Comput Optim Appl (2015) 61:463–487
DOI 10.1007/s10589-014-9719-z

Estimation-based metaheuristics for the single vehicle routing problem with stochastic demands and customers

Prasanna Balaprakash · Mauro Birattari · Thomas Stützle · Marco Dorigo

Received: 26 June 2013 / Published online: 13 December 2014

Abstract The vehicle routing problem with stochastic demands and customers (VRPSDC) requires finding the optimal route for a capacitated vehicle that delivers goods to a set of customers, where each customer has a fixed probability of requiring being visited and a stochastic demand. For large instances, the evaluation of the cost function is a primary bottleneck when searching for high quality solutions within a limited computation time. We tackle this issue by using an empirical estimation approach. Moreover, we adopt a recently developed state-of-the-art iterative improvement algorithm for the closely related probabilistic traveling salesman problem. We integrate these two components into several metaheuristics and we show that they outperform substantially the current best algorithm for this problem.

Keywords Metaheuristics · Empirical estimation · Vehicle routing with stochastic demands and customers

P. Balaprakash (✉)
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA
e-mail: pbalapra@mcs.anl.gov

M. Birattari · T. Stützle · M. Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
e-mail: mbiro@ulb.ac.be

T. Stützle
e-mail: stuetzle@ulb.ac.be

M. Dorigo
e-mail: mdorigo@ulb.ac.be

1 Introduction

Stochastic vehicle routing problems (SVRPs) have enormous practical importance in transportation, strategic planning and scheduling [1]. They are similar to deterministic vehicle routing problems except that customers visits, their demands, or travel times are described by probability distributions. The introduction of probabilistic elements into a vehicle routing problem increases its difficulty and solving it becomes a much more challenging task than for the deterministic counterpart. Classical examples of SVRPs are the probabilistic traveling salesman problem [2,3], the vehicle routing problem with stochastic customers [1,4,5], the vehicle routing problem with stochastic demands [6], the vehicle routing problem with stochastic demands and customers [1,7], and the vehicle routing problem with stochastic travel and service times [8]. These problems are academic variants of various real world SVRPs [1,2,9,10]. Exact methods [11–15] can solve only small sized SVRPs [10]. Pragmatic approaches to tackle SVRPs mainly involve the application of stochastic local search methods [16], in particular iterative improvement algorithms and metaheuristics [17–23].

The vehicle routing problem with stochastic demands and customers (VRPSDC) [1] is concerned with minimizing the cost involved in routing a capacitated vehicle that distributes goods to a number of customers. Each customer has a probability of requiring being visited and a stochastic demand. The VRPSDC is an $\mathcal{NP}$-hard problem that models a number of practical problems in the areas of truckload operations and package delivery systems [1,7]. This problem is one of the most difficult, classical stochastic vehicle routing problems [10]. It combines two stochastic vehicle routing problems: the probabilistic traveling salesman problem (PTSP) and the vehicle routing problem with stochastic demands. The former is similar to the traveling salesman problem except that each customer has a probability of requiring being visited; in the latter, a capacitated vehicle has to serve all customers, where each customer has a stochastic demand.

An effective strategy to tackle SVRPs such as the VRPSDC is *a priori* optimization [1,10,24]. In this strategy, the goal is to find an *a priori* solution that minimizes the expected cost of the associated *a posteriori* solution. The *a priori* solution is a sequence of all customers, which is decided before knowing which customers need to be served and what is their demand; the associated *a posteriori* solution, which is computed after the realization of the stochastic elements is known, is obtained by following the *a priori* solution but with some recourse actions such as, for example, skipping customers that do not require being visited and returning to the depot for replenishment. An alternative to *a priori* optimization is re-optimization [21]. For algorithms that solve other SVRPs using re-optimization, we refer the reader to [25–29].

A priori optimization can be addressed by two classes of approaches: *analytical computation*, which computes the cost of a solution using closed-form expressions; and *empirical estimation*, which estimates the solution cost through Monte Carlo simulation. So far, the VRPSDC has been tackled by algorithms that adopt the analytical computation approach. Unfortunately, this approach is computationally very expensive. Consequently, for large instances with several hundreds of nodes, the adoption of the analytical computation approach considerably affects the performance of the algorithms. This issue can be addressed by an empirical estimation approach, which has not yet been investigated for the VRPSDC.

In our recent research [30–33], we developed `2.5-opt-EEais`, an effective estimation-based iterative improvement algorithm for the PTSP. Then, we used this algorithm as a local search inside three metaheuristics: iterated local search (ILS) [34], memetic algorithms (MAs) [35,36], and ant colony optimization (ACO) [37,38]. All these metaheuristics use an estimation-based procedure to evaluate the solution cost. Our `2.5-opt-EEais` algorithm and estimation-based metaheuristics outperform by quite a wide margin previously proposed algorithms for the PTSP. In this paper, we extend the aforementioned metaheuristics to tackle the VRPSDC by customizing the estimation-based procedure to evaluate the solution cost of the VRPSDC. We directly use `2.5-opt-EEais` as a local search algorithm in all metaheuristics. Thus, during the iterative improvement phase, customers demand and vehicle capacity are completely ignored. This particular usage is aimed to exploit the speed advantage of `2.5-opt-EEais`.

In this paper, we carry out three sets of experiments. In the first set, we show that a random restart local search that adopts `2.5-opt-EEais` as the local search outperforms the existing tailor-made tabu search algorithm. In the second set, we compare our proposed estimation procedure with the currently used analytical computation procedure in a random restart local search and we show that the former is highly effective as it substantially reduces the computation time needed to compare the cost of the solutions. Finally, we study estimation-based ILS, MAs, and ACO. All three algorithms obtain high quality solutions that are better than that of random restart local search. Besides the adoption of `2.5-opt-EEais` as local search, the high performance of the three algorithms is to be ascribed to the rigorous parameter tuning, which is performed by grouping the instances into a number of classes and by tuning each algorithm on each instance class.

The paper is organized as follows: In Sect. 2, we describe the VRPSDC and its solution approaches. In Sect. 3, we describe some estimation-based metaheuristics for the VRPSDC. In Sect. 4, we present an experimental study of the proposed algorithms and we show their effectiveness. In Sect. 5, we conclude the paper.

2 The VRPSDC

Formally, an instance of the VRPSDC can be defined on a graph G with the following elements: a set $V = \{1, 2, ..., n\}$ of nodes that represent customers, with node 1 being the depot; a set $A = \{\langle i, j\rangle : i, j \in V, i \neq j\}$ of edges, where edge $\langle i, j\rangle$ connects nodes i and j; a set $C = \{c_{ij} : \langle i, j\rangle \in A\}$ of travel costs, where c_{ij} is the cost of using edge $\langle i, j\rangle \in A$; a set $P = \{p_i : i \in V, i \neq 1, 0 \leq p_i \leq 1\}$ of probabilities, where p_i is the probability that a node i requires being visited; a set $\psi = \{\psi_i : i \in V, i \neq 1\}$ of random variables, where ψ_i describes the stochastic demand of node i and a vehicle of capacity Q. It is assumed that (i) travel costs are symmetric, that is, for all pairs of nodes i, j we have $c_{ij} = c_{ji}$; (ii) the events that two distinct nodes i and j require being visited and their demands are assumed to be independent; (iii) demand ψ_i of a node i is a discrete random variable that can take a finite set of distinct demand levels

$\{\xi_i^1, \ldots, \xi_i^{L_i}\}$ and the probability that $\psi_i = \xi_i^l$ is given by p_i^l; (iv) the information that a node i does not require being visited is revealed before the vehicle leaves the predecessor of node i; (v) the exact demand of a node i is revealed only when the vehicle reaches node i; and (vi) the maximum demand of any node does not exceed the vehicle capacity Q.

The *a priori* optimization for the VRPSDC consists in finding an *a priori* solution that minimizes the expected cost of the *a posteriori* solution. The *a priori* solution is a Hamiltonian tour, which is found before knowing which nodes require being visited and their respective demands; the *a posteriori* solution is obtained by following the nodes in the order of the *a priori* solution with the following recourse actions: nodes that do not require being visited are skipped; if the vehicle is empty after serving a node, it goes back to the depot for replenishment and resumes the delivery from the next node that requires being visited; if the amount of goods available in the vehicle is not enough to satisfy the demand of a node, all the goods in the vehicle are delivered to the node, then the vehicle returns to the depot for replenishment, and it goes back to the same node to deliver the remaining goods.

Bertsimas [1] and Gendreau et al. [12,39] derived closed-form expressions to compute the exact cost of a VRPSDC solution. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n), \pi(n+1) = \pi(1))$ be an *a priori* solution for the VRPSDC and it is a permutation of the set V. The cost $F(\pi)$ of π can be obtained as follows:

$$F(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n+1} c_{\pi(i)\pi(j)} \overline{p}_{\pi(i)\pi(j)} + \gamma_2(Q), \tag{1}$$

where

$$\overline{p}_{\pi(i)\pi(j)} = \begin{cases} p_{\pi(i)} p_{\pi(j)}, \\ \qquad j = i+1, \\ p_{\pi(i)} p_{\pi(j)} \prod_{h=i+1}^{j-1} (1 - p_{\pi(h)}), \\ \qquad j > i+1, \end{cases} \tag{2}$$

$$\gamma_i(g) = \begin{cases} p_{\pi(n)} (c_{\pi(n)\pi(1)} + c_{\pi(1)\pi(n)}) \sum_{l|\xi^l_{\pi(n)} > g} p^l_{\pi(n)}, \\ \qquad (i = n,\ 1 \le g \le Q), \\ (1 - p_{\pi(i)}) \gamma_{i+1}(g) \\ + p_{\pi(i)} \Big[\sum_{l|\xi_{\pi(i)} > g} p^l_{\pi(i)} (\gamma_{i+1}(Q - \xi^l_{\pi(i)}) + c_{\pi(i)\pi(1)} + c_{\pi(1)\pi(i)}) \\ \qquad + \sum_{l|\xi_{\pi(i)} < g} p^l_{\pi(i)} \gamma_{i+1}(g - \xi^l_{\pi(i)}) \Big] \\ + P(\xi_{\pi(i)} = g | \pi(i) \text{ is present}) \Big[p_{\pi(i)} \gamma_{i+1}(Q) \\ \qquad + \sum_{j=i+1}^{n} \overline{p}_{\pi(i)\pi(j)} (c_{\pi(i)\pi(1)} \\ \qquad + c_{\pi(1)\pi(j)} - c_{\pi(i)\pi(j)}) \Big] \\ \qquad (i = 2, \ldots, n-1;\ 1 \le g \le Q). \end{cases} \tag{3}$$

In Eq. 1, the first term represents the expected cost of π under node skipping recourse actions; the second term $\gamma_i(g)$ represents the expected non-negative penalty cost from the node $\pi(i)$ to the depot, given that the residual capacity of the vehicle is equal to g upon arrival at $\pi(i)$. Since the first node in π is always the depot, the second term computation starts with index 2 and the vehicle capacity Q. The time complexity of evaluating Eq. 1 is $O(n^2 + n\bar{l}Q)$, where $\bar{l}$ is the maximum number of different demand levels for any node [12]. It should be noted that in Eq. 1, an undirected solution must be evaluated twice—once in the clockwise direction and once in the anti-clockwise direction. This is due to the fact that a vehicle with a finite capacity Q might have different recourse actions depending on whether it visits the nodes in the clockwise or the anti-clockwise direction.

