.71	E-51-5	564.15	21.77	3.58	1.97
P-20-2	234.26	0.52	0.52	P-45-5	553.04	19.55	4.79	1.28
P-51-10	817.90	4.27	1.01	P-50-10	**781.05**	3.64	3.12	−1.41
P-55-15	1,076.31	3.99	0.77	P-50-7	608.11	7.39	4.63	0.28
A-45-7	1,327.79	18.84	4.98	P-50-8	702.97	4.34	5.41	−3.00
P-60-15	1,110.26	5.53	2.28	P-55-10	**759.35**	5.16	2.44	−4.75
				P-55-7	622.97	22.06	6.40	1.06
				P-60-10	839.57	22.38	5.00	1.00
Avg.		2.66	1.96			10.45	3.53	−0.66
Min.		0.09	0.00			1.53	0.19	−5.16
Max.		18.84	6.20			23.23	6.40	2.81
<3%			15/19				8/21	21/21
<5%			18/19				17/21	21/21

Notes. $E[C]$: total expected cost of the solution. CPU (s): execution time on a PC with a 2.4 GHz Core 2 Duo processor, 4 GB of RAM, and running Windows XP. Δ_{OPT} (%): Gap with respect to the optimal solution reported by Christiansen and Lysgaard (2007). Δ_{LB} (%): Gap with respect to the lower bound reported by Christiansen and Lysgaard (2007). Δ_{BCL} (%): Gap with respect to the best integer solution reported by Christiansen and Lysgaard (2007).

of designing collection routes to serve a set of customers who have independent stochastic demands for different products that must be transported in independent compartments due to product incompatibility constraints. Despite its practical applicability, the MC-VRPSD has not received the deserved attention in the literature.

We proposed a set of three constructive heuristics for the MC-VRPSD. The stochastic Clarke and Wright heuristic (SCW) extends the savings-based approach of Dror and Trudeau (1986) for the VRPSD to (i) work on the multicompartment scenario and (ii) handle a different recourse action, that is, a round trip to the depot to restore the capacity of the vehicle compartments whenever it is exceeded. The other two heuristics are based on the principle of coupling route-first, cluster-second (RC) procedures with a look-ahead mechanism known as the pilot method (Voß, Fink, and Duin 2005). To the best of our knowledge, this is the first proposal of a pilot-based method in the field of vehicle routing. The RC procedures use for the routing phase two fast and easy-to-implement construction algorithms for the deterministic TSP, namely, the nearest neighbor (NN) and the nearest insertion (NI) heuristics. The clustering phase uses a dynamic programming algorithm (s-split) specially tailored to handle the stochastic nature of demands in the complex multicompartment scenario. The pilot method iterates with the RC procedures to further explore the evaluation of each move in the NN and NI heuristics by (i) considering K alternatives instead of only one, and (ii) measuring the impact of each of the K choices in the final output of the solution rather than in the partial tour. The two resulting heuristics, Pilot(NN) and Pilot(NI), along with the SCW heuristic were complemented with S2-Opt, a new and effective post-optimization procedure for the MC-VRPSD.

The three proposed heuristics were tested on a large set of instances for the MC-VRPSD and their results compared with those obtained by the memetic algorithm (MA) of Mendoza et al. (2010). The experiments indicate that the heuristics are competitive, in terms of both accuracy and speed. The proposed heuristics followed by S2-Opt were able to produce in short execution times ranging from 1 to 144 seconds (on average), solutions with an average gap (with respect to the BKSs) of 3.26%, 2.73%, and 3.00% for SCW, Pilot(NN) and Pilot(NI), respectively. Moreover, the three proposed algorithms unveiled new best known solutions for 26 out of the 180 instances in the test set.

