J Math Model Algor (2009) 8:387–407
DOI 10.1007/s10852-009-9113-5

A Practical Approach for Robust and Flexible Vehicle Routing Using Metaheuristics and Monte Carlo Sampling

Kenneth Sörensen · Marc Sevaux

Received: 27 February 2009 / Accepted: 28 May 2009 / Published online: 18 June 2009

Abstract In this paper, we investigate how robust and flexible solutions of a number of stochastic variants of the capacitated vehicle routing problem can be obtained. To this end, we develop and discuss a method that combines a sampling based approach to estimate the robustness or flexibility of a solution with a metaheuristic optimization technique. This combination allows us to solve larger problems with more complex stochastic structures than traditional methods based on stochastic programming. It is also more flexible in the sense that adaptation of the approach to more complex problems can be easily done. We explicitly recognize the fact that the decision maker's risk preference should be taken into account when choosing a robust or flexible solution and show how this can be done using our approach.

Keywords Stochastic vehicle routing · Robustness · Flexibility · Monte Carlo sampling · Metaheuristics · Memetic algorithm

1 Robust and Flexible Vehicle Routing

The objective of vehicle routing problems is to determine the order in which to visit a spatially distributed set of customers using a set of vehicles so that the total travel cost (distance, time) is minimized. Standard vehicle routing formulations however, assume that all data concerning customer demand, travel costs, etc. are known with perfect certainty at design time. For many reasons, e.g. the uncertainty related to traffic conditions, these assumptions are unwarranted in a large number

K. Sörensen
Faculty of Applied Economics, Universiteit Antwerpen, Prinsstraat 13, 2000 Antwerp, Belgium
e-mail: kenneth.sorensen@ua.ac.be

M. Sevaux (✉)
Lab-STICC - UEB, Université de Bretagne-Sud, Centre de Recherche,
BP 92116, F-S6321 Lorient, France
e-mail: marc.sevaux@univ-ubs.fr

of practical situations. In such cases *stochastic* vehicle routing formulations are more appropriate.

For stochastic problems in general, *robust* or *flexible* solutions are required. Consistent with the definitions found in the literature (see e.g., Kouvelis and Yu [25], Mulvey et al. [29], Roy [34], Vincke [45], Dias and Clímaco [9], Branke [6]), we call a solution to a stochastic optimization problem *robust* if it has a high quality regardless of the actual realization of the stochastic parameters. In other words, although in almost all cases the risk of a poor realization cannot be completely excluded, a solution that consistently has a high performance across most possible outcomes of the stochastic parameters, is called robust. Usually, stochastic optimization algorithms attempt to find the solution that has either the best *expected* quality or the best *worst-case* quality over all potential outcomes of the stochastic parameters. Many situations exist in which a solution can be (at least partially) changed once the actual values of the stochastic parameters become known. In this case, a solution is preferred that can be successfully adapted to most realizations of the stochastic parameters, even though it is usually impossible to exclude the risk of a realization to which the solution cannot be adapted. Such a solution is called *flexible*. In other words, a *flexible* solution is one that has a high quality after adaptation to the outcomes of the stochastic parameters, whereas a *robust* solution is one that has a high quality without adaptation to the stochastic parameters. The procedure that is used to adapt the solution to the actual values of the stochastic parameters is called a *recourse* procedure. An important remark is that the definition of a robust or flexible solution is highly dependent on the specific situation and the decision maker involved. It is our firm belief that methods for stochastic optimization should offer considerable freedom to the decision maker to determine his/her own method of measuring robustness or flexibility, and to determine the amount of risk he/she is willing to take.

Several problems arise with traditional stochastic programming methods, that inhibit their use in real-life vehicle routing applications. First, most methods for stochastic optimization are unable to solve problems of realistic size. Indeed, as indicated by Birge [5], the complexity of stochastic programs grows proportionally to the number of possible realisations of the stochastic parameters, which in turn grows exponentially with the number of stochastic parameters. As a result, only very small problems can be solved to “optimality”. A second drawback to traditional stochastic programming methods is the fact that they make heavy use of the specific stochastic structure of the problem they consider and are very difficult to adapt to other problems. As a result, commercial packages for vehicle routing, although widely used in practice, can usually not accommodate for stochastic parameters. Third, most algorithms developed for stochastic routing attempt to find the solution with the optimal *expected value*. Although the average performance of a solution is certainly an important indication of its robustness, it does not take into account the risk preference of the decision maker. Some approaches to incorporate the risk preference in the vehicle routing formulation exist, see e.g. the chance-constrained stochastic problems discussed in Dror [10], Dror et al. [11]. The value of correctly addressing uncertainty and risk in operational planning has been determined by Reimann [33] in the context of a vehicle routing problem with stochastic demand.

To partially overcome the curse of dimensionality in stochastic optimization, a part of the stochastic programming literature has fused on methods that use some form of *Monte Carlo sampling* of the stochastic parameters, an idea that can be traced back to Jagannathan [21]. For stochastic linear programming problems, *stochastic decomposition* [17] and *importance sampling* [20] have been developed. For discrete stochastic problems, Norkin et al. [30, 31] develop a sampling branch-and-bound. Whereas these methods use sampling at different steps of the optimization method to estimate e.g., function values, the *sample average approximation method* [24] uses sampling to generate a set of sample scenarios and then attempts to optimize the corresponding deterministic expected-value problem. This method has been successfully applied to supply chain design problems [14] and routing problems [44], among others.

Specifically in vehicle routing, a number of problem formulations designed to find robust or flexible solutions have been developed. Three types of such formulations, commonly referred to as *stochastic vehicle routing problems* can be distinguished: problems with stochastic *travel times* (e.g., Lambert et al. [26]), problems with stochastic *demands* (e.g., Stewart and Golden [42]), and problems with stochastic *customers*, i.e., in which each customer has a certain probability of requiring service (e.g., Bertsimas [3]). For more details on these different problem formulations, we refer to Gendreau et al. [13].

The best-studied stochastic vehicle routing problem is the vehicle routing problem with stochastic demand (VRPSD). The main issue in this problem is that the vehicle may have to visit the depot at previously unforeseeable moments for restocking. A question that arises naturally is to which extent re-optimization of the routes as new information becomes available is beneficial. Haughton [15] and Haughton and Stenger [16] study different strategies for route modification and estimate the benefits of using re-optimization. A re-optimization approach is developed by Secomandi [35–37], who uses various Markov decision process formulations of the problem, and presents a rollout policy heuristic for two of them. A dynamic re-optimization approach to solve a stochastic VRP with both stochastic customer locations and demands can be found in Hvattum et al. [19]. An alternative to re-optimization is a-priori optimization, in which the recourse action consists of simply returning to the depot for restocking if demand at a customer cannot be fully delivered. Gendreau et al. [12] and Laporte et al. [27] describe exact algorithms for this problem.

Recently, a rather large body of literature has emerged on sampling-based methods in vehicle routing. Kenyon and Morton [23], for example, develop a branch-and-cut framework that includes a sampling-based estimation of the objective function value. Some attention in the literature has been given to determine the sample size in Monte Carlo sampling. Homem-De-Mello [18] studies variable sampling schemes for discrete stochastic optimization problems. In these schemes, the sampling size is adaptively decided at each iteration. In [7], the authors apply such a scheme to the VRPSD.