For solving the VRPSDC, Gendreau et al. [12,39] proposed an exact method based on the Integer L-shaped algorithm. This method has solved instances with up to 46 nodes. An important result established by Gendreau et al. [12,39] is that the presence of probabilistic nodes makes the problem more difficult to solve than the presence of stochastic demands. To study the impact of the number of probabilistic nodes on the difficulty of solving the problem, the authors tested the exact method on instances with 1, $(n-1)/2$, and $n-1$ probabilistic nodes. The results showed that the performance of the exact method decreases with the number of probabilistic nodes. For example, to solve an instance with 11 of which 10 are probabilistic nodes, the Integer L-shaped method required up to 53903 CPU seconds on a Silicon Graphics computer with a 33 Mhz processor.

Gendreau et al. [17,39] tackled this problem using a tabu search algorithm called TABUSTOCH. For a complete description of the algorithm, we refer the reader to [17]. To the best of our knowledge, TABUSTOCH is the only metaheuristic that explicitly tackles the VRPSDC. The main novelty in TABUSTOCH is the adoption of the analytical approximation approach in delta evaluation. The authors systematically tested four analytical approximation schemes: a TSP delta evaluation scheme that considers edge costs but completely ignores node probabilities and demands; a PTSP delta evaluation that adopts edge costs weighted by node probabilities; two VRPSDC delta evaluation schemes that consider edge costs, node probabilities, and demands. After some preliminary experiments, the authors adopted the PTSP delta evaluation scheme, which they found to be more effective than the TSP and VRPSDC schemes.

It is worthwhile to focus on some implementation details of TABUSTOCH. Gendreau et al. [17] developed TABUSTOCH to tackle the VRPSDC with K vehicles, where K is a parameter. A penalized cost function is used to force the algorithm to achieve K *a priori* routes. The approximation scheme is more sophisticated when a node from one *a priori* route is moved to another *a priori* route during randomized node-insertion moves. In the experimental analysis, TABUSTOCH is applied to solve VRPSDC instances of up to 46 nodes with 2 vehicles. On the instance of size 11 with 10 probabilistic nodes, where the Integer L-shaped method needed up to 53903 CPU seconds, TABUSTOCH required only 15.46 CPU seconds to find the optimal solution on the same hardware.

There are two main factors that motivated us to develop estimation-based algorithms for the VRPSDC. First, the currently available metaheuristic, TABUSTOCH, uses a computationally expensive analytical computation approach, which affects the performance of the algorithm for large instances. The effectiveness of using the alternative empirical estimation approach has never been studied within metaheuristics to tackle the VRPSDC. Second, there is only one metaheuristic for the VRPSDC; the possibility of using other metaheuristics to tackle the VRPSDC has not been studied. However, there exists a number of previous works on estimation-based metaheuristics for other SVRPs. Some examples are a stochastic simulated annealing algorithm [40] and an ACO algorithm [41,42] for the PTSP as well as the cross entropy method [19] and a multi descent algorithm [22] for the vehicle routing problem with stochastic demands.

In this paper, we focus on a single vehicle VRPSDC because it will allow us to exactly assess the computational overhead involved in using the closed-form VRPSDC cost-function [12] and it is straightforward to extend the PTSP algorithms to the single vehicle version.

3 Estimation-based metaheuristics

In this section, we describe the proposed algorithms starting by the estimation-based evaluation procedure.

3.1 Estimation-based approach

The empirical estimation technique consists in estimating the cost $F(\pi)$ on the basis of sample costs $f(\pi, \omega_1), f(\pi, \omega_2), \ldots, f(\pi, \omega_M)$ of *a posteriori* solutions obtained from M independent realizations $\omega_1, \omega_2, \ldots, \omega_M$ of probabilities and stochastic demands of nodes. $\hat{F}_M(\pi) = \frac{1}{M} \sum_{r=1}^{M} f(\pi, \omega_r)$ is an *unbiased* estimator of $F(\pi)$. A realization comprises two components for each node i: the first component takes a value `'1'` with a probability p_i and a value `'0'` with a probability $1 - p_i$; the second component is a demand value sampled from the random distribution that describes ξ_i when the first component is `'1'`. See Fig. 1 for an illustration of *a priori* and *a posteriori* solutions. The time complexity involved in evaluating the cost of a solution by empirical estimation is $O(nM)$, which is independent from the vehicle capacity, Q, and from the maximum number of different demand levels for any node, $\bar{l}$.

Crucial to the effectiveness of the estimation approach is the number of realizations M used to estimate the cost of a solution. For this purpose, we use an adaptive sample size procedure, called ANOVA-Race [32]. It is based on sequential hypothesis testing where the sample size for testing the null hypothesis is not determined in advance, but chosen adaptively. In ANOVA-Race, for a given set of *a priori* solutions that need to be compared, their corresponding *a posteriori* solution costs are computed on a realization-by-realization basis and each time a new realization is considered, the ANOVA statistical test is applied on the computed estimates. If the ANOVA test indicates that the cost estimate of a solution is significantly worse than at least another one, the inferior solution is discarded from further evaluation. Tukeys' honestly significant differences test [43] is used to identify the inferior solution. The race continues until a single solution remains, or a maximum number M of realizations is considered.

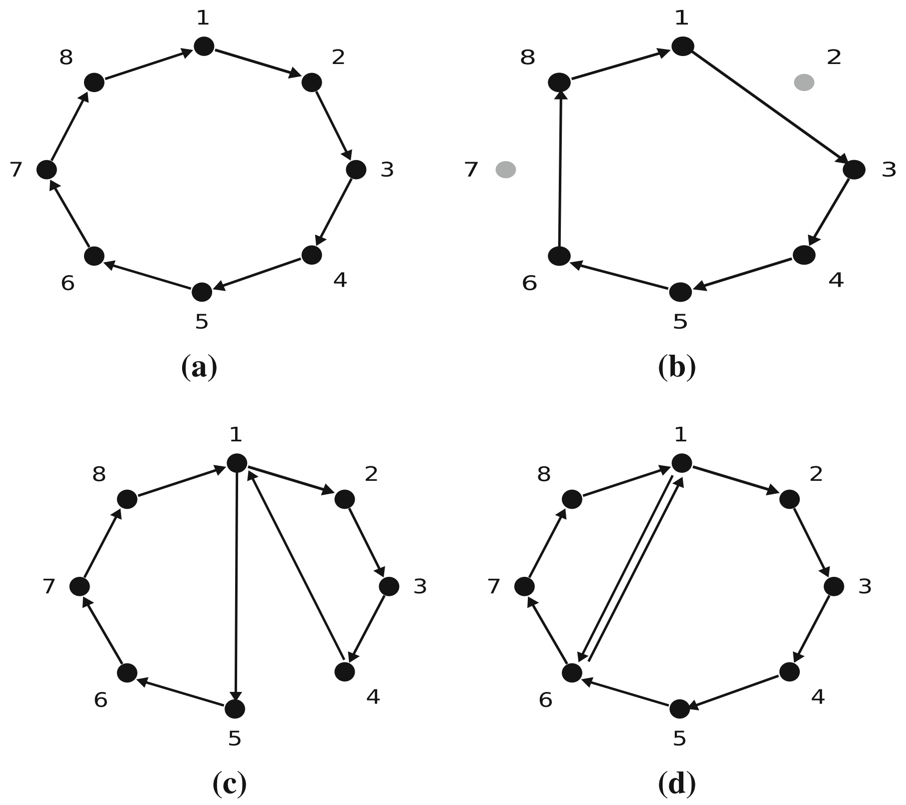

Fig. 1 Illustration of *a priori* and *a posteriori* solutions. Plot 1**a** shows an *a priori* solution for a VRPSDC instance with eight nodes, where node 1 is the depot and the nodes are visited in the following order: 1, 2, 3, 4, 5, 6, 7, 8, and 1. Plot 1**b** shows an *a posteriori* solution in which nodes 2 and 7 that do not require being visited are skipped assuming that the amount of goods available in the vehicle is enough to satisfy the demands of other nodes. Plot 1**c** shows an *a posteriori* solution in which the vehicle is empty at node 4 after serving the node. The vehicle goes back to the depot (node 1) and it resumes the delivery from the next node 5 that requires being visited. Plot 1**d** shows an *a posteriori* solution in which the amount of goods available in the vehicle is not enough to satisfy the demand of node 6. The vehicle delivers all goods, returns to the depot (node 1), and goes back to node 6 to deliver the remaining goods

In case more than one solution survive the race, the one with the least cost estimate is selected as the best one. The ANOVA-Race procedure also adopts the method of common random numbers [44] to reduce variance: a same set of realizations is used to sequentially evaluate and compare two or more solutions costs obtained at each iteration of a metaheuristic.