In addition to the experiments conducted on the MC-VRPSD, we tested the pilot-based heuristics on a 40-instance testbed for the VRPSD, a particular case of the MC-VRPSD in which each customer has a random demand for a single product. The solutions found by the two heuristics were compared with those reported by Christiansen and Lysgaard (2007). The results showed that Pilot(NN) and Pilot(NI) set 12 new best known solutions for the testbed and reported gaps within 5% with respect to lower bounds reported in (Christiansen and Lysgaard 2007) on 24 (60%) and 38 (95%) instances, respectively. In conclusion, thanks to their accuracy, speed, and simplicity, the proposed approaches can be used as stand-alone solution methods or to induce a warm startup for metaheuristics and exact methods.

Research currently underway includes (i) the extension of the look-ahead heuristics to solve other VRP variants, (ii) development of new pilot-based methods to work on top of other constructive heuristics for VRPs, and (iii) the implementation of a version of s-split based on simulation that allows the constructive heuristics to handle the MC-VRPSD with correlated demands at the customer site. Some problems that remain unsolved but pose interesting challenges are the derivation of a lower bound for the MC-VRPSD, an extension to the scenario where preventive trips to the depot (restocking) are allowed, the inclusion of a constraint over the maximum number of routes (limited fleet), the extension to handle different recourse actions such as cooperation between routes upon route failures, and the study of the MC-VRPSD under the scenario in which each product can be delivered to a customer by a different route.

Acknowledgments
The authors thank the referees and editors for their insightful comments and suggestions. This research was partially funded by Grant 471-2007 from COLCIENCIAS/SENA (Colombia).

Appendix. Calculation of the Failure Probability in the Multicompartment Scenario

Note that in the formulation given by (2) and (3) in §2, no constraints restrict the number of failures in route r. However, because demand realizations $\vec{\xi}$ are nonnegative and less than the capacity of the corresponding compartment, if a route failure occurs at customer v_i it is in the worst case the $(i-1)$th failure on route r. For notational simplicity, let us define $z(v_i)$ as a function that takes the value of 1 if a route failure occurs in route r during the service of customer v_i and the value of 0 otherwise. Because all compartments are emptied after each visit to the depot, traveling a route is a process that is renewed after each failure. Therefore, the value of $\Pr(z(v_i)=1)$ can be written as

$$\Pr(z(v_i)=1) = \Pr(z(v_i)=1,\, z(v_k)=0,\, 0<k<i) + \sum_{j=1}^{i-1}\Pr(z(v_i)=1,\, z(v_j)=1,\, z(v_k)=0,\, j<k<i), \quad (9)$$

where the first term represents the probability that the first route failure occurs at customer v_i and the second term the probability that at least one failure occurs between the start of the route at depot and customer v_i. Because the first renewal process occurs at the depot before visiting the first customer, $\Pr(z(0)=1)=1$, and (9) can be rewritten as

$$\Pr(z(v_i)=1) = \sum_{j=0}^{i-1}\Pr(z(v_i)=1,\, z(v_j)=1,\, z(v_k)=0,\, j<k<i). \quad (10)$$

Considering that the last failure occurs at customer v_j, an expression to calculate each term in (10) can be derived as follows. Let $A(p)$ be the event that capacity Q_p for product p is exceeded while servicing customer v_i, and B the event that no route failures occurred between vertices v_j and v_i given that the last failure occurred at customer v_j (i.e., $z(v_j)=1, z(v_k)=0,\ \forall\, j<k<i$). The probabilities of events $A(p)$ and B are given by Equations (11) and (12), respectively.