In this paper, we adopt a pragmatic approach to stochastic optimization and approximate the stochastically optimal solution on two levels. First, we use a sampling-based approach to quickly estimate the robustness of a solution. This allows us to quickly and simultaneously evaluate both average and worst-case performance of a solution, but more complicated measures of robustness can be incorporated. Our sampling-based approach also allows us to evaluate solutions of problems with many stochastic parameters. Second, we use a metaheuristic to find the solution with the best approximate robustness characteristic, which allows us to solve large problems. Our approach adapts a metaheuristic for a deterministic problem by replacing its evaluation function by a so-called *robustness/flexibility evaluation function*. An application of this method to the capacitated facility location problem can be found in Sörensen [40]. In Sörensen [39], it was shown how the method can be used to increase the stability of vehicle routing solutions over time, an important issue in practice but often overlooked in the literature. For a general survey of the use of metaheuristics in stochastic optimization, we refer to Bianci et al. [4]. The approach in this paper differs from approaches generally found in the literature in that it explicitly recognizes the need for the decision maker to incorporate a measure of risk in the objective function, and allows to focus on a more complicated—and possibly multi-objective—measure of robustness than the expected value. Moreover, the flexibility of the approach also allows us to solve problems with different stochastic structures (e.g. with or without second-stage decisions) by making minimal changes to the algorithm. The fact that the MA|PM by design generates diverse solutions helps the algorithm to track down those solutions that are most robust and/or flexible.

The rest of this paper is organized as follows. In Section 2, we describe the deterministic capacitated vehicle routing problem (CVRP) and develop a memetic algorithm with population management to solve it. In Section 3, we show how this algorithm can be adapted so that it searches for robust and/or flexible solutions. Rather than focus on a specific stochastic vehicle routing problem, we use this method in Section 4 to solve a variety of different stochastic variants of the CVRP. Section 5 discusses the computing times required by our method and the choice of the number of samples. Finally, some conclusions and pointers for future research can be found in Section 6.

2 Problem Description and Deterministic Solution Method

2.1 The Capacitated Vehicle Routing Problem

The CVRP is defined on an undirected graph $G = (V, E)$ with a set of nodes $V = \{0, 1, \ldots, n\}$. Node 0 corresponds to the depot, where a set of identical vehicles are located, that all have capacity Q. Each vehicle may execute a route that has a maximal travel cost of C. Nodes 1 to n represent a set of spatially distributed customers, the demand of which is given by q_i. The travel cost between customer i and customer j is given by c_{ij}, the weight of the edge between node i and node j. The objective of the deterministic VRP is to find a set of minimum total cost routes that have the following properties: (1) each route begins and ends at the depot, (2) each customer is visited exactly once, (3) the capacity and maximal travel cost per route of the vehicles is not exceeded. For a mathematical formulation of this problem, we refer to Toth and Vigo [43].

2.2 A Memetic Algorithm With Population Management for the CVRP

To solve this problem, we have developed a MA|PM, or menetic algorithm with population management. A MA|PM is a memetic algorithm (a GA hybridised with local search) that uses distances to measure and control the diversity of a small population. This allows to maintain diversity in the population while keeping the size of the population small. For a more elaborate description of MA|PM, we refer to Sörensen and Sevaux [41].

A memetic algorithm with population management is structured much like a standard memetic algorithm, but differs in the use of population management. An outline is given in Algorithm 1.

In Algorithm 1, $d_P(o)$ is the distance of solution o to the population P. In Section 2.2.2, we discuss how this distance can be calculated. Δ is the so-called population diversity parameter, i.e. the minimum distance to the population that a solution should have before it can be considered for insertion into the population. This parameter can be used to control the diversity level of the population. b is a solution from the population that is chosen for removal by a selection operator (see Section 2.2.5).

2.2.1 Solution Representation

In our MA|PM, a solution S is represented as a vector of customers, *without trip delimiters* $S = (S_1, S_2, \ldots, S_n)$ with $S_i \in \{1, \ldots, n\}$ and $S_i \neq S_j$ if $i \neq j$. When necessary (e.g., to calculate their objective function value) they are decoded optimally using the *split decoding procedure* by Prins [32]. This procedure finds the optimal points at which to split the solution into tours (i.e. schedule visits to the depot) so that the resulting decoded solution is feasible and the total travel cost is minimized. It makes use of an auxiliary directed weighted graph $H = (X, F, W)$. The vertex set X contains $n + 1$ nodes (where n is the number of customers served), labelled from 0 (the depot) to n.

Algorithm 1: MA|PM outline

```
initialize population $P$;
set population diversity parameter $\Delta$;
repeat
    select: $p_1$ and $p_2$ from $P$;
    crossover: $p_1 \otimes p_2 \rightarrow o_1, o_2$;
    local search: improve $o_1$ and $o_2$;
    for each offspring $o$ do
        while $d_P(o) < \Delta$ do
            mutate $o$;
        remove solution: $P \leftarrow P \backslash \{b\}$;
        add solution: $P \leftarrow P \cup \{o\}$;
    update diversity parameter $\Delta$;
until stopping criterion satisfied ;
```

The edge set F is constructed as follows. An edge (i, j) from vertex i to vertex j $(i < j)$ is added when a trip containing customers S_{i+1} to S_j is feasible, i.e., if

$$\sum_{k=i+1}^{j} q_{S_k} \leq Q \tag{1}$$

and

$$w_{ij} = c_{0S_{i+1}} + \sum_{k=i+1}^{j-1} c_{S_k S_{k+1}} + c_{S_j 0} + \sum_{k=i+1}^{j} e_{S_k} \leq C. \tag{2}$$

The weight w_{ij} of the arc (i, j) is equal to the sum of the travel costs covered in the trip $(0, S_{i+1}, S_{i+2}, \ldots, S_j, 0)$. The minimum-cost path from vertex 0 to vertex n in H gives the shortest set of trips corresponding to this encoding. If more than one min-cost path exists, each of these paths will give a set of trips with equal total length. Since the graph H contains only positive weights and no circuits, the shortest path can be efficiently calculated using Bellman's algorithm [2].

2.2.2 A Distance Measure for CVRP Solutions

A second procedure used heavily by our MA|PM for the CVRP is a distance measure based on the edit distance. Given three possible edit operations (add character, remove character and substitute character), the edit distance $d(s, t)$ is defined as the minimal number of edit operations required to transform a string s into another string t. The edit distance is heavily studied in the literature. A simple dynamic programming algorithm [46] calculates this measure in $O(l^2)$ where l is the length of both strings. Other, more efficient algorithms have been developed, but these are beyond the scope of this paper.

Vehicle routing solutions can be regarded as sets of strings, each string representing a tour. As tours have no direction, each string may be reversed. The distance measure for the CVRP developed in Sörensen [38] calculates the minimal number of edit operations required to transform a VRP solution into another one, keeping into account that both solutions consist of a set of reversable strings. This is done by assigning tours from the first solution to the second one so that the total number of required edit operations is minimized.