3.2 The `2.5-opt-EEais` algorithm

`2.5-opt-EEais` [30,31] is an iterative improvement algorithm for the PTSP. It adopts a 2.5-exchange neighborhood, which combines the 2-exchange and the node-insertion neighborhoods [45], and estimation approach within the delta evaluation. The estimation approach additionally includes an adaptive sample size procedure based on Student's t test and two variance reduction techniques: the method of common random numbers and importance sampling [44]. The algorithm also exploits standard neighborhood reduction techniques. We refer the reader to [30] and [31] for more details.

The customization of the PTSP delta evaluation in `2.5-opt-EEais` to the VRPSDC delta evaluation is not feasible because local modifications in a solution entail a global change in the cost of a solution. Therefore, we use `2.5-opt-EEais` as developed for the PTSP without any modification as a local search algorithm in all metaheuristics. Thus, in the local search phase, a PTSP approximation is used for the VRPSDC by ignoring the nodes demands and the vehicle capacity. This usage is aimed to maintain the speed advantage of `2.5-opt-EEais`. We expect that `2.5-opt-EEais` is effective for the VRPSDC because it has been shown that the stochasticity due to probabilistic nodes has more influence than the one due to the stochastic demands for solving the VRPSDC and that the PTSP approximation in TABUSTOCH is more effective than the problem-specific approximation [12,39].

3.3 Metaheuristics

In this section, we present the metaheuristics that we use to tackle the VRPSDC. We choose random restart local search as a baseline.

3.3.1 Random restart local search

In random restart local search (RRLS), a local search algorithm is repeatedly applied to new initial solutions, which are generated independently of the previously found local optima. In our implementation of RRLS, the new initial solution for the local search is generated by the nearest neighbor heuristic for the first n iterations, where n is the size of the instance. After n iterations, the algorithm uses a random solution as the new initial solution.

3.3.2 Iterated local search

Iterated local search (ILS) is similar to RRLS except that the initial solution for the local search is obtained by a perturbation of the incumbent local optimum and it is always applied to the best-so-far solution. The perturbation scheme consists in applying n_{db} random double-bridge moves, where n_{db} is a parameter, and randomly choosing ps % of the n nodes, where ps is a parameter. These nodes are removed from the solution and re-inserted according to the farthest insertion heuristic. In the context of the PTSP, insertion moves and edge-exchange moves are effective when the customer probability values are small and large, respectively [30,31,46,47]. The search is restarted from a new nearest neighbor solution when no improvement is found for $rst_{it} \cdot n$ iterations, where $rst_{it} \in [0, 1]$ is a parameter.

3.3.3 Memetic algorithms

The memetic algorithm (MA) we use for the VRPSDC is based on MAGX, a high performing TSP algorithm [48]. This algorithm starts with an initial population of *pop_size* solutions generated by a randomized variant of the greedy construction heuristic and applies local search to each of them. At each iteration, *off_frac* $\times$ *pop_size* offspring are obtained using a greedy recombination operator, where *off_frac* $\in (0, 1]$ is a parameter. This operator generates an offspring from two parent solutions in three phases: (i) all solution components that are common to the parents are copied to the offspring, (ii) new low cost solution components that are not common to the parents are added to the offspring (this is determined by a parameter p_n), and (iii) low cost solution components from the parents are added to the offspring (this is determined by a parameter p_c). The individuals for the mutation are chosen at random and the ILS hybrid perturbation mechanism parameterized by n_{db} and ps is used to mutate them. As soon as a new solution is obtained by mutation or recombination, local search is applied to it.

3.3.4 Ant colony optimization

We use ant colony system (ACS) [49] to tackle the VRPSDC. This is one of the most effective ACO algorithms for the TSP [37] and for the PTSP [32]. In this algorithm, at each iteration m ants construct solutions as follows: the probability for an ant k to move from node i to node j depends on a random variable q uniformly distributed in the interval [0,1], and a parameter q_0. If $q \leq q_0$, then, among the feasible components, the component that maximizes the product $\tau_{ij}\eta_{ij}^{\beta}$ is chosen, where τ_{ij} and $\eta_{ij} = 1/c_{ij}$ are the pheromone value and the heuristic value associated with the edge $\langle i, j\rangle$, respectively; and β is a parameter that determines the relative influence of the heuristic information as an ant traverses an edge $\langle i, j\rangle$. Otherwise, the ant k at node i chooses to move to the node j with a probability p_{ij}^k, which is given by

$$p_{ij}^k = \frac{\tau_{ij} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il} \cdot \eta_{il}^{\beta}} \qquad \text{if } j \in N_i^k, \tag{4}$$

where N_i^k is the set of feasible neighbors of node i. As soon as an ant moves from node i to node j, the pheromone value associated with the edge $\langle i, j\rangle$ is updated to $\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0$, where $\varphi \in (0, 1]$ is a parameter, and τ_0 is the initial value of the pheromone. At the end of each iteration, solutions are compared and the pheromone value associated with each edge $\langle i, j\rangle$ of the best-so-far solution is updated to $\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{best}$, where $\rho \in (0, 1]$ is a parameter and $\Delta\tau_{ij}^{best}$ is set to the inverse of the cost of the best-so-far solution. Note that local search is applied to all solutions constructed by the ants prior to the pheromone update with the best-so-far solution.

3.3.5 General remarks on the metaheuristics

In all metaheuristics, `2.5-opt-EEais` is used as the local search algorithm. To handle the stochastic element of the VRPSDC, all metaheuristics use ANOVA-Race to compare the solutions produced at each iteration. Moreover, all these solutions are undirected. Therefore, given a number of solutions that need to be compared, ANOVA-Race compares twice this number of solutions since each solution is evaluated once in clockwise direction and once in anti-clockwise direction. We denote the customized metaheuristic algorithms RRLS-EE, ILS-EE, MAGX-EE, and ACS-EE, respectively, where the suffix EE indicates that these algorithms adopt the empirical estimation approach.

4 Experimental analysis

4.1 Experimental setup

The VRPSDC instances we used for the experiments are obtained as follows. First, TSP instances are generated with the DIMACS instance generator of the TSP [50], where the nodes are arranged as a number of clusters in a $10^6 \times 10^6$ square. Four instance sizes are considered: 30, 100, 300, and 1,000. For each TSP instance, a same probability value p is assigned to all nodes except the first node, which is the depot. We considered values for p ranging from 0.050 to 0.200 with an increment of 0.025 and additional values 0.3, 0.5, 0.8, and 1.0. For generating heterogenous probabilities, we generated values using a beta distribution as described in [33]. The demand distributions for the nodes and the vehicle capacity are assigned as described in [10]: each node takes at random one of three discrete uniform distributions. The three distributions are on the values in $\{1, \dots, 9\}$, $\{5, \dots, 15\}$, and $\{10, \dots, 20\}$. In this way, for a given instance, the expected demand of a node that requires being visited is equal to 10. The vehicle capacity Q is set to $10 \times \sum_{i=2}^{n}(p/fc)$, where fc is the so-called *filling coefficient*. We consider values of fc in $\{1.0, 2.0, 4.0, 8.0\}$. The generated instances are grouped into three classes according to p: $\{0.050, 0.075, 0.100\}$ (Class I), $\{0.150, 0.175, 0.200\}$ (Class II), $\{0.300, 0.500, 0.800, 1.000\}$ (Class III).

For the experimental analysis, we implemented TABUSTOCH as described in [10] but we applied it to VRPSDC instances with only one vehicle. This includes two modifications. First, the penalization in the cost function is not used. Second, in the approximation scheme, the auxiliary computations are excluded. We use the same parameter values as suggested in [10].

All algorithms are implemented in `C` and compiled with `gcc`, version 3.3. The implementation of ACS-EE is based on ACOTSP [51]. Experiments are carried out on AMD Opteron$^{\text{TM}}$244 processors running at 1.75 GHz with 1 MB L2-Cache and 2 GB RAM, under Rocks Cluster GNU/Linux.

For all algorithms, the quadrant nearest-neighbor strategy [52,53] is used to construct the neighborhood list for each node. The length of the list is set to 40. The nearest-neighbor heuristic is used to generate the initial solution in TABUSTOCH, RRLS-EE, ILS-EE, and MAGX-EE. In ACS-EE, the candidate list for solution construction is set to size 40. The minimum number of realizations used in the adaptive sampling procedure before applying ANOVA-Race is set to five; the null hypothesis is rejected at a significance level of 0.05. All algorithms use a same set of M realizations for all iterations; however, realizations are selected randomly from this set at each iteration. The maximum number M of realizations is set to one thousand in all algorithms. The critical values of the F-distribution and Tukey's HSD test are pre-computed and stored in a look up table. The closed-form VRPSDC cost function is used for the post-evaluation of the best-so-far solutions found by all algorithms. For `2.5-opt-EEais`, we use the same parameter values obtained for the PTSP, as reported in [33].

Since in [10], the impact of the values of node probabilities on the effectiveness of the algorithms has not been investigated, we present exemplary results obtained on several instances with homogeneous node probability values. The trend of the results obtained on instances with heterogenous node probability values is similar to the homogeneous cases. Complete results and numerical values are given in an online supplementary document [54].

4.2 Effectiveness of `2.5-opt-EEais`

In this section, we show that a simple random restart local search that uses `2.5-opt-EEais`, is more effective than TABUSTOCH. Since RRLS-EE uses `2.5-opt-EEais` and ANOVA-Race, it will be difficult for us to attribute the effectiveness only to the adoption of `2.5-opt-EEais`. Therefore, we use RRLS-AC, a random restart local search similar to RRLS-EE with the exception that local optima are compared using the closed-form VRPSDC cost function. We evaluate the two algorithms on instances with 30, 100, and 300 nodes.