$$\Pr(A(p)) = \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i}\xi_{v_k,p} > Q_p\right), \quad (11)$$

$$\Pr(B) = \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1}\xi_{v_k,p} \le Q_p,\ \forall\, p\in\mathcal{P}\right), \quad (12)$$

where $\xi'_{v_j,p}$ is the unattended demand of customer v_j for product p after the last route failure. Figure A.1 describes the probability tree at arrival to customer v_i.
From the probability tree it can be seen that

$$\begin{aligned}\Pr(z(v_i)=0 \mid z(v_j)=1,\ z(v_k)=0,\ j<k<i) &= (1-\Pr(A(1)\mid B))\times(1-\Pr(A(2)\mid B))\\ &\quad\times\cdots\times(1-\Pr(A(m)\mid B)),\end{aligned} \quad (13)$$

and

$$\begin{aligned}\Pr(z(v_i)=1 \mid z(v_j)=1,\ z(v_k)=0,\ j<k<i) &= 1-\Pr(z(v_i)=0\mid z(v_j)=1,\ z(v_k)=0,\ j<k<i)\\ &= 1-[(1-\Pr(A(1)\mid B))\times(1-\Pr(A(2)\mid B))\\ &\quad\times\cdots\times(1-\Pr(A(m)\mid B))].\end{aligned} \quad (14)$$

From Bayes' theorem, for product 1 we have

$$\begin{aligned}\Pr(A(1)\mid B) &= \frac{\Pr(A(1),B)}{\Pr(B)}\\ &= \left(\Pr\left(\xi'_{v_j,1}+\sum_{k=j+1}^{i-1}\xi_{v_k,1}\le Q_1 < \xi'_{v_j,1}+\sum_{k=j+1}^{i}\xi_{v_k,1},\ \xi'_{v_j,p}\right.\right.\\ &\qquad\left.\left. +\sum_{k=j+1}^{i-1}\xi_{v_k,p}\le Q_p,\ p\in\mathcal{P}\backslash\{1\}\right)\right)\\ &\quad\cdot\left(\Pr\left(\xi'_{v_j,p}+\sum_{k=j+1}^{i-1}\xi_{v_k,p}\le Q_p,\ p\in\mathcal{P}\right)\right)^{-1}\end{aligned}$$

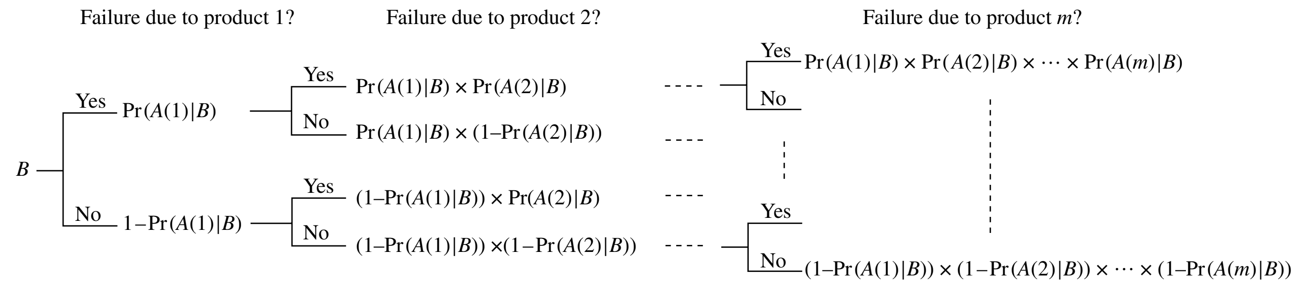

Failure due to product 1?
Failure due to product 2?
Failure due to product m?
B
Yes
$\Pr(A(1)|B)$
No
$1-\Pr(A(1)|B)$
Yes
$\Pr(A(1)|B)\times\Pr(A(2)|B)$
No
$\Pr(A(1)|B)\times(1-\Pr(A(2)|B))$
Yes
$(1-\Pr(A(1)|B))\times\Pr(A(2)|B)$
No
$(1-\Pr(A(1)|B))\times(1-\Pr(A(2)|B))$
Yes
$\Pr(A(1)|B)\times\Pr(A(2)|B)\times\cdots\times\Pr(A(m)|B)$
No
Yes
No
$(1-\Pr(A(1)|B))\times(1-\Pr(A(2)|B))\times\cdots\times(1-\Pr(A(m)|B))$