Given two solutions s and t having m and n trips respectively, the distance measure constructs a square matrix M of size $\max(m, n)$. Each column and each row represents a trip of s and t respectively. Empty trips are added to the solution with the smallest number of trips to give both solutions an equal number of trips. Cell (i, j) of M contains the minimum of $d(s_i, t_j)$ and $d(s_i, \tilde{t}_j)$, where s_i is the i-th trip of solution s and $\tilde{t}_j$ is the reversed j-th trip of solution t. A linear assignment problem is then calculated to find a minimum-cost assignment of trips in solution s to trips in solution t. The cost of the assignment problem is the distance between the two solutions.

Consider the example in Fig. 1, in which we find the distance between two solutions to a vehicle routing problem with 10 customers, labeled a to i.

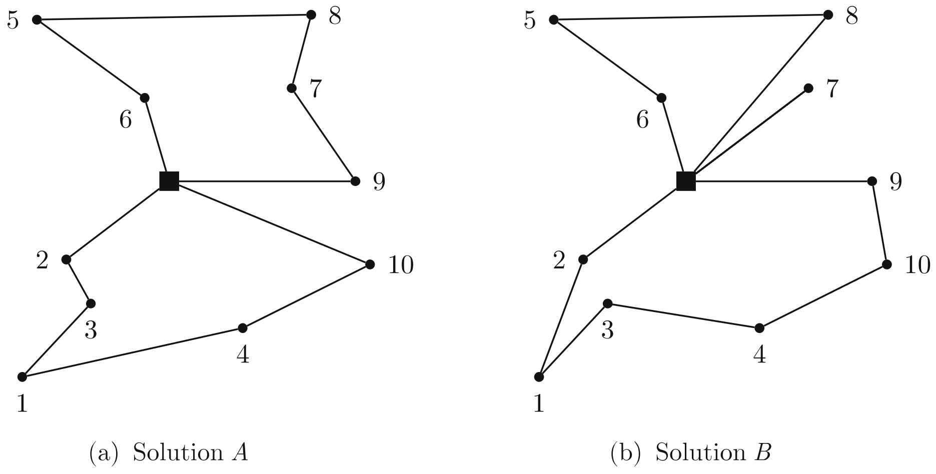

Fig. 1 Two solutions of a vehicle routing problem (**a**, **b**)

The solutions in Figs. 1a and b can be encoded, for example, in the following ways: $A = (0, 2, 3, 1, 4, 10, 0, 6, 5, 8, 7, 9, 0)$ and $B = (0, 8, 5, 6, 0, 7, 0, 9, 10, 4, 3, 1, 2, 0)$. Note that the order in which the trips are listed in the encoding, as well as the direction of the trips in the encoding, may be changed. For example, $(0, 9, 7, 8, 5, 6, 0, 2, 3, 1, 4, 10, 0)$ is a valid encoding of solution A. Also note that the solutions do not contain the same amount of trips. The distance measure developed in this section takes all this into account.

To calculate the distance between solutions A and B, we calculate the distance between each trip in A and each trip in B, adding empty trips to the solution with the smallest number of trips so that both solutions have the same amount of trips (Table 1a, empty trips are indicated by Λ). We also calculate the distance between each trip in A and the reverse of each trip in B (Table 1b). The minimum of corresponding cells in both matrices is shown in matrix M, Table 1c. In all tables, the depot has been omitted. The solution of the linear assignment problem with M as cost matrix is indicated in bold. The distance between the two solutions is equal to the total cost of the assignment, i.e. 6.

In the MA|PM, the distance measure is used for *population management* in order to maintain the diversity of a small population of high-quality individuals and overcome problems of premature convergence. Before a (candidate) solution is added to the population, the algorithm checks to see whether it satisfies the *population diversity criterion*, i.e., whether it is sufficiently different from all other members of the population. This is done by measuring the so-called *distance to the population*, i.e., the minimum distance of the candidate solution o to any solution already in the population. In symbols, $d_P(o) = \min_{x \in P} d(o, x)$. The solution cannot be added until its distance to the population is greater than or equal to the value of the *population diversity parameter* Δ. This value can be used to control the diversity of the population as an increase of Δ will tend to increase population diversity, while a small value will tend to decrease it. Intensification and diversification phases can be alternated by setting the value of Δ to appropriate levels [41].

Table 1 Distances between trips for calculation of the distance between the VRP solutions

(a)

	(2, 3, 1, 4, 10)	(9, 7, 8, 5, 6)	Λ
(2, 1, 3, 4, 10, 9)	3	6	6
7	5	4	1
(6, 5, 8)	5	4	3

(b)

	(10, 4, 1, 3, 2)	(6, 5, 8, 7, 9)	Λ
(2, 1, 3, 4, 10, 9)	5	6	6
7	5	5	1
(6, 5, 8)	5	2	3

(c) Matrix M

min	(2, 3, 1, 4, 10)	(9, 7, 8, 5, 6)	Λ
	(10, 4, 1, 3, 2)	(6, 5, 8, 7, 9)	Λ
(2, 1, 3, 4, 10, 9)	**3**	6	6
7	5	4	**1**
(6, 5, 8)	5	**2**	3

2.2.3 Local Search Optimization

The MA|PM also uses a fast local optimization algorithm, that consists of two simple tabu search procedures that are used alternatively until neither of them finds any more improvements. The *insert* tabu search procedure attempts to locate any customer at any other potential location in the solution (i.e., any other location in any other tour) and takes the move that decreases the total cost most (or increases it least when no improving moves can be found). An insertion of a customer into another location is only allowed when no constraints are violated by this move and when the customer does not appear on the tabu list. This insert procedure is repeated until the best-found solution during this tabu search phase has not been improved for a fixed number of moves. A fixed-length tabu list is maintained to avoid cycling. The tabu list contains the customers that were most recently moved. The *swap* tabu search procedure attempts to swap every pair of customers and swaps the pair that yields the largest improvement in objective function value. The swap of a pair of customers is only allowed when the resulting solution does not violate any constraints and when the pair of customers does not appear on the tabu list. This procedure is repeated until a fixed number of non-improving moves. In this case, the tabu list contains pairs of items that were most recently swapped. Pairs that appear on the tabu list are excluded from swapping.

2.2.4 Crossover

The crossover operator used in our MA|PM is the *linear ordered crossover* (LOX). This operator starts by randomly dividing both parents into three parts. The first offspring solution is found by copying the middle part from parent one and completing the solution by circularly scanning parent two, starting from the third part and wrapping to the first part. The second offspring is formed by reversing the roles of the parents. An example is shown in Fig. 2. The bold letters in this figure indicate which customers end up in the first offspring solution. The arrows indicate the order

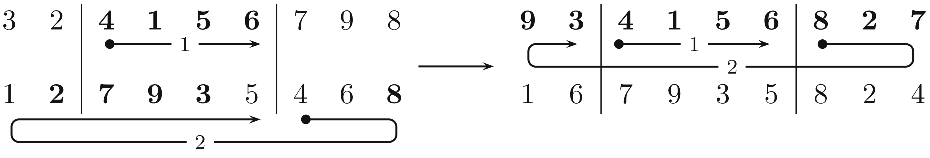

Fig. 2 Linear ordered crossover (LOX)

in which the two parents are scanned to produce the first offspring and how this offspring is built. For the second offspring, the roles of the parents are reversed.