The stopping criterion for each instance is chosen as follows: TABUSTOCH is run until it performs 1,000 iterations and the time needed for completion is recorded. The time limit for RRLS-AC is then set to the time taken by TABUSTOCH. Table 1 shows the resulting average computation times. From the results we can observe that the instance size has a strong impact on the computation time of TABUSTOCH. In particular, the computation time increases quite drastically (more than two orders of magnitude) by increasing the instance size from 30 to 300. However, there is no considerable difference in computation time for different values of *fc*. The observed trends are due to the fact that the term n^2 in the $O(n^2 + n\bar{l}Q)$ time complexity of the closed-form VRPSDC cost function clearly dominates the other terms.

Figure 2 shows exemplary run-time development plots on instances of size 100. From these plots, we can observe that RRLS-AC completely outperforms TABUSTOCH. We can also observe from the plots that the improvement in TABUSTOCH when moving from an incumbent solution to a neighboring solution is rather small. This is typical when an algorithm moves in the space of neighboring solutions. Furthermore, it should be noted that, for a fixed value of *fc*, the relative difference in the average solution cost between TABUSTOCH and RRLS-AC increases with an increase of p. This is due to the adoption of the randomized node-insertion neighborhood in TABUSTOCH, which we guess becomes less effective for large values of p.

Table 1 Computation time (in CPU seconds) needed by TABUSTOCH to complete 1,000 iterations.

$p(fc)$	$n = 30$		$n = 100$		$n = 300$	
	mean	s.d.	mean	s.d.	mean	s.d.
0.100(1.00)	38	1	502	10	5457	114
0.100(8.00)	40	1	526	13	5727	99
0.200(1.00)	37	1	487	11	5268	130
0.200(8.00)	40	1	521	8	5347	149
0.300(1.00)	37	0	473	8	4999	91
0.300(8.00)	40	1	494	11	5035	135
0.500(1.00)	36	0	464	10	4842	78
0.500(8.00)	40	0	477	8	4917	122
0.800(1.00)	36	1	457	6	4738	55
0.800(8.00)	38	0	466	5	4781	78
1.000(1.00)	36	1	456	5	4638	60
1.000(8.00)	37	0	464	5	4749	78

For each combination of p and fc, the mean and the standard deviation (s.d.) are computed over 10 instances of size 30, 100, and 300

Note that for the node-insertion neighborhood, we also observed a similar behavior in iterative improvement algorithms for the PTSP [30,31].

The high performance of RRLS-AC is due to the adoption of `2.5-opt-EEais`. This can be inferred from Figure 2, which shows that for RRLS-AC there is a large improvement within a short computation time (about 5 s). The traces of RRLS-AC show that this large improvement is achieved in the very first iteration after the first run of `2.5-opt-EEais` on the initial solution.

For a given value of p, the relative difference in the average solution cost between the two algorithms decreases with an increase in the value of fc. This can be explained as follows: the number of recourse actions in which the vehicle has to return back to the depot for replenishment increases with an increase in the value of fc. Since this recourse action is ignored by `2.5-opt-EEais`, the quality of the local optima found by `2.5-opt-EEais` decreases with an increase of fc.

Table 2 reports the observed relative difference between the final solution cost achieved by the two algorithms, with a 95 % confidence bound obtained through a t test. For the absolute values, we refer the reader to [54]. The results show as general trend, that RRLS-AC is better than TABUSTOCH across a wide range of instance sizes, node probabilities, and vehicle capacities. Most of the observed differences are significant according to the t test.

4.3 Effectiveness of ANOVA-Race

In this section, we assess the effectiveness of the estimation-based evaluation procedure, ANOVA-Race, by comparing RRLS-EE to RRLS-AC. The two algorithms differ only with respect to the evaluation procedure. Given that the PTSP-specific

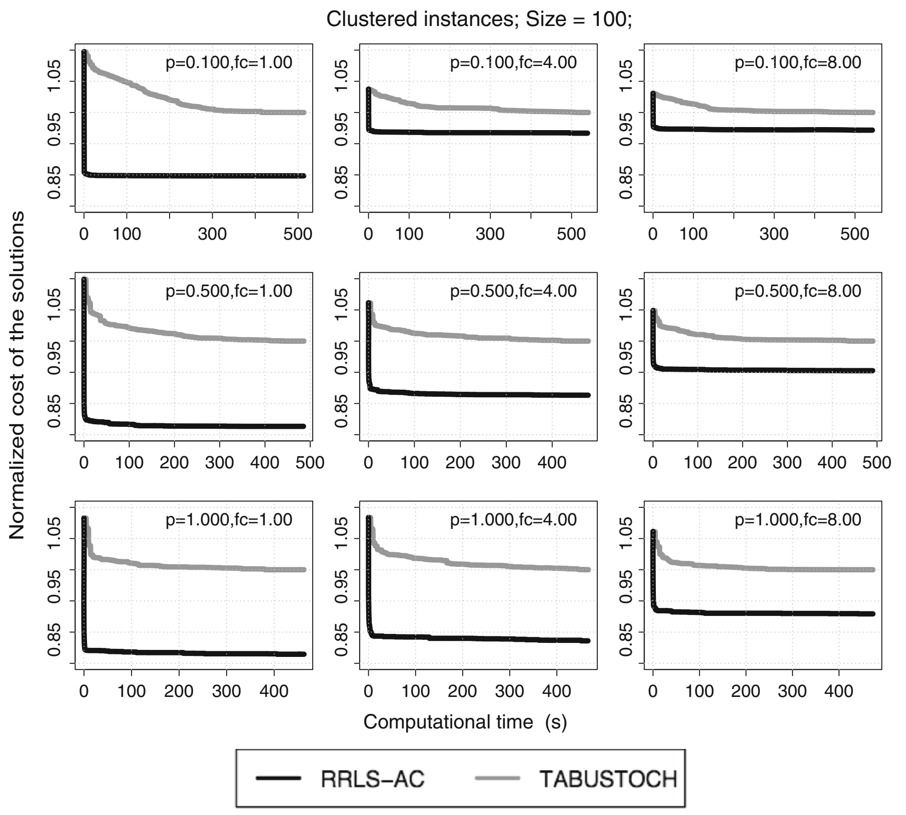

Fig. 2 Experimental results on clustered VRPSDC instances of size 100. The plots represent the development of solution cost over computation time for RRLS-AC and TABUSTOCH. Solution costs are normalized with respect to the best solution found by TABUSTOCH on an instance-by-instance basis for ten instances; the normalized solution cost is then aggregated

`2.5-opt-EEais` obtained high quality solutions, we include in our analysis the RRLS-EE algorithm developed for the PTSP. We denote this algorithm RRLS-EE(PTSP). Note that RRLS-EE(PTSP) ignores node demands and uses the *t* test to compare two local optima.

The three algorithms are compared on three instance sizes: 100, 300, and 1,000. For each level of instance size and for each combination of *p* and *fc*, we use 10 instances for the comparison. Since the time limit used in Sect. 4.2 is rather high, we allow the three algorithms to run for *n* CPU seconds, where *n* is the size of the instance.

Figure 3 shows the run-time development plots obtained on instances of size 1,000. Note that in the plots the computation time is shown in log scale because the difference in speed between the two algorithms is rather large. From the plots we can observe that RRLS-EE obtains the average solution cost of RRLS-AC in between 0.5 and 2 orders of magnitude less CPU time. From the run-time development plots reported in [54], we can observe the following: RRLS-EE is faster than RRLS-AC by a factor of 1 to 2 on instances of size 300; we could not observe considerable speedup for instances of size 100.

Table 2 Comparison of the average cost obtained by RRLS-AC and TABUSTOCH over ten instances of size 30 and 300.

	$n = 30$		$n = 300$	
$p(fc)$	d	[95 % CI]	d	[95 % CI]
0.100(1.00)	+0.04	[−1.53, +1.62]	**−12.58**	[−14.37, −10.78]
0.100(8.00)	*+1.85*	[+0.58, +3.13]	**−3.40**	[−4.39, −2.41]
0.200(1.00)	**−2.94**	[−5.34, −0.53]	**−14.32**	[−17.21, −11.44]
0.200(8.00)	−0.21	[−2.38, +1.97]	**−5.36**	[−6.35, −4.36]
0.300(1.00)	**−5.78**	[−10.31, −1.24]	**−15.76**	[−18.48, −13.05]
0.300(8.00)	**−2.08**	[−3.46, −0.70]	**−6.74**	[−7.53, −5.94]
0.500(1.00)	**−8.02**	[−13.34, −2.70]	**−14.69**	[−18.57, −10.81]
0.500(8.00)	**−2.63**	[−3.70, −1.56]	**−6.55**	[−7.76, −5.34]
0.800(1.00)	**−12.46**	[−18.41, −6.50]	**−12.56**	[−15.64, −9.48]
0.800(8.00)	**−3.15**	[−4.58, −1.72]	**−7.17**	[−8.51, −5.82]
1.000(1.00)	**−11.98**	[−16.36, −7.59]	**−14.49**	[−17.41, −11.57]
1.000(8.00)	**−3.85**	[−5.38, −2.31]	**−8.92**	[−11.06, −6.79]

Explanation of the contents and the typographic conventions For a given comparison A versus B, the table reports the observed relative difference d between the two algorithms A and B and the 95 % confidence interval, CI, obtained through the t test. If the value is positive, algorithm A obtained an average cost that is larger than the one obtained by algorithm B. In this case, the value is typeset in italics if it is significantly different from zero, according to the t test, at a confidence level of 95 %. If the value is negative, algorithm A obtained an average cost that is smaller than the one obtained by algorithm B. In this case, the value is typeset in boldface if it is significantly different from zero, according to the t test, at a confidence level of 95 %

Table 3 shows the average number of iterations performed by the two algorithms for the given computation time. On instances of size 1,000, the average number of iterations increases with an increase in the value of p: it increases from 34 to 43 in RRLS-AC, while in RRLS-EE, it increases from 330 to 3006. This is due to the effectiveness of ANOVA-Race, which needs only few realizations to select the best solution for large values of p. Although the average number of iterations performed by RRLS-EE(PTSP) is slightly higher than that of RRLS-EE, the observed differences are rather small. This shows that the VRPSDC -specific evaluation does not involve a large computational overhead for taking into account the stochastic demands. We can observe a similar trend for instances of size 100 and 300 [54].