Figure A.1 Probability Tree of Events at Arrival to Customer v_i Given That the Last Failure Occurred at Customer v_j

$$= \left(\Pr\left(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1 < \xi'_{v_j,1} + \sum_{k=j+1}^{i} \xi_{v_k,1} \right) \times \Pr\left(\xi'_{v_j,2} + \sum_{k=j+1}^{i-1} \xi_{v_k,2} \le Q_2 \right) \times \cdots \times \Pr\left(\xi'_{v_j,m} + \sum_{k=j+1}^{i-1} \xi_{v_k,m} \le Q_m \right) \right)$$
$$\cdot \left(\Pr\left(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1 \right) \times \Pr\left(\xi'_{v_j,2} + \sum_{k=j+1}^{i-1} \xi_{v_k,2} \le Q_2 \right) \times \cdots \times \Pr\left(\xi'_{v_j,m} + \sum_{k=j+1}^{i-1} \xi_{v_k,m} \le Q_m \right) \right)^{-1}$$
$$= \frac{\Pr(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1 < \xi'_{v_j,1} + \sum_{k=j+1}^{i} \xi_{v_k,1})}{\Pr(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1)}, \qquad (15)$$

where $\Pr(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1)$ is the cumulative probability that the total demand for product 1, collected between customers v_j and v_{i-1} after failing at vertex v_j, does not exceed the capacity of the compartment Q_1. The numerator in (15) can be rewritten as

$$\Pr\left(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1 < \xi'_{v_j,1} + \sum_{k=j+1}^{i} \xi_{v_k,1} \right) = \Pr\left(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1 \right) - \Pr\left(\xi'_{v_j,1} + \sum_{k=j+1}^{i} \xi_{v_k,1} \le Q_1 \right). \qquad (16)$$

Re-writing expression (15) in terms of (16) and simplifying, we obtain

$$1 - \Pr(A(1) \mid B) = \frac{\Pr(\xi'_{v_j,1} + \sum_{k=j+1}^{i} \xi_{v_k,1} \le Q_1)}{\Pr(\xi'_{v_j,1} + \sum_{k=j+1}^{i-1} \xi_{v_k,1} \le Q_1)}. \qquad (17)$$

The result on (17) extends to any $p \in \mathcal{P}$. Hence, (13) can be rewritten as

$$\Pr(z(v_i) = 0 \mid z(v_j) = 1,\ z(v_k) = 0,\ j < k < i) = \prod_{p=1}^{m} \frac{\Pr(\xi'_{v_j,p} + \sum_{k=j+1}^{i} \xi_{v_k,p} \le Q_p)}{\Pr(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p)}, \qquad (18)$$

and (14) as

$$\Pr(z(v_i) = 1 \mid z(v_j) = 1,\ z(v_k) = 0,\ j < k < i) = 1 - \prod_{p=1}^{m} \frac{\Pr(\xi'_{v_j,p} + \sum_{k=j+1}^{i} \xi_{v_k,p} \le Q_p)}{\Pr(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p)}. \qquad (19)$$

From Bayes' theorem,

$$\Pr(z(v_i) = 1 \mid z(v_j) = 1,\ z(v_k) = 0,\ j < k < i) = \frac{\Pr(z(v_i) = 1,\ z(v_j) = 1,\ z(v_k) = 0,\ j < k < i)}{\Pr(z(v_j) = 1,\ z(v_k) = 0,\ j < k < i)}, \qquad (20)$$

where

$$\Pr(z(v_j) = 1,\ z(v_k) = 0,\ j < k < i) = \prod_{p=1}^{m} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p \right) \times \Pr(z(v_j) = 1), \qquad (21)$$