2.2.5 Overview and Adaptive Population Management

To build an initial population, our MA|PM for the CVRP starts by generating a set of random permutations of all customers. These are then decoded (using the split procedure) and subjected to local search (alternating the two tabu search operators until no more improvement is found). The initial population consists of a small number of solutions, usually 10. At each generation, two solutions are drawn from the population using a binary tournament selection operator (i.e., we retain the best of two randomly chosen solutions). These two solutions are subjected to the LOX operator, to generate two offspring solutions. Both offspring solutions are then decoded and subjected to the tabu search local searches. When an improved offspring solution satisfies the diversity criterion, it is added to the population. If not, it is *mutated* until it does satisfy the diversity criterion and then added to the population. The mutation operator randomly swaps two customers and repeats this step until the resulting solution satisfies the diversity criterion. The resulting solution is then added to the population, replacing a solution chosen by reverse binary tournament selection (i.e. the worst of two randomly chosen solutions is removed).

We use an *adaptive* population management scheme, in which the value of Δ is slowly decreased to allow the search to intensify and increased again when the search appears to be stuck in a local optimum, i.e., no more improving solutions are found for a fixed number of generations. The increase of Δ forces the search into yet unexplored regions of the search space, by increasing the distance between solutions in the population.

Some experiments show that our MA|PM is competitive with the best-known approaches in the literature. For detailed results, we refer to Sörensen [38].

3 A Sampling-based Approach for Robust and Flexible Vehicle Routing

In this section, we modify the MA|PM developed in the previous section, so that it is able to deal with stochastic vehicle routing problems. This is done by replacing the objective function by a *robustness* or *flexibility evaluation function*, that measures the robustness or flexibility of the solution respectively. A *recourse function* is used if the solution can be adapted when the actual values of the stochastic parameters become known.

In a nutshell, our approach is the following. We use a metaheuristic, such as our MA|PM, that generates a set of solutions that are both diverse and have a high quality. For each solution we encounter, we calculate one or more measures of robustness or flexibility, referred to as a robustness/flexibility evaluation function value. Finally, we choose the solution that has the best robustness/flexibility evaluation function value. When two or more measures of robustness or flexibility are calculated, we choose the solution that has the best combined value using a multi-objective decision making process.

A robustness evaluation function value for a given solution is calculated by repeatedly applying the solution to a sampling of the stochastic parameters and calculating the corresponding (deterministic) objective function value. These objective function values are then combined into one or more measures of robustness. If a recourse procedure is available, the solution is repaired before it is evaluated, and the resulting objective function is called a flexibility evaluation function. The recourse procedure may be exact or heuristic in nature.

A typical robustness evaluation function f^* is of the form

$$f^*(x) = \frac{1}{n_e}\sum_{i=1}^{n_e} f(x, \mathcal{S}_i(\pi)), \tag{3}$$

where f is the (deterministic) objective function, π is the set of stochastic parameters, and $\mathcal{S}$ is the sampling function. $\mathcal{S}_i(\pi)$ represents the i-th sampling of the stochastic parameters and $f(x, \mathcal{S}_i(\pi))$ is the objective function value of solution x when applied to the i-th sampling. n_e is the number of deterministic evaluations per robustness evaluation, i.e., the number of evaluations of the solution on a random sampling of the stochastic parameters. If the solution can be repaired, the recourse function $\mathcal{R}$ is invoked before evaluation the solution and the flexibility evaluation function becomes

$$f^*(x) = \frac{1}{n_e}\sum_{i=1}^{n_e} f(\mathcal{R}(x, \mathcal{S}_i(\pi))). \tag{4}$$

Other useful robustness/flexibility evaluation functions include the *worst-case* performance, given (for a minimization problem) by

$$f^*_{\text{wc}}(x) = \max_i f(x, \mathcal{S}_i(\pi)), \tag{5}$$

or the standard deviation

$$\sigma^*(x) = \sqrt{\frac{1}{n_e - 1}\sum_{i=1}^{n_e}\left[f\left(x; \mathcal{S}_i(\pi)\right) - f^*(x)\right]^2}. \tag{6}$$

These measures explicitly incorporate the risk preference of the decision maker into the process. E.g. the solution that has the best worst-case performance will generally be more "conservative" or "risk-averse" than the one that has the best average-case performance. Although in some specific cases, the worst-case performance may be easy to determine analytically (such as in the CVRP with stochastic travel cost, in which the worst-case scenario is when all travel costs are at their maximum value), the sampling-based approximation may be preferred for a number of reasons. First, this approximation may be a more realistic estimate of a performance that may be found in practice. This is especially true in situations in which the theoretical worst-case scenario is extremely unlikely to occur. A decision maker that wants a solution that hedges against the “worst-case scenario” generally means a scenario that has a greater than infinitesimal probability of occurring. A scenario which corresponds to the worst of a sample of 100 random scenarios may be closer to what the decision maker means by “worst-case” than a theoretical worst-case scenario that has almost zero probability to occur. Second, it is not always easy or even tractable to analytically determine the worst-case performance of a given solution. In vehicle routing with stochastic demand, this might be the case when penalties for exceeding demand in a route may make the solution worse than not having any demand in a route at all.

Artzner et al. [1] argue that measures of risk should be *coherent*, and define the four properties that a coherent risk measure should possess (translation invariance, subadditivity, positive homogeneity, and monotonicity). Although many arguments can be given to prefer such risk measures, and although the above mentioned risk measures are not coherent by this definition, our method does not depend on the risk measures used and so we prefer to stick to the (simpler and better-known) risk measures above.

The decision maker may decide to find the solution that minimizes $f^*(x) + \lambda\sigma^*(x)$, where λ is a parameter expressing the risk-averseness of the decision maker. Alternatively, he/she may choose to generate a set of non-dominated solutions using two or more measures of robustness with varying degrees of risk averseness and resort to a multi-criteria decision making method to choose between them. Other, even more complex measures of robustness may be used, such as the probability that the quality of the solution falls below a certain threshold. Of course, since they are obtained using Monte Carlo simulation, the robustness/flexibility evaluation function values are only estimates of the true robustness measures. However, statistical theory can be used to estimate the reliability of the estimates. For a sufficiently large number of evaluations (e.g., $n_e > 30$), the central limit theorem states that $f^*(x)$ is approximately normally distributed. From this, it follows that an $100(1-\alpha)$ confidence interval for the real average performance of a solution is given by

$$f^*(x) \pm z_{1-\frac{\alpha}{2}}\sqrt{\frac{(\sigma^*(x))^2}{n_e}}. \tag{7}$$

For a more elaborate discussion, we refer to Law and Kelton [28].