Table 4 shows the difference in the average cost between the three algorithms on instances of size 1,000. The results confirm that the VRPSDC-specific estimation approach is more effective than the PTSP-specific estimation approach and the analytical computation approach. RRLS-EE obtains average solution costs that are between 0.23% and 1.90% lower than those of RRLS-AC. Although the difference in the number of iterations between RRLS-EE and RRLS-AC is quite large, the difference in the final solution cost is rather small. This is due to the adoption of `2.5-opt-EEais`, which, as shown in Sect. 4.2, produces a large improvement at the first iteration. Concerning the comparison between RRLS-EE and RRLS-EE(PTSP), we can observe that the two algorithms achieve similar average costs for $fc = 1.0$, where the instances

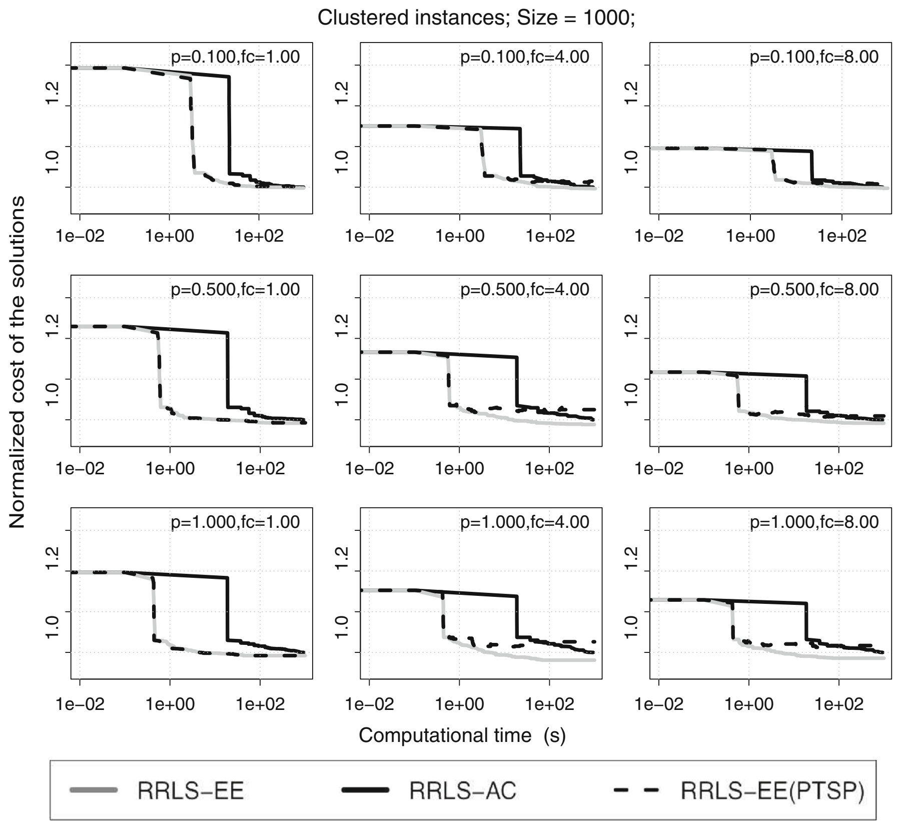

Fig. 3 Experimental results on clustered VRPSDC instances of size 1,000 for 1,000 CPU seconds. The plots represent the development of the solution cost over computation time for RRLS-EE, RRLS-EE(PTSP), and RRLS-AC. Solution costs are normalized with respect to the best solution found by RRLS-AC on an instance-by-instance basis for ten instances; the normalized solution cost is then aggregated

are similar to PTSP instances. However, as fc increases, the PTSP-specific estimation approach becomes less effective. Note that for a given value of p, the difference in the average cost between RRLS-EE and RRLS-EE(PTSP) increases up to $fc = 4.0$; the observed difference for $fc = 8.0$ is less than that of $fc = 4.0$. The results on instances of size 100 and 300, which are given in [54], shows that the difference is less for smaller instances.

4.4 Comparison between estimation-based metaheuristics

In this section, we compare ILS-EE, MAGX-EE, ACS-EE, and RRLS-EE on a new set of instances with 100, 300, and 1000 nodes. We allow each algorithm to run for n CPU seconds, where n is the size of the instance.

First, we tune the parameters of ILS-EE, MAGX-EE, and ACS-EE using Iterated F-Race [55,56]. For the tuning task, we use a different set of clustered instances with

Table 3 The average number of iterations performed by RRLS-EE and RRLS-AC.

$p(fc)$	$n = 100$			$n = 1,000$		
	RRLS-EE	RRLS-AC	RRLS-EE (PTSP)	RRLS-EE	RRLS-AC	RRLS-EE (PTSP)
0.100(1.00)	269	220	278	330	34	340
0.100(8.00)	268	216	278	326	30	336
0.200(1.00)	529	364	556	969	43	998
0.200(8.00)	514	353	554	962	41	999
0.300(1.00)	797	464	825	1243	43	1254
0.300(8.00)	759	452	832	1238	48	1248
0.500(1.00)	1364	594	1408	1645	44	1662
0.500(8.00)	1302	581	1402	1650	46	1662
0.800(1.00)	2394	715	2445	2244	41	2239
0.800(8.00)	2383	705	2435	2249	41	2259
1.000(1.00)	3938	800	3976	2985	43	3038
1.000(8.00)	3940	791	3975	3006	43	3014

Each algorithm is allowed to run for n CPU seconds, where n is the size of the instance

1,000 nodes: we generated 120 instances (ten instances times three values of p times four values of fc) for instance Class I and Class II, respectively, and 160 instances (ten instances times four values of p times four values of fc) for instance Class III. Iterated F-Race is run nine times (three metaheuristics times three instance classes), each time with a computational budget of 1,000 runs. Each run of a metaheuristic is given a stopping criterion of n (=1,000) CPU seconds. Since Iterated F-Race is a stochastic algorithm, the parameter tuning for the VRPSDC algorithms is repeated ten times. For each metaheuristic and each instance class, we have a set of ten fine tuned parameter configurations. The obtained parameter configurations are reported in [54].

To study the cost of the solutions obtained by each algorithm, we use the expected solution cost obtained by a metaheuristic, where the expectation is taken with respect to the distribution of tuned parameter configurations (ten parameter configurations) and the distribution of test instances. The costs of the solutions obtained by ILS-EE, MAGX-EE, and ACS-EE are normalized by the final solution cost reached by RRLS-EE. The normalization is done on an instance-by-instance basis on ten instances for each combination of p and fc.

Figure 4 shows exemplary run-time development plots of the four estimation-based algorithms for up to 1,000 CPU seconds on instance size 1,000. For MAGX-EE and ACS-EE, the plots take into account the improvement obtained by the first local search applied to an individual of the population. Due to the adoption of `2.5-opt-EEais`, in all algorithms the initial solution is improved between 10 and 40 % in a very short computation time. Further improvements in the following iterations are considerably smaller than that of the first iteration.

Figure 5 shows the box plot of the solution cost of the algorithms on instance size 1,000. We can observe that ILS-EE, MAGX-EE, and ACS-EE outperform RRLS-EE

Table 4 Comparison of the average cost obtained by RRLS-EE and RRLS-AC over ten instances of size 1,000 for each parameter setting in the instance generation.

$p(fc)$	$n = 1,000$ RRLS-EE versus RRLS-AC		RRLS-EE versus RRLS-EE(PTSP)		RRLS-AC versus RRLS-EE(PTSP)	
	d	[95 % CI]	d	[95 % CI]	d	[95 % CI]
0.100(1.00)	**−0.23**	[−0.40, −0.06]	−0.45	[−0.94, +0.04]	−0.22	[−0.65, +0.21]
0.100(8.00)	**−0.26**	[−0.43, −0.09]	**−1.14**	[−1.73, −0.55]	**−0.88**	[−1.46, −0.30]
0.200(1.00)	**−0.94**	[−1.29, −0.58]	−0.24	[−0.49, +0.01]	*+0.70*	[+0.23, +1.17]
0.200(8.00)	**−0.58**	[−0.81, −0.35]	**−1.21**	[−1.97, −0.46]	−0.64	[−1.47, +0.19]
0.300(1.00)	**−0.73**	[−1.16, −0.31]	**−0.24**	[−0.47, −0.01]	*+0.50*	[+0.02, +0.97]
0.300(8.00)	**−0.32**	[−0.48, −0.17]	**−1.22**	[−2.02, −0.41]	**−0.90**	[−1.73, −0.06]
0.500(1.00)	**−0.83**	[−1.18, −0.48]	−0.10	[−0.26, +0.06]	*+0.73*	[+0.28, +1.19]
0.500(8.00)	**−0.85**	[−1.20, −0.51]	**−1.78**	[−2.72, −0.84]	−0.94	[−1.95, +0.08]
0.800(1.00)	**−0.76**	[−1.11, −0.40]	−0.03	[−0.08, +0.02]	*+0.73*	[+0.36, +1.10]
0.800(8.00)	**−0.57**	[−1.05, −0.09]	**−2.19**	[−3.30, −1.09]	**−1.63**	[−2.75, −0.51]
1.000(1.00)	**−0.79**	[−1.18, −0.39]	0.00	[0.00, 0.00]	*+0.79*	[+0.40, +1.19]
1.000(8.00)	**−1.42**	[−2.04, −0.80]	**−3.06**	[−4.51, −1.61]	−1.66	[−3.36, +0.03]

Typographic conventions are the same as in Table 2.