replacing (19) and (21) in (20) and simplifying we obtain

$$\Pr(z(v_i) = 1,\ z(v_j) = 1,\ z(v_k) = 0,\ j < k < i) = \left[\prod_{p=1}^{m} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p \right) - \prod_{p=1}^{m} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i} \xi_{v_k,p} \le Q_p \right) \right] \times \Pr(z(v_j) = 1), \qquad (22)$$

finally, replacing (22) in (10) we have that the probability of having a failure in route r while servicing customer $v_i \in r$ is equal to

$$\Pr(z(v_i) = 1) = \sum_{j=0}^{i-1} \left[\prod_{p=1}^{m} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i-1} \xi_{v_k,p} \le Q_p \right) - \prod_{p=1}^{m} \Pr\left(\xi'_{v_j,p} + \sum_{k=j+1}^{i} \xi_{v_k,p} \le Q_p \right) \right] \times \Pr(z(v_j) = 1). \quad \square \qquad (23)$$

References

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

Augerat, P. 1995. Approche polyèdrale du problème de tournées de véhicules. Ph.D. thesis, Institut National Polytechnique de Grenoble-INPG.

Bentley, J. B. 1992. Fast algorithms for geometric traveling salesman problem. *INFORMS J. Comput.* **4**(4) 387–411.

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

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

Caramia, M., F. Guerriero. 2010. A milk collection problem with incompatibility constraints. *Interfaces* **40**(2) 130–143.

Chajakis, E. D., M. Guignard. 2003. Scheduling deliveries in vehicles with multiple compartments. *J. Global Optim.* **26**(1) 43–78.

Christiansen, C., J. Lysgaard. 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Oper. Res. Lett.* **35**(6) 773–781.

Christofides, N., S. Eilon. 1969. An algorithm for the vehicle dispatching problem. *Oper. Res.* **20**(3) 309–318.

Cordeau, J. F., G. Laporte, M. Savelsbergh, D. Vigo. 2006. Vehicle routing. C. Barnhart, G. Laporte, eds. *Handbooks in Operations Research and Management Science: Transportation*, Vol. 14. Elsevier, Amsterdam, 367–428.

Cordeau, J. F., M. Gendreau, G. Laporte, J. Y. Potvin, F. Semet. 2002. A guide to vehicle routing heuristics. *J. Oper. Res. Soc.* **53**(5) 512–522.

Derigs, U., J. Gottlieb, J. Kalkoff, M. Piesche, F. Rothlauf, U. Vogel. 2010. Vehicle routing with compartments: Applications, modelling and heuristics. *OR Spectrum.* ePub ahead of print, http://dx.doi.org/10.1007/s00291-010-0194-3.

Dongarra, J. J. 2009. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville.

Dror, M., P. Trudeau. 1986. Stochastic vehicle routing with modified savings algorithm. *Eur. J. Oper. Res.* **23**(2) 228–235.

Duin, C., S. Voß. 1999. The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks* **34**(3) 181–191.

El Fallahi, A., C. Prins, R. Wolfler-Calvo. 2008. A memetic algorithm and a tabu search for the multicompartment vehicle routing problem. *Comput. Oper. Res.* **35**(5) 1725–1741.

Fink, A., S. Voß. 2003. Solving the continuous flow-shop scheduling problem by metaheuristics. *Eur. J. Oper. Res.* **151**(2) 400–414.

Fink, A., G. Schneidereit, S. Voß. 2000. Solving general ring network design problems by meta-heuristics. M. Laguna, J. L. González-Velarde, eds. *Computing Tools for Modeling, Optimization, and Simulation*. Kluwer Academic Publishers, Boston, 91–113.

Fischetti, M., A. Lodi. 2003. Local branching. *Math. Programming* **98**(1–3) 23–47.

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

Gendreau, M., G. Laporte, R. Séguin. 1996a. Stochastic vehicle routing. *Eur. J. Oper. Res.* **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. *Oper. Res.* **44**(3) 469–477.

Golden, B., S. Raghavan, E. Wasil, eds. 2008. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, New York.