One possible extension of our method is the use of *penalty functions* to penalize violation of constraints. For some samples of the stochastic parameters, a solution might become infeasible. A penalty function $\mathcal{P}$ may be used to express the severity of the constraint violations and added to the robustness evaluation function value as follows:

$$f^*(x) = \frac{1}{n_e}\sum_{i=1}^{n_e}\left[f(x;\mathcal{S}_i(\pi)) + \mathcal{P}(x;\mathcal{S}_i(\pi))\right]. \tag{8}$$

In our method, stochastic information can be expressed both as independent probability distributions on the parameters of the problem, or as a set of scenarios in which each of the parameters has a fixed value. In the latter case, the value of n_e is the number of scenarios, and the sampling function $\mathcal{S}$ is used to evaluate the performance of a solution under the different scenarios.

Determining the number of evaluations to perform at each robustness evaluation is a decision that needs to be taken with some care, because it has an impact on the computational effort needed and determines to a large extent the precision of the robustness evaluation function. In Section 5.2, this issue is discussed in more detail.

We believe that our approach has several advantages over traditional methods based on stochastic programming. As we will show in the experiments section, our method is considerably more flexible, considerably easier to implement and applicable to much larger and much more complex problems. Moreover, it allows the decision maker to evaluate complex robustness characteristics of a solution, using any type of stochastic information that is available and incorporating his risk preference.

4 Experiments

In this section, we perform some experiments on different stochastic versions of the CVRP. Unless otherwise indicated, our MA|PM uses a population of size 10. The population diversity parameter Δ is set to 20 initially and reduced by one unit for each 3 generations. Δ is set to its initial level after 60 unproductive generations. Both the *swap tabu search* and the *move tabu search* procedures use a tabu tenure of 30 and repeat until 10 non-improving moves have been observed.

Data sets for the stochastic versions are derived from the Christofides et al. [8] instances for the deterministic CVRP, available from the OR Library.[1] We generally assume that the distributions of the stochastic parameters are independent and have their mean equal to the corresponding value in the deterministic data set. Unless otherwise indicated, the robustness/flexibility evaluation function value is based on $n_e = 1000$ evaluations.

4.1 Vehicle Routing with Stochastic Demand and Cost

In this set of experiments, customer demand q_i is assumed to be uniformly distributed between $0.75\bar{q}_i$ and $1.25\bar{q}_i$. Travel cost between customers i and j are uniformly distributed between $0.8\bar{c}_{ij}$ and $1.2\bar{c}_{ij}$. $\bar{q}_i$ and $\bar{c}_{ij}$ are the averages of the demand and travel cost distributions respectively and equal to the values found in the deterministic data set. Note that the deterministic problem is the *mean value formulation* of the stochastic problem, i.e., the deterministic problem can be obtained by setting all stochastic parameters to their expected value.

We assume that the routes have to be determined before the actual values of customers demand and travel cost are known and cannot be changed afterwards. We therefore attempt to find *robust* solutions. For some samples of the stochastic

[1] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/vrpinfo.html

demand and cost, a solution might violate the maximum travel cost constraints per route or the capacity constraints of the vehicles. We assume that violation of the maximum travel cost constraints and the capacity constraints results in (large) penalty cost due to overtime costs payable to the drivers or loss of goodwill by the customers whose demand the vehicle is not able to satisfy. We therefore introduce two penalty functions. Let c_{ijk} and q_{ik} represent the k-th random numbers taken from the distributions of c_{ij} and q_i respectively. The deterministic objective function value of solution x, evaluated on the k-th sampling of the stochastic parameters can be calculated as follows.

$$f(x; \mathcal{S}_k(\pi)) = \sum_{l} \sum_{e_{ij} \in E_l(x)} c_{ijk}, \tag{9}$$

where $E_l(x)$ represents the set of edges included in the l-th tour of solution x, e_{ij} the edge between customers i and j. If the total cost in a given route is larger than C (the maximum cost), then a fixed penalty per unit of exceeded cost is added.

$$\mathcal{P}_{\text{cost}}\,(x; \mathcal{S}_k(\pi)) = \alpha_1 \sum_{l} \max\left(0, \sum_{e_{ij} \in E_l(x)} c_{ijk} - C\right). \tag{10}$$

Similarly, if the total demand served in a given route exceeds the maximum demand Q, a fixed penalty cost per unit of exceeded demand is added.

$$\mathcal{P}_{\text{capacity}}\,(x; \mathcal{S}_k(\pi)) = \alpha_2 \sum_{l} \max\left(0, \sum_{v_i \in V_l(x)} q_{ik} - Q\right), \tag{11}$$

where $V_l(x)$ represents the set of vertices (customers) in tour l of solution x. α_1 is the penalty cost per unit of exceeded travel cost. α_2 is the penalty per unit of exceeded capacity. We set $\alpha_1 = 100$ and $\alpha_2 = 500$. These penalties are set to relatively high values to clearly distinguish between solutions that are robust and solutions that are not.

We look for solutions that have both a good average-case and a good worst-case performance. To this end, we define two robustness evaluation functions: $f^*(x)$ and f^*_{wc}. To calculate f^* for solution x, we calculate the total travel cost increased with the values of both penalty functions for n_e random samples of the stochastic parameters and take the average. For f^*_{wc}, we take the maximum value. For each solution encountered, we also measure the standard deviation $\sigma^*(x)$ and the value $f(x)$, the deterministic objective function value.

The MA|PM is applied to the 14 vehicle routing problems. Robustness evaluation function f^* is used in the binary tournament selection procedure to select solutions from the population for crossover. This guides the search towards more robust solutions. We stop the MA|PM after 200 generations, yielding 400 solutions. Detailed results are in Appendix (Table 4). Table 2 is an extract from this table holding the results of a single experiment.

For each data set, three rows are presented. The first row contains the characteristics of the solution that minimizes f, i.e., the solution that would be found by our method in the deterministic case, if we would ignore all stochastic information. Note

Table 2 Vehicle routing with stochastic demand and cost, example result

Data file	n	Criterion	f	f^*	σ^*	f^*_{wc}
vrpnc01	50	f	549.76	3211.60	2891.00	17779.56
		f^*	604.76	605.01	13.21	876.66
		f^*_{wc}	629.12	629.28	0.61	661.48

that the solutions found in this way are not up to par with those found by the best-known methods in the literature. This is not due to a bad performance of our method, but rather to the fact that the method does not explicitly search for solutions that have a good deterministic objective function value. Rather, the solutions reported in this row are the best ones encountered while searching for robust solutions. The second and third rows contain the characteristics of the solution that minimizes f^* and f^*_{wc} respectively. It can be seen that the solutions in the second and third rows score considerably better with respect to all robustness indicators (f^*, f^*_{wc} and σ^*). These two solutions are therefore considerably more robust than the solution in the first row, proving that there is a strong need to take the stochastic information into account.

This is further illustrated in Fig. 3, where we plot the values of f versus f^* for all 400 solutions. From this figure, it is clear that all robust solutions (low f^*) have a good deterministic objective function value, but that the reverse is not necessarily true. The decision maker can choose one of the two robust solutions presented in Table 4 or alternatively one of the other non-dominated solutions (solutions for which there does not exist a solution which scores better with respect to all robustness evaluation functions) generated by the MA|PM. These are not shown because of space restrictions.