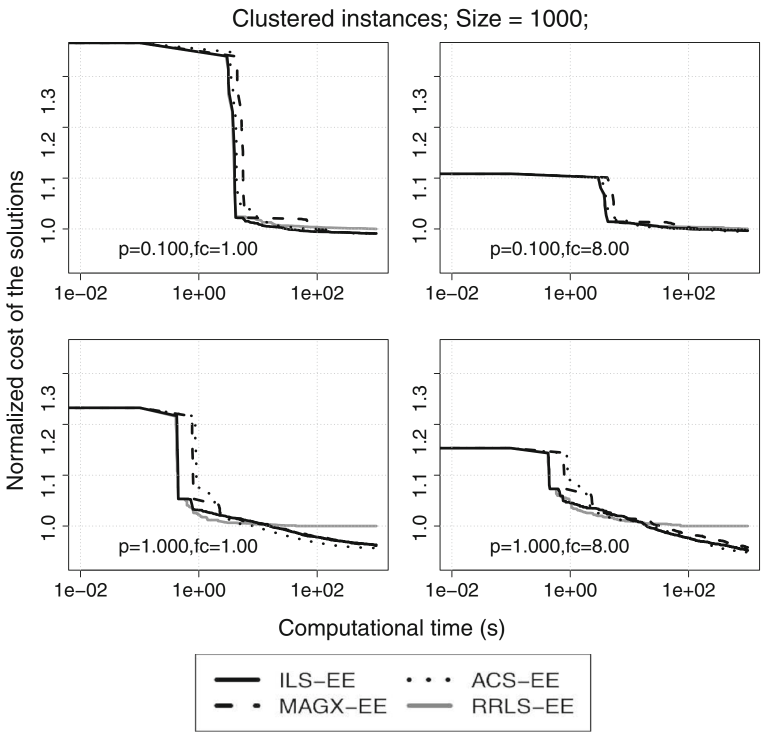

Fig. 4 Experimental results on clustered VRPSDC instances of size 1,000 for 1,000 CPU seconds. The plots represent the cost of the solutions obtained by ILS-EE, MAGX-EE, and ACS-EE normalized by the one obtained by RRLS-EE. The normalization is done on an instance-by-instance basis for ten instances; the normalized solution cost is then aggregated

across most probability levels. The difference in the solution cost between RRLS-EE and the other algorithms increases with an increase in instance size—see the online supplementary document [54]. For a given instance size, the observed differences in the solution cost between the four estimation-based algorithms increase with an increase in the node probability p. This shows that for instances with small values of p, it is rather easy to find high quality solutions by restarting `2.5-opt-EEais` a number of times. Nevertheless, for instances with large values of p, in addition to `2.5-opt-EEais`, the use of sophisticated metaheuristics is crucial to find high quality solutions.

Table 5 reports the observed relative difference between the solution costs obtained by the algorithms, with a 95 % confidence bound given by the t test. The results confirm that the three estimation-based metaheuristics are more effective than RRLS-EE and that they obtain average solution costs that are significantly less than the cost of the best solution obtained by RRLS-EE on a wide range of instance sizes and probability levels: ILS-EE, MAGX-EE, and ACS-EE obtain average solution costs that are 2.32, 2.19, and 2.55 % less than that of RRLS-EE, respectively. The differences in the average solution costs between ILS-EE, MAGX-EE, and ACS-EE are rather small and the observed differences are less than 1 %. On Class I instances, the average solution cost

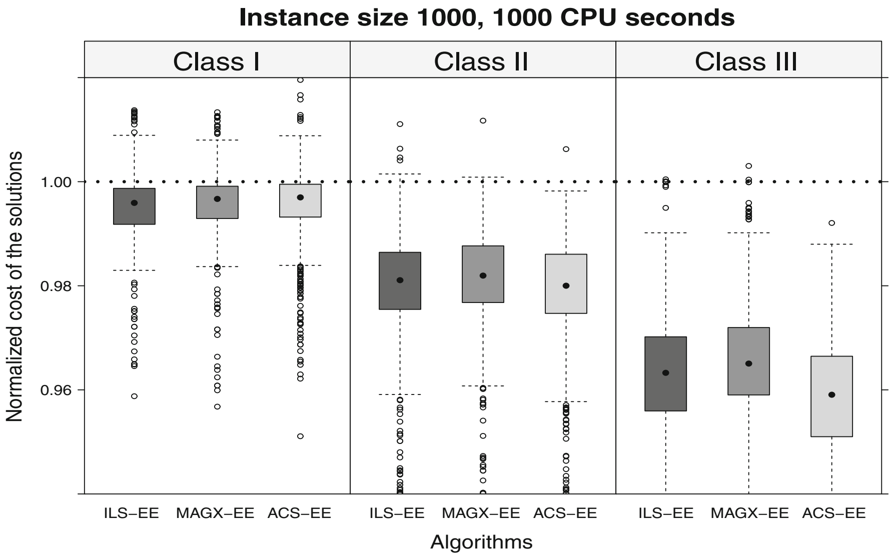

Fig. 5 Experimental results on clustered VRPSDC instances. The *box plots* represent the normalized cost of the solutions obtained by ILS-EE, MAGX-EE, and ACS-EE. The obtained solution costs of the algorithms are normalized by the final solution cost reached by RRLS-EE. The normalization is done on an instance-by-instance basis for 10 instances; the normalized solution cost is then aggregated. The *dotted horizontal line* denotes therefore the final cost of RRLS-EE

obtained by ILS-EE is 0.09 and 0.07 % less than that of MAGX-EE and ACS-EE, respectively. On Class II instances, the average solution cost obtained by ACS-EE is 0.08 and 0.19 % less than that of ILS-EE and MAGX-EE, respectively. On Class III instances, the observed differences are 0.60 and 0.79 %. The aggregated results over all instances show that ACS-EE is more effective than ILS-EE and MAGX-EE: the average solution cost obtained by ACS-EE is 0.23 and 0.37 % lower than that of ILS-EE and MAGX-EE, respectively. The general trends of the results on instances of size 100 and 300, which are reported in [54], are consistent with the results presented here except that the observed differences are smaller.

5 Conclusions

In this paper, we customized a number of metaheuristic algorithms to tackle the vehicle routing problem with stochastic demands and customers (VRPSDC). The customization primarily consists in using a VRPSDC-specific cost evaluation procedure and in tuning the parameters of the algorithms for the VRPSDC. All algorithms use the `2.5-opt-EEais` local search, the state-of-the-art iterative improvement algorithm for the PTSP.

The two main contributions of the paper are the following. From a methodological perspective, we tackle for the first time the VRPSDC using the empirical estimation approach and we show that this approach is more effective than the previously proposed analytical computation approach, particularly on large instances. From an algorithmic

Table 5 Comparison of the average cost obtained by ILS-EE, MAGX-EE, ACS-EE and RRLS-EE, on clustered instances of size 1,000 for 1,000 CPU seconds.

	$p(fc)$	ILS-EE versus MAGX-EE		ILS-EE versus ACS-EE		ILS-EE versus RRLS-EE		MAGX-EE versus ACS-EE		MAGX-EE versus RRLS-EE		ACS-EE versus RRLS-EE	
		d	[95% CI]	d	[95% CI]	d	[95% CI]	d	[95% CI]	d	[95% CI]	d	[95% CI]
Class I	0.050(1.00)	**−0.05**	[−0.08, −0.03]	**−0.11**	[−0.13, −0.08]	**−0.20**	[−0.23, −0.16]	**−0.05**	[−0.08, −0.03]	**−0.14**	[−0.19, −0.10]	**−0.09**	[−0.13, −0.05]
	0.050(2.00)	−0.03	[−0.23, +0.18]	−0.11	[−0.31, +0.09]	+0.01	[−0.21, +0.23]	−0.08	[−0.30, +0.13]	+0.04	[−0.18, +0.25]	+0.12	[−0.10, +0.35]
	0.050(4.00)	**−0.22**	[−0.30, −0.14]	**−0.13**	[−0.21, −0.05]	**−0.20**	[−0.27, −0.12]	*+0.09*	[+0.01, +0.18]	+0.03	[−0.05, +0.10]	**−0.07**	[−0.13, −0.01]
	0.050(8.00)	**−0.04**	[−0.08, −0.01]	−0.01	[−0.05, +0.04]	+0.05	[−0.00, +0.11]	+0.03	[−0.01, +0.08]	*+0.09*	[+0.04, +0.15]	*+0.06*	[+0.02, +0.11]
	0.075(1.00)	**−0.04**	[−0.08, −0.01]	**−0.09**	[−0.14, −0.05]	**−0.47**	[−0.52, −0.43]	**−0.05**	[−0.08, −0.02]	**−0.43**	[−0.47, −0.39]	**−0.38**	[−0.42, −0.34]
	0.075(2.00)	−0.12	[−0.24, +0.00]	−0.12	[−0.27, +0.03]	**−0.99**	[−1.18, −0.80]	−0.00	[−0.15, +0.15]	**−0.87**	[−1.05, −0.70]	**−0.87**	[−1.05, −0.69]
	0.075(4.00)	**−0.13**	[−0.22, −0.04]	−0.11	[−0.26, +0.04]	**−0.44**	[−0.54, −0.33]	+0.02	[−0.11, +0.14]	**−0.31**	[−0.41, −0.21]	**−0.33**	[−0.43, −0.22]
	0.075(8.00)	**−0.08**	[−0.15, −0.00]	−0.03	[−0.10, +0.03]	**−0.29**	[−0.37, −0.21]	+0.04	[−0.03, +0.12]	**−0.21**	[−0.29, −0.13]	**−0.25**	[−0.31, −0.20]
	0.100(1.00)	**−0.10**	[−0.14, −0.05]	**−0.15**	[−0.20, −0.10]	**−1.02**	[−1.07, −0.97]	−0.05	[−0.11, +0.01]	**−0.92**	[−0.99, −0.86]	**−0.87**	[−0.93, −0.81]
	0.100(2.00)	−0.11	[−0.25, +0.02]	−0.07	[−0.23, +0.08]	**−1.15**	[−1.24, −1.06]	+0.04	[−0.14, +0.22]	**−1.04**	[−1.18, −0.90]	**−1.08**	[−1.22, −0.93]
	0.100(4.00)	−0.07	[−0.16, +0.03]	−0.06	[−0.18, +0.06]	**−0.67**	[−0.76, −0.58]	+0.00	[−0.11, +0.12]	**−0.61**	[−0.70, −0.51]	**−0.61**	[−0.72, −0.49]
	0.100(8.00)	−0.05	[−0.11, +0.01]	*+0.11*	[+0.01, +0.21]	**−0.36**	[−0.48, −0.24]	*+0.16*	[+0.05, +0.27]	**−0.31**	[−0.41, −0.20]	**−0.47**	[−0.60, −0.34]
	overall	**−0.09**	[−0.11, −0.06]	**−0.07**	[−0.11, −0.04]	**−0.48**	[−0.51, −0.44]	+0.01	[−0.02, +0.05]	**−0.39**	[−0.43, −0.35]	**−0.40**	[−0.44, −0.36]
Class II	0.150(1.00)	**−0.10**	[−0.16, −0.05]	+0.04	[−0.04, +0.12]	**−1.62**	[−1.70, −1.54]	*+0.14*	[+0.06, +0.22]	**−1.52**	[−1.60, −1.44]	**−1.66**	[−1.75, −1.57]
	0.150(2.00)	**−0.22**	[−0.32, −0.11]	−0.11	[−0.28, +0.06]	**−2.50**	[−2.75, −2.25]	+0.11	[−0.04, +0.25]	**−2.29**	[−2.54, −2.04]	**−2.39**	[−2.61, −2.16]
	0.150(4.00)	−0.04	[−0.17, +0.10]	+0.08	[−0.05, +0.21]	**−1.54**	[−1.74, −1.34]	*+0.11*	[+0.00, +0.23]	**−1.51**	[−1.69, −1.32]	**−1.62**	[−1.79, −1.45]
	0.150(8.00)	**−0.10**	[−0.18, −0.02]	+0.07	[−0.03, +0.18]	**−1.16**	[−1.27, −1.05]	*+0.17*	[+0.07, +0.28]	**−1.06**	[−1.17, −0.95]	**−1.23**	[−1.38, −1.09]
	0.175(1.00)	−0.02	[−0.09, +0.06]	+0.07	[−0.01, +0.14]	**−1.99**	[−2.08, −1.91]	*+0.08*	[+0.03, +0.14]	**−1.98**	[−2.06, −1.89]	**−2.06**	[−2.15, −1.97]
	0.175(2.00)	**−0.18**	[−0.28, −0.08]	+0.04	[−0.10, +0.19]	**−2.62**	[−2.87, −2.36]	*+0.22*	[+0.07, +0.38]	**−2.44**	[−2.69, −2.19]	**−2.66**	[−2.92, −2.40]