Hansen, P., N. Mladenović. 2006. First vs. best improvement: An empirical study. *Discrete Appl. Math.* **154**(5) 802–817.

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

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

Law, A. M., W. D. Kelton. 2000. *Simulation Modeling and Analysis*. McGraw-Hill, Singapore.

L'Ecuyer, P., L. Meliani, J. Vaucher. 2002. SSJ: A framework for stochastic simulation in Java. E. Yücesan, C. H. Chen, J. L. Snowdon, J. M. Charnes, eds. *Proc. 2002 Winter Simulation Conf.*, IEEE Press, Piscataway, 234–242.

Liu, C.-S., M.-Y. Lai. 2009. The vehicle routing problem with uncertain demand at nodes. *Transportation Res. Part E: Logist. and Transportation Rev.* **45**(4) 517–524.

Mendoza, J. E., B. Castanier, C. Guéret, A. L. Medaglia, N. Velasco. 2008. Approximating the expected cost of recourse on a multi-compartment vehicle routing problem with stochastic demands. Technical Report 08/03/AUTO, Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN). École des Mines de Nantes, France.

Mendoza, J. E., B. Castanier, C. Guéret, A. L. Medaglia, N. Velasco. 2009. A simulation-based MOEA for the multi-compartment vehicle routing problem with stochastic demands. *Proc. VIII Metaheuristics Internat. Conf. (MIC), Hamburg, Germany*.

Mendoza, J. E., B. Castanier, C. Guéret, A. L. Medaglia, N. Velasco. 2010. A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Comput. Oper. Res.* **37**(11) 1886–1898.

Muyldermans, L., G. Pang. 2010a. A guided local search procedure for the multi-compartment capacitated arc routing problem. *Comput. Oper. Res.* **37**(9) 1662–1673.

Muyldermans, L., G. Pang. 2010b. On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *Eur. J. Oper. Res.* **206**(1) 93–103.

Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* **31**(12) 1985–2002.

Prins, C., N. Labadi, M. Reghioui. 2009. Tour splitting algorithms for vehicle routing problems. *Internat. J. Production Res.* **47**(2) 507–535.

Rei, W., M. Gendreau, P. Soriano. 2007. Local branching cuts for the 0-1 integer L-shaped algorithm. Technical Report CIRRELT-2007-23, Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Montréal.

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

Ruiz, R., C. Maroto, J. Alcaraz. 2004. A decision support system for a real vehicle routing problem. *Eur. J. Oper. Res.* **153**(3) 593–606.

Savelsbergh, M. W. P., M. Goetschalckx. 1995. A comparison of the efficiency of fixed versus variable vehicle routes. *J. Bus. Logist.* **16** 163–187.

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

Sörensen, K., M. Sevaux. 2009. A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling. *J. Math. Modeling Algorithms* **8**(4) 387–407.

Tan, K. C., C. Y. Cheonga, C. K. Goh. 2007. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. *Eur. J. Oper. Res.* **177**(2) 813–839.

Tatarakis, A., I. Minis. 2009. Stochastic single vehicle routing with a predefined customer sequence and multiple depot returns. *Eur. J. Oper. Res.* **197**(2) 557–571.

Toth, P., D. Vigo, eds. 2002. The vehicle routing problem. *Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia.

Van der Bruggen, R., L. Gruson, M. Salomon. 1995. Reconsidering the distribution structure of gasoline products for a large oil company. *Eur. J. Oper. Res.* **81**(3) 460–473.

Villegas, J., A. L. Medaglia, J. E. Mendoza, C. Prins, C. Prodhon, N. Velasco. 2008. A split-based framework for the vehicle routing problem. *XIV Latin Ibero-American Congress on Operations Research (CLAIO)*. Cartagena, Colombia.

Voß, S., A. Fink, C. Duin. 2005. Looking ahead with the pilot method. *Ann. Oper. Res.* **136**(1) 285–302.

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