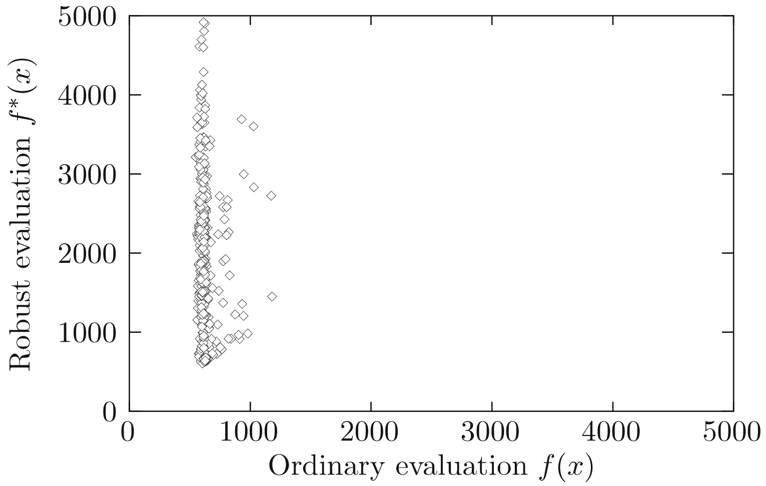

Fig. 3 Ordinary evaluation and robustness evaluation for vrpnc01

4.2 Vehicle Routing with Stochastic Customers and a Recourse Function

In some situations, the vehicle dispatcher does not know in advance which customers will require service and which will not. A possible strategy is to route all customers and repair the solution once the list of customers that do require service becomes known. This situation may occur when customers need to book a service in advance, but may cancel at short notice. A solution that can be effectively repaired is called *flexible*.

In an experiment, each customer has a 50% probability of requiring service. A 100% service level is required and the company therefore decides to design conservative solutions that are still feasible if all customers in a route require service. The recourse procedure used by the company is to remove all customers that do not require service from a given route. If the customer in i-th position in a given route is removed, the vehicle travels directly from the customer in position $i-1$ to the customer in position $i+1$. Total travel cost is measured after the customers have been removed from the routes.

The MA|PM is run for 200 generations. To determine the solution that has the highest flexibility, each of the 400 generated solutions is evaluated 1000 times on a random list of customers. As in the previous experiment, we record the average f^*, the maximum f^*_{wc} and the standard deviation σ^* over all n_e evaluations. Results can be found in Table 5.

From this table, it is clear that not much profit can be gained by using the robust optimization approach, as the solution that has the best value of f has approximately the same flexibility characteristics as the solution that optimizes one of the flexibility evaluation function values. The main reason for this is the requirement that the solution should be feasible even when all customers are present. This results in very conservative solutions that—when the actual customers to serve are observed—use only a fraction of the available resources (capacity and maximum cost). The best solution of the deterministic problem, utilising as much of these resources as possible, is therefore likely to be a better solution for the reduced set of customers, than other solutions that do not use the resources as fully. As a result, this solution is very likely to be flexible.

5 Use of the Robustness/Flexibility Evaluation Function and Computation Times

5.1 Guiding the Search

The use of our approach for robust and flexible vehicle routing requires some design issues to be tackled. Especially the question when to use the robustness/flexibility evaluation function (from now on referred to as robustness evaluation function), should be taken with some care. Obviously, the robustness evaluation function is used to evaluate the robustness/flexibility properties of a solution. However, it can also be used to guide the search towards more robust solutions. In our MA|PM, the robustness evaluation function is used to guide the search in the binary tournament selection process. However, it is not used for the move selection of the tabu search local optimization procedures. In both tabu search procedures, the (deterministic) objective function value is used to select the next move. The reason for this is that the use of the robustness evaluation function for move selection in the tabu search procedures would have resulted in prohibitive computing time increases. The bulk of computing time is spent on improving solutions using the tabu search procedures. While this is necessary to obtain high quality solutions, the tabu search procedures require a large number of evaluations. Moreover, the calculation of the cost of a move can easily be short-circuited when a deterministic objective function is used, but this no longer holds when a robustness evaluation function is used. Indeed, it can be easily seen that when a customer k is inserted between customers i and j, the total cost of this route changes by $c_{ik} + c_{kj} - c_{ij}$ (provided that the route remains feasible). Similar reasoning applies for removing a customer from a route or swapping two customers. These shortcut calculations allow a move to be evaluated without re-computing the entire solution, but they are no longer valid for a robustness evaluation function.

By using the robustness evaluation function for selection of solutions from the population, it only has to be calculated twice for each generation (when both solutions are added to the population). This results in relatively small computing time increases as can be seen in Table 3. This table compares the computing times for 200 generations for the MA|PM in the deterministic and the stochastic case (experiment with stochastic demand and cost). Computation time never increases by more than a factor of 3.08 (Computational time for each instance can be found in Tables 4 and 5).

Although the problems mentioned in this paper could be adequately solved by using the ordinary evaluation function to guide the search, situations are imaginable in which this is not the case. In such cases, the decision maker will be forced to use the robustness evaluation function to guide the search, incurring a much higher computational cost. In the experiments described here, the number of evaluations n_e could be relatively arbitrarily set to a very high level. In cases where the robustness evaluation should be used to guide the search, a much more careful decision is warranted. This is discussed in the next section.

Table 3 Computing times (200 generations)

Data file	n	avg. determ. (s)	avg. stoch. (s)	$\frac{\text{avg.stoch.}}{\text{avg.determ.}}$
vrpnc01	50	68.84	99.30	1.44
vrpnc02	75	205.59	408.59	1.99
vrpnc03	100	363.23	573.17	1.58
vrpnc04	150	932.70	1856.12	1.99
vrpnc05	199	1969.29	4266.21	2.17
vrpnc06	50	92.57	97.52	1.05
vrpnc07	75	205.43	423.80	2.06
vrpnc08	100	369.90	655.88	1.77
vrpnc09	150	967.54	2329.98	2.41
vrpnc10	199	1657.46	5102.75	3.08
vrpnc11	120	667.53	1215.21	1.82
vrpnc12	100	292.09	698.23	2.39
vrpnc13	120	772.81	1008.80	1.31
vrpnc14	100	526.89	722.35	1.37
All	N/A	649.42	1389.85	2.14

5.2 Determining the Number of Evaluations

One of the design choices that influence the computation time when using our robust optimization approach is the value of n_e, the number of objective function evaluations to perform to calculate the value of the robustness evaluation function. In the experiments performed in this paper, we put the value of n_e at 1000 and still obtained relatively small computing time increases, but this may not always be the case. Especially in situations where the objective function is time-intensive to calculate, the value of n_e must be carefully chosen.