Table 5 continued

	$p(fc)$	ILS-EE versus MAGX-EE		ILS-EE versus ACS-EE		ILS-EE versus RRLS-EE		MAGX-EE versus ACS-EE		MAGX-EE versus RRLS-EE		ACS-EE versus RRLS-EE	
		d	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]
	0.175(4.00)	−0.09	[−0.22, +0.04]	+0.12	[−0.02, +0.26]	**−2.02**	[−2.15, −1.90]	*+0.22*	[+0.07, +0.36]	**−1.93**	[−2.04, −1.83]	**−2.14**	[−2.29, (2.00)]
	0.175(8.00)	−0.06	[−0.14, +0.03]	*+0.15*	[+0.05, +0.24]	**−1.22**	[−1.29, −1.14]	*+0.20*	[+0.10, +0.31]	**−1.16**	[−1.24, −1.08]	**−1.36**	[−1.47, −1.25]
	0.200(1.00)	**−0.08**	[−0.16, −0.01]	+0.02	[−0.06, +0.10]	**−2.30**	[−2.38, −2.21]	*+0.10*	[+0.02, +0.18]	**−2.22**	[−2.31, −2.12]	**−2.32**	[−2.40, −2.23]
	0.200(2.00)	**−0.23**	[−0.41, −0.04]	+0.11	[−0.06, +0.28]	**−2.90**	[−3.10, −2.69]	*+0.34*	[+0.20, +0.48]	**−2.67**	[−2.83, −2.52]	**−3.00**	[−3.17, −2.84]
	0.200(4.00)	−0.05	[−0.19, +0.10]	*+0.19*	[+0.04, +0.35]	**−2.50**	[−2.67, −2.34]	*+0.24*	[+0.07, +0.41]	**−2.46**	[−2.62, −2.30]	**−2.69**	[−2.87, −2.51]
	0.200(8.00)	**−0.13**	[−0.23, −0.03]	*+0.18*	[+0.10, +0.27]	**−1.44**	[−1.53, −1.35]	*+0.31*	[+0.22, +0.41]	**−1.31**	[−1.39, −1.23]	**−1.62**	[−1.72, −1.51]
	overall	**−0.11**	[−0.14, −0.08]	*+0.08*	[+0.05, +0.12]	**−1.98**	[−2.04, −1.93]	*+0.19*	[+0.15, +0.22]	**−1.88**	[−1.93, −1.83]	**−2.06**	[−2.12, −2.01]
Class III	0.300(1.00)	−0.08	[−0.22, +0.06]	+0.01	[−0.13, +0.14]	**−3.38**	[−3.53, −3.24]	*+0.09*	[+0.01, +0.16]	**−3.30**	[−3.40, −3.21]	**−3.39**	[−3.47, −3.31]
	0.300(2.00)	−0.12	[−0.32, +0.09]	*+0.20*	[+0.01, +0.40]	**−3.79**	[−3.99, −3.58]	*+0.32*	[+0.16, +0.48]	**−3.67**	[−3.83, −3.52]	**−3.98**	[−4.18, −3.78]
	0.300(4.00)	**−0.28**	[−0.48, −0.08]	+0.04	[−0.16, +0.24]	**−3.43**	[−3.65, −3.21]	*+0.32*	[+0.12, +0.52]	**−3.16**	[−3.38, −2.94]	**−3.47**	[−3.63, −3.31]
	0.300(8.00)	**−0.27**	[−0.39, −0.15]	−0.07	[−0.17, +0.03]	**−2.10**	[−2.20, (2.00)]	*+0.20*	[+0.07, +0.33]	**−1.84**	[−1.94, −1.73]	**−2.03**	[−2.11, −1.95]
	0.500(1.00)	+0.13	[−0.06, +0.31]	*+0.48*	[+0.29, +0.67]	**−3.67**	[−3.84, −3.50]	*+0.35*	[+0.27, +0.42]	**−3.80**	[−3.86, −3.74]	**−4.13**	[−4.19, −4.07]
	0.500(2.00)	−0.11	[−0.37, +0.15]	*+0.77*	[+0.52, +1.02]	**−4.54**	[−4.86, −4.23]	*+0.88*	[+0.69, +1.07]	**−4.44**	[−4.69, −4.19]	**−5.27**	[−5.53, −5.01]
	0.500(4.00)	−0.09	[−0.37, +0.19]	*+0.68*	[+0.39, +0.97]	**−3.98**	[−4.26, −3.69]	*+0.77*	[+0.60, +0.94]	**−3.90**	[−4.11, −3.68]	**−4.63**	[−4.82, −4.44]
	0.500(8.00)	+0.00	[−0.14, +0.14]	*+0.30*	[+0.14, +0.45]	**−2.98**	[−3.09, −2.86]	*+0.29*	[+0.15, +0.44]	**−2.98**	[−3.09, −2.87]	**−3.26**	[−3.39, −3.14]
	0.800(1.00)	**−0.32**	[−0.45, −0.18]	*+0.68*	[+0.54, +0.81]	**−3.60**	[−3.76, −3.45]	*+1.00*	[+0.88, +1.12]	**−3.30**	[−3.45, −3.15]	**−4.25**	[−4.36, −4.15]
	0.800(2.00)	**−0.93**	[−1.22, −0.64]	*+0.66*	[+0.43, +0.89]	**−5.10**	[−5.35, −4.84]	*+1.61*	[+1.35, +1.86]	**−4.21**	[−4.45, −3.96]	**−5.72**	[−5.94, −5.50]
	0.800(4.00)	**−0.64**	[−0.91, −0.37]	*+1.01*	[+0.80, +1.22]	**−4.50**	[−4.77, −4.24]	*+1.66*	[+1.40, +1.92]	**−3.89**	[−4.17, −3.60]	**−5.46**	[−5.70, −5.22]
	0.800(8.00)	**−0.43**	[−0.64, −0.21]	*+0.83*	[+0.63, +1.03]	**−3.79**	[−3.99, −3.59]	*+1.27*	[+1.10, +1.43]	**−3.38**	[−3.54, −3.22]	**−4.59**	[−4.73, −4.44]

Table 5 continued

$p(fc)$	ILS-EE versus MAGX-EE		ILS-EE versus ACS-EE		ILS-EE versus RRLS-EE		MAGX-EE versus ACS-EE		MAGX-EE versus RRLS-EE		ACS-EE versus RRLS-EE	
	d	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]	*d*	[95% CI]
1.000(1.00)	*+0.29*	[+0.16, +0.43]	*+0.85*	[+0.71, +0.98]	**−3.63**	[−3.77, −3.49]	*+0.55*	[+0.46, +0.64]	**−3.91**	[−4.02, −3.81]	**−4.44**	[−4.53, −4.35]
1.000(2.00)	−0.08	[−0.34, +0.17]	*+1.10*	[+0.84, +1.36]	**−5.28**	[−5.55, −5.00]	*+1.18*	[+0.94, +1.43]	**−5.20**	[−5.42, −4.97]	**−6.30**	[−6.52, −6.09]
1.000(4.00)	−0.14	[−0.42, +0.15]	*+1.17*	[+0.90, +1.45]	**−4.82**	[−5.16, −4.47]	*+1.31*	[+1.09, +1.53]	**−4.69**	[−5.02, −4.35]	**−5.92**	[−6.22, −5.62]
1.000(8.00)	−0.07	[−0.35, +0.20]	*+0.89*	[+0.60, +1.18]	**−4.70**	[−4.97, −4.44]	*+0.97*	[+0.74, +1.19]	**−4.63**	[−4.84, −4.43]	**−5.55**	[−5.77, −5.33]
overall	**−0.20**	[−0.25, −0.14]	*+0.60*	[+0.54, +0.65]	**−3.96**	[−4.02, −3.89]	*+0.79*	[+0.74, +0.84]	**−3.77**	[−3.83, −3.71]	**−4.52**	[−4.60, −4.45]
overall (3 classes)	**−0.14**	[−0.16, −0.11]	*+0.23*	[+0.21, +0.26]	**−2.32**	[−2.38, −2.26]	*+0.37*	[+0.34, +0.40]	**−2.19**	[−2.24, −2.13]	**−2.55**	[−2.61, −2.49]

Typographic conventions are the same as in Table 2.

point-of-view, we showed that the proposed estimation-based algorithms are more effective than the existing analytical computation tabu search algorithm. This allowed us to obtain new state-of-the-art algorithms for the VRPSDC.