As mentioned before, a confidence interval can be calculated around the value of f^* (see Eq. 7). The value of n_e should be chosen in such a way that the confidence interval is small enough. How small exactly the confidence interval should be, depends on the specific application, the level of confidence that is required by the decision maker and the computing time available. Determining the value of n_e can either be done *off-line*, i.e. before the optimization process starts or *on-line*, i.e. during the optimization phase. In the off-line case, n_e is fixed to a certain value, and the same value is used throughout the optimization run. We used this technique in the examples in Section 4. In the on-line case, the value of n_e is determined every time a solution is evaluated with a robustness evaluation function. In both cases, an initial number of evaluations n_0 needs to be made, after which evaluations are added until the desired precision is reached. A quick way to estimate the number of evaluations required is given by Kelton et al. [22], who propose to use the following rule of thumb:

$$n_e = n_0 \sqrt{\frac{\sigma_0}{\sigma_e}}, \tag{12}$$

where σ_0 is the standard deviation of the initial replications and σ_e is the desired standard deviation.

6 Conclusions

In this paper, we have developed an approach to find robust and flexible solutions of several stochastic versions of the capacitated vehicle routing problem. Our approach combines the optimization power of metaheuristics with Monte Carlo simulation based estimation of the robustness or flexibility of a solution. A recourse procedure is introduced when flexible solutions are needed and penalty functions may be introduced to penalize violation of constraints. Our approach recognizes the need for more complex expressions of robustness or flexibility than the often-used average performance to be entered into the decision-making process.

We have tested our approach on vehicle routing problems with stochastic demand and cost and with stochastic customers. We have also discussed some design issues, specifically how the search can be guided towards robust or flexible solutions and how the computing time increase can be kept within limits.

We believe that our approach offers considerable advantages over more traditional methods based on stochastic programming, especially when applied to large-scale, real-life stochastic vehicle routing applications. Further research will therefore focus on applying our method in real-life applications.

Appendix

Table 4 Vehicle routing with stochastic demand and costs

Data file	n	Criterion	f	f^*	σ^*	f^*_{wc}	CPU (s)
vrpnc01	50	f	549.76	3211.60	2891.00	17779.56	101
		f^*	604.76	605.01	13.21	876.66	
		f^*_{wc}	629.12	629.28	10.61	661.48	
vrpnc02	75	f	891.86	8335.36	4986.75	30836.52	349
		f^*	956.57	1134.44	617.56	6509.91	
		f^*_{wc}	956.57	1134.44	617.56	6509.91	
vrpnc03	100	f	887.97	2904.39	2753.68	16150.60	541
		f^*	971.38	977.51	72.99	2236.30	
		f^*_{wc}	971.38	977.51	72.99	2236.30	
vrpnc04	150	f	1149.53	5313.22	3869.58	23450.44	1893
		f^*	1208.36	1596.03	1063.23	9598.45	
		f^*_{wc}	1208.36	1596.03	1063.23	9598.45	
vrpnc05	199	f	1512.77	10112.13	5737.19	33182.36	4153
		f^*	1687.18	3040.09	2118.64	15630.12	
		f^*_{wc}	1687.18	3040.09	2118.64	15630.12	
vrpnc06	50	f	559.87	567.68	122.64	3350.03	98
		f^*	559.87	567.68	122.64	3350.03	
		f^*_{wc}	579.51	579.42	9.58	605.79	
vrpnc07	75	f	995.55	2583.40	2285.52	11686.45	415
		f^*	1070.72	1077.22	79.39	2512.38	
		f^*_{wc}	1070.72	1077.22	79.39	2512.38	
vrpnc08	100	f	945.33	946.03	38.49	2106.94	648
		f^*	945.33	946.03	38.49	2106.94	
		f^*_{wc}	965.35	965.21	11.81	999.43	
vrpnc09	150	f	1352.35	1506.98	659.95	6979.06	2265
		f^*	1404.10	1404.68	16.24	1460.92	
		f^*_{wc}	1404.10	1404.68	16.24	1460.92	
vrpnc10	199	f	1597.51	3683.01	2822.85	18401.49	5117
		f^*	1697.23	1698.18	31.61	2415.77	
		f^*_{wc}	1784.14	1783.08	18.10	1837.73	
vrpnc11	120	f	1260.55	4775.36	3331.22	21620.35	1229
		f^*	1492.17	1504.12	212.15	6115.15	
		f^*_{wc}	1507.27	1507.52	26.71	1601.84	
vrpnc12	100	f	883.55	8004.79	5372.87	30825.31	698
		f^*	1138.66	1638.14	1359.21	9534.97	
		f^*_{wc}	1138.66	1638.14	1359.21	9534.97	
vrpnc13	120	f	1368.33	4629.15	3227.07	18598.48	1079
		f^*	1565.28	1611.20	315.01	6819.88	
		f^*_{wc}	1727.20	1734.02	73.03	3117.71	
vrpnc14	100	f	952.46	7706.49	5126.85	29321.32	719
		f^*	1033.58	1775.53	1707.77	13073.67	
		f^*_{wc}	1125.71	2192.62	1891.78	12757.21	

Table 5 Vehicle routing with stochastic customers and recourse procedure

Data file	n	Criterion	f	f^*	σ^*	f^*_{wc}	CPU (s)
vrpnc01	50	f	527.46	424.39	28.30	494.89	99.30
		f^*	528.22	423.92	28.85	488.88	
		f^*_{wc}	528.22	423.92	28.85	488.88	
vrpnc02	75	f	905.22	736.65	45.95	854.83	408.59
		f^*	942.20	721.21	43.89	846.89	
		f^*_{wc}	927.41	728.53	47.00	836.21	
vrpnc03	100	f	852.05	703.72	34.09	788.00	573.17
		f^*	852.14	696.88	32.00	776.41	
		f^*_{wc}	852.14	696.88	32.00	776.41	
vrpnc04	150	f	1130.42	977.25	32.34	1067.67	1856.12
		f^*	1130.42	977.25	32.34	1067.67	
		f^*_{wc}	1134.48	979.88	33.22	1065.78	
vrpnc05	199	f	1466.17	1286.28	32.03	1366.59	4266.21
		f^*	1483.99	1282.70	39.68	1384.81	
		f^*_{wc}	1466.17	1286.28	32.03	1366.59	
vrpnc06	50	f	567.92	457.43	32.55	533.51	97.52
		f^*	577.01	456.09	34.14	553.61	
		f^*_{wc}	569.51	459.30	32.16	528.00	
vrpnc07	75	f	990.71	809.90	51.78	919.41	423.80
		f^*	1000.02	806.21	56.00	945.15	
		f^*_{wc}	990.71	809.90	51.78	919.41	
vrpnc08	100	f	933.72	786.23	36.69	872.24	655.88
		f^*	942.00	779.18	40.24	890.61	
		f^*_{wc}	933.72	786.23	36.69	872.24	
vrpnc09	150	f	1351.54	1152.77	40.75	1257.38	2329.98
		f^*	1351.54	1152.77	40.75	1257.38	
		f^*_{wc}	1351.54	1152.77	40.75	1257.38	
vrpnc10	199	f	1574.00	1379.91	41.69	1497.29	5102.75
		f^*	1577.60	1377.88	42.73	1483.68	
		f^*_{wc}	1577.60	1377.88	42.73	1483.68	
vrpnc11	120	f	1122.20	1019.96	21.55	1083.10	1215.21
		f^*	1123.32	1019.26	22.61	1081.70	
		f^*_{wc}	1125.26	1020.72	21.19	1074.73	
vrpnc12	100	f	861.27	774.77	24.27	839.91	698.23
		f^*	862.98	771.46	24.77	820.59	
		f^*_{wc}	862.98	771.46	24.77	820.59	
vrpnc13	120	f	1239.06	1121.11	22.49	1177.47	1008.80
		f^*	1250.98	1117.50	23.57	1177.95	
		f^*_{wc}	1239.06	1121.11	22.49	1177.47	
vrpnc14	100	f	878.46	768.03	23.92	833.47	722.35
		f^*	878.46	768.03	23.92	833.47	
		f^*_{wc}	878.46	768.03	23.92	833.47	