Further work will be devoted to extend the proposed estimation-based algorithms to the VRPSDC with multiple vehicles. Given the observed superior performance of the PTSP-specific iterative improvement algorithm for the VRPSDC and the previously proposed analytical approximation scheme that ignores stochastic demands, it seems that the stochastic demand element, which makes delta evaluation difficult, is not crucial on the tested class of instances. Also, for the related vehicle routing problem with stochastic demands (VRPSD), Bianchi et al. [20] showed that the TSP approximation is better than the problem-specific delta evaluation. In this context, the impact of stochastic demands in the VRPSD and the VRPSDC needs further investigation.

Acknowledgments This research has been supported by "E-SWARM – Engineering Swarm Intelligence Systems", an European Research Council Advanced Grant awarded to Marco Dorigo (Grant Number 246939). The authors acknowledge support from the *Fonds de la Recherche Scientifique, F.R.S.-FNRS* of the French Community of Belgium.

References

1. Bertsimas, D.: Probabilistic combinatorial optimization problems. PhD Thesis, Massachusetts Institute of Technology, Cambridge, (1988)
2. Jaillet, P.: Probabilistic traveling salesman problems. PhD Thesis, Massachusetts Institute of Technology, Cambridge, (1985)
3. Jaillet, P.: A priori solution of a travelling salesman problem in which a random subset of the customers are visited. Oper. Res. **36**(6), 929–936 (1988)
4. Jézéquel, A.: Probabilistic vehicle routing problems. Master's Thesis, Massachusetts Institute of Technology, Cambridge, (1985)
5. Bertsimas, D., Jaillet, P., Odoni, A.: A priori optimization. Oper. Res. **38**(6), 1019–1033 (1990)
6. Tillman, F.: The multiple terminal delivery problem with probabilistic demands. Transp. Sci. **3**(3), 192–204 (1969)
7. Bertsimas, D.J.: A vehicle routing problem with stochastic demand. Oper. Res. **40**(3), 574–585 (1992)
8. Laporte, G., Louveaux, F., Mercure, H.: The vehicle routing problem with stochastic travel times. Transp. Sci. **26**(3), 161–170 (1992)
9. Jaillet, P.: Stochastic routing problems. In: Andreatta, G., Mason, F., Serafini, P. (eds.) Advanced School on Stochastics in Combinatorial Optimization, pp. 192–213. World Scientific, Singapore (1987)
10. Gendreau, M., Laporte, G., Séguin, R.: Stochastic vehicle routing. Eur. J. Oper. Res. **88**, 3–12 (1996)
11. Laporte, G., Louveaux, F.V., Mercure, H.: Models and exact solutions for a class of stochastic location-routing problems. Eur. J. Oper. Res. **39**(1), 71–78 (1989)
12. Gendreau, M., Laporte, G., Séguin, R.: An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp. Sci. **29**(2), 143–155 (1995)
13. Hjorring, C., Holt, J.: New optimality cuts for a single-vehicle stochastic routing problem. Ann. Oper. Res. **86**(0), 569–584 (1999)
14. Laporte, G., Louveaux, F., Van Hamme, L.: An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. Oper. Res. **50**(3), 415–423 (2002)
15. Rei, W., Gendreau, M., Soriano, P.: Local branching cuts for the 0–1 integer L-shaped algorithm. Technical Report CIRRELT-2007-23, CIRRELT, Montréal, Canada (2007)
16. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco (2005)
17. Gendreau, M., Laporte, G., Séguin, R.: A tabu search algorithm for the vehicle routing problem with stochastic demands and customers. Oper. Res. **44**(3), 469–477 (1996)
18. Yang, W., Mathur, K., Ballou, R.H.: Stochastic vehicle routing problem with restocking. Transp. Sci. **34**(1), 99–112 (2000)

19. Chepuri, K., Homem-de-Mello, T.: Solving the vehicle routing problem with stochastic demands using the cross-entropy method. Ann. Oper. Res. **134**(1), 153–181 (2005)
20. Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T.: Hybrid metaheuristics for the vehicle routing problem with stochastic demands. J. Math. Model. Algorithms **5**(1), 91–110 (2006)
21. Secomandi, N., Margot, F.: Reoptimization approaches for the vehicle-routing problem with stochastic demands. Oper. Res. **57**(1), 214–230 (2009)
22. Rei, W., Gendreau, M., Soriano, P.: A hybrid Monte Carlo local branching algorithm for the single vehicle routing problem with stochastic demands. Transp. Sci. **44**(1), 136–146 (2010)
23. Balaprakash, P.: Estimation-based metaheuristics for stochastic combinatorial optimization: Case studies in stochastic routing problems. PhD Thesis, Université Libre de Bruxelles, Brussels, Belgium (2010)
24. Stewart Jr, W.R., Golden, B.L.: Stochastic vehicle routing: a comprehensive approach. Eur. J. Oper. Res. **14**(4), 371–385 (1983)
25. Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with stochastic demands: properties and solution frameworks. Transp. Sci. **23**(3), 166–176 (1989)
26. Dror, M.: Modeling vehicle routing with uncertain demands as a stochastic program: properties of the corresponding solution. Eur. J. Oper. Res. **64**(3), 432–441 (1993)
27. Psaraftis, H.: Dynamic vehicle routing: status and prospects. Ann. Oper. Res. **61**(1), 143–164 (1995)
28. Secomandi, N.: Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Comput. Oper. Res. **27**(11), 1201–1225 (2000)
29. Secomandi, N.: A rollout policy for the vehicle routing problem with stochastic demands. Oper. Res. **49**(5), 796–802 (2001)
30. Birattari, M., Balaprakash, P., Stützle, T., Dorigo, M.: Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study in the probabilistic traveling salesman problem. INFORMS J. Comput. **20**(4), 644–658 (2008)
31. Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M.: Adaptive sample size and importance sampling in estimation-based local search for the probabilistic traveling salesman problem. Eur. J. Oper. Res. **199**(1), 98–110 (2009)
32. Balaprakash, P., Birattari, M., Stützle, T., Yuan, Z., Dorigo, M.: Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. Swarm Intell **3**(3), 223–242 (2009)
33. Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M.: Estimation-based metaheuristics for the probabilistic traveling salesman problem. Comput. Oper. Res. **37**(11), 1939–1951 (2010)
34. Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol. 57, pp. 321–353. Kluwar Academic Publishers, Norwell (2002)
35. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California (1989)
36. Moscato, P.: Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 219–234. McGraw Hill, London (1999)
37. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
38. Dorigo, M., Birattari, M.: Swarm intelligence. Scholarpedia **2**(9), 1462 (2007)
39. Séguin, R.: Problèmes stochastiques de tournées de véhicules. PhD Thesis, Université de Montréal, Montréal, Canada (1994)
40. Bowler, N.E., Fink, T.M.A., Ball, R.C.: Characterization of the probabilistic traveling salesman problem. Phys. Rev. E **68**(3), 036703–036710 (2003)
41. Gutjahr, W.J.: A converging ACO algorithm for stochastic combinatorial optimization. In: Albrecht, A., Steinhofl, K. (eds.) Stochastic Algorithms: Foundations and Applications. LNCS, vol. 2827, pp. 10–25. Springer, Berlin (2003)
42. Gutjahr, W.J.: S-ACO: an ant based approach to combinatorial optimization under uncertainty. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M. (eds.) Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2004. LNCS, vol. 3172, pp. 238–249. Springer, Berlin (2004)
43. Tukey, J.W.: Comparing individual means in the analysis of variance. Biometrics **5**(2), 99–114 (1949)
44. Rubinstein, R.Y.: Simulation and the Monte Carlo Method. Wiley, New York (1981)

45. Bentley, J.L.: Fast algorithms for geometric traveling salesman problems. ORSA J. Comput. **4**(4), 387–411 (1992)
46. Bianchi, L., Knowles, J., Bowler, N.: Local search for the probabilistic traveling salesman problem: correction to the 2-p-opt and 1-shift algorithms. Eur. J. Oper. Res. **162**, 206–219 (2005)
47. Bianchi, L., Campbell, A.: Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. Eur. J. Oper. Res. **176**(1), 131–144 (2007)
48. Merz, P., Freisleben, B.: Memetic algorithms for the traveling salesman problem. Complex Syst. **13**(4), 297–345 (2001)
49. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
50. Johnson, D.S., McGeoch, L.A., Rego, C., Glover, F.: 8th DIMACS implementation challenge (2001)
51. Stützle, T.: ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem (2002)
52. Penky, J.F., Miller, D.L.: A staged primal-dual algorithm for finding a minimum cost perfect two-matching in an undirected graph. ORSA J. Comput. **6**(1), 68–81 (1994)
53. Johnson, D.S., McGeoch, L.A.: The travelling salesman problem: a case study in local optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. Wiley, Chichester (1997)
54. Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M.: Estimation-based metaheuristics for the the vehicle routing problem with stochastic demands and customers. IRIDIA Supplementary page (2011)
55. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-Race algorithm: sampling design and iterative refinement. In: Bartz-Beielstein, T., Blesa, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) Hybrid Metaheuristics. LNCS, vol. 4771, pp. 113–127. Springer, Berlin (2007)
56. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: an overview. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Empirical Methods for the Analysis of Optimization Algorithms. Springer, New York (2010)