References

1. Artzner, P., Delbaen, F., Eber, J.-M., Heath, D.: Coherent measures of risk. Math. Financ. **9**(3), 203–228 (1999)
2. Bellman, R.: On a routing problem. Q. J. Appl. Math. **16**(1), 87–90 (1958)
3. Bertsimas, D.J.: A vehicle routing problem with stochastic demand. Oper. Res. **40**, 574–585 (1992)
4. Bianci, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. **8**(2), 239–287 (2008). doi:10.1007/s11047-008-9098-4
5. Birge, J.R.: Stochastic programming computation and applications. INFORMS J. Comput. **9**, 111–133 (1997)
6. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer, Dordrecht (2001)
7. 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)
8. Christofides, N., Mingozzi, A., Toth, P., Sandi, C.: Combinatorial Optimization. Wiley, Chichester (1979)
9. Dias, L.C., Clímaco, J.: On computing ELECTRE's credibility indices under partial information. J. Multi-criteria Decis. Anal. **8**, 74–92 (1999)
10. Dror, M.: Vehicle routing with stochastic demands: properties and solution frameworks. Transp. Sci. **23**(3), 166–176 (1989)
11. Dror, M., Laporte, G., Louveaux, F.V.: Vehicle routing with stochastic demands and restricted failures. Math. Methods Oper. Res. (ZOR) **37**(3), 273–283 (1993)
12. Gendreau, M., Laporte, G., Seguin, R.: An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp. Sci. **29**(2), 143–155 (1995)
13. Gendreau, M., Laporte, G., Séguin, R.: Stochastic vehicle routing. Eur. J. Oper. Res. **88**, 3–12 (1996)
14. Goetschalckx, M., Ahmed, S., Shapiro, A., Santoso, T.: Designing flexible and robust supply chains. In: Artiba, A. (ed.) Proceedings of the International Conference on Industrial Engineering and Production Management, pp. 539–551, Quebec, Canada (2001)
15. Haughton, M.A.: The performance of route modification and demand stabilization strategies in stochastic vehicle routing. Transp. Res. Part B **32**(8), 551–566 (1998)
16. Haughton, M.A., Stenger, A.J.: Comparing strategies for addressing delivery shortages in stochastic demand settings. Transp. Res., Part E **35**(1), 25–41 (1999)
17. Higle, J.L., Sen, S.: Stochastic decomposition: an algorithm for two stage stochastic linear programs with recourse. Math. Oper. Res. **16**, 650–669 (1991)
18. Homem-De-Mello, T.: Variable-sample methods for stochastic optimization. ACM Trans. Model. Comput. Simul. (TOMACS) **13**(2), 108–133 (2003)
19. Hvattum, L.M., Lokketangen, A., Laporte, G.: A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. Networks **49**(4), 330 (2007)
20. Infanger, G.: Planning Under Uncertainty: Solving Large Scale Stochastic Linear Programs. Boydand Fraser, Danvers (1994)
21. Jagannathan, R.: Use of sample information in stochastic recourse and chance-constrained programming models. Manage. Sci. **31**, 96–108 (1985)
22. Kelton, W.D., Sadowski, R.P., Sturrock, D.T.: Simulation with Arena. McGraw-Hill, London (2003)
23. Kenyon, A.S., Morton, D.P.: Stochastic vehicle routing with random travel times. Transp. Sci. **37**(1), 69 (2003)
24. Kleywegt, A.J., Shapiro, A., Homem-de-Mello, T.: The sample average approximation method for stochastic discrete optimization. SIAM J. Optim. **12**, 479–502 (2001)
25. Kouvelis, P., Yu, G.: Robust discrete optimisation and its applications. In: Nonconvex Optimization and its Applications, vol. 14. Kluwer, Dordrecht (1997)
26. Lambert, V., Laporte, G., Louveaux, F.V.: Designing collection routes through bank branches. Comput. Oper. Res. **20**, 783–791 (1993)
27. Laporte, G., Louveaux, F.C.V., van Hamme, L.: An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. Oper. Res. **50**(3), 415 (2002)
28. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. McGraw-Hill, London (1999)
29. Mulvey, J.M., Vanderbei, R.J., Zenios, S.A.: Robust optimization of large-scale systems. Oper. Res. **43**, 264–281 (1995)

30. Norkin, V.I., Ermoliev, Y.M., Ruszczyński, A.: On optimal allocation of indivisibles under uncertainty. Oper. Res. **46**, 381–395 (1998)
31. Norkin, V.I., Pflug, G.Ch., Ruszczyński, A.: A branch and bound method for stochastic global optimization. Math. Program. **83**, 425–450 (1998)
32. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**(12), 1985–2002 (2004)
33. Reimann, M.: Analysing risk orientation in a stochastic VRP. Eur. J. Ind. Eng. **1**(2), 111–130 (2007)
34. Roy, B.: A missing link in OR-DA: robustness analysis. Found. Comput. Decis. Sci. **23**, 141–160 (1998)
35. Secomandi, N.: Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Comput. Oper. Res. **27**(11–12), 1201–1225 (2000)
36. Secomandi, N.: A rollout policy for the vehicle routing problem with stochastic demands. Oper. Res. **49**(5), 796 (2001)
37. Secomandi, N.: Analysis of a rollout approach to sequencing problems with stochastic routing applications. J. Heuristics **9**(4), 321–352 (2003)
38. Sörensen, K.: A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design. Ph.D. thesis, University of Antwerp, Belgium (2003)
39. Sörensen, K.: Route stability in vehicle routing decisions: a bi-objective approach using metaheuristics. Cent. Eur. J. Oper. Res. **14**, 193–207 (2006)
40. Sörensen, K.: Investigation of practical robust and flexible decisions for facility location problems using tabu search and simulation. J. Oper. Res. Soc. **59**, 624–636 (2008)
41. Sörensen, K., Sevaux, M.: MA|PM: memetic algorithms with population management. Comput. Oper. Res. **33**(5), 1214–1225 (2005)
42. Stewart, W.R., Golden, B.L.: Stochastic vehicle routing: a comprehensive approach. Eur. J. Oper. Res. **14**, 371–385 (1983)
43. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia (2001)
44. Verweij, B., Ahmed, S., Kleywegt, A.J., Nemhauser, G., Shapiro, A.: The sample average approximation method applied to stochastic routing problems: a computational study. Comput. Optim. Appl. **24**, 289–333 (2003)
45. Vincke, P.: Robust solutions and methods in decision aid. J. Multi-criteria Decis. Anal. **8**, 181–187 (1999)
46. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**, 168–173 (1974)