A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling

Kenneth Sörensen, Marc Sevaux

A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling

Kenneth Sörensen*, Marc Sevaux†

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.

Key words : 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 of practical situations. As a result,

*University of Antwerp, Faculty of Applied Economics. kenneth.sorensen@ua.ac.be
†University of South-Brittany, LESTER, CNRS FRE 2734. marc.sevaux@univ-ubs.fr

a number of vehicle routing formulations designed to find *robust* 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. [1993]), problems with stochastic *demands* (e.g. Stewart and Golden [1983]), and problems with stochastic *customers*, i.e. in which each customer has a certain probability of requiring service (e.g. Bertsimas [1992]). For a survey, we refer to Gendreau et al. [1996].

For stochastic problems, *robust* or *flexible* solutions are required. Consistent with the definitions found in the literature (see e.g. Kouvelis and Yu [1997], Mulvey et al. [1995], Roy [1998], Vincke [1999], Dias and Clímaco [1999], Branke [2001]), 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, a solution that consistently has a high performance across all 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. In many cases however, a solution can be (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 any realization of the stochastic parameters. 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. In the remainder, we will call the procedure that is used to adapt the solution to the actual values of the stochastic parameters a *repair* procedure.

Several problems arise with these methods, that inhibit their use in real-life vehicle routing applications. First, most of the methods are *exact* methods, implying that they are unable to solve problems of realistic size. Indeed, as indicated by Birge [1997], 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. Because in realistic cases this number is usually very large or even infinite (in the case of continuous distributions), only very small problems can be solved to “optimality”. To partially overcome this problem, a part of the stochastic programming literature has fused on methods that use some form of *Monte Carlo sampling* of the stochastic parameters. For stochastic linear programming problems, *stochastic decomposition* [Higle and Sen, 1991] and *importance sampling* [Infanger, 1994] have been developed. For discrete stochastic problems, Norkin et al. [1998a,b] 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* [Kleywegt et al., 2001] 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 [Goetschalckx

et al., 2001] and routing problems [Verweij et al., 2003], among others.

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. Thirdly, almost all 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.

In this paper, we develop a method that is able to overcome these problems. 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. Secondly, 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 *robust evaluation function*.

2 Problem description and deterministic solution method

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, that has a set of identical vehicles of capacity Q and maximal travel cost 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 of the vehicles is not exceeded.

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 [2005].

In our MA|PM, solutions are represented as a string of customers, *without trip delimiters* $S = S_1 S_2 \ldots S_n$. When necessary (e.g. to calculate their objective function value) they are decoded optimally using the *split decoding procedure* by Prins [2004]. This pro-

cedure finds the optimal points at which to split the solution into tours 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.

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 distances covered in the trip $0 \rightarrow S_{i+1} \rightarrow S_{i+2} \rightarrow \ldots \rightarrow S_j \rightarrow 0$, plus the sum of drop costs in this trip. 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 [Bellman, 1958].

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 [Wagner and Fischer, 1974] calculates this measure in $O(n^2)$ where n is the length of both strings. Other, more efficient algorithms have been developed [Sörensen, 2003].

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 [2003] 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 the matching of trips in solution s to trips in solution t. The cost of the assignment problem is the distance between the two solutions.

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 to any solution already in the population. 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 [Sörensen and Sevaux, 2005].

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 distance 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 of items that were most recently swapped. Pairs that appear on the tabu list are excluded from swapping.

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.

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 (usually 10) solutions. 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. Both offspring solutions are then decoded and subjected to the tabu search local searches. When the solution satisfies the diversity criterion, it is added to the population. If not, it is *mutated* until it does satisfy the diversity criterion and added to the population. The mutation operator randomly swaps two customers and repeats this step until the resulting solution satisfies the diversity criterion.

We use an *adaptive* population management scheme, in which the value of Δ is slowly decreased to allow it 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.

Some experiments show that our MA|PM is competitive with the best-known approaches in the literature. For detailed results, we refer to Sörensen [2003].

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 so-called *robust evaluation function*, that measures the robustness or flexibility of the solution. A *repair 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 robust evaluation function value. Finally, we choose the solution that has the best robust 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 robust 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 repair procedure is available, the solution is repaired before it is evaluated. The concept of repair function is closely related to the stochastic programming concept of *recourse*, but differs from it that any procedure, even a heuristic one, may be used.

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

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

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 evalutions per robust evaluations, i.e. the number of evaluations of the solution on a random sampling of the stochastic parameters. If the solution can be repaired, the repair function $\mathcal{R}$ is invoked before evaluation the solution and the robust evaluation function becomes

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

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

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

or the standard deviation

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

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. 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. 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 robust 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

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

For a more elaborate discussion, we refer to Law and Kelton [1999].

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 robust evaluation function value as follows:

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

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.

We believe that our approach has several advantages over traditional methods based on stochastic programming, most importantly the fact that it 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 -improving moves.

Data sets for the stochastic versions are derived from the Christofides et al. [1979] 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 robust evaluation function value is based on $n_e = 1000$ evaluations.

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

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 demand and cost, a solution might violate the maximum travel cost constraints per route or the capacity constraints of the vehicles. 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.

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

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.

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

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.

$$\mathcal{P}_{\text{capacity}}(x; \mathcal{S}_k(\pi)) = \alpha_2 \sum_l \max(0, \sum_{v_i \in V_l(x)} q_{ik} - Q), \tag{3}$$

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 $\alpha_2 = 500$.

We look for solutions that have both a good average-case and a good worst-case performance. To this end, we define two robust 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. Robust 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 (table3). Table1 is an extract from this table holding the results of a single experiment.

Data file	n	**Criterion**	f	f^*	σ^*	f^*_{wc}
vrpnc01	50	f	549.76	3211.60	2891.00	17779.56
		f^*_1	604.76	605.01	13.21	876.66
		f^*_2	629.12	629.28	10.61	661.48

Table 1: Vehicle routing with stochastic demand and costs, example result

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. 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 their is a strong need to take the stochastic information into account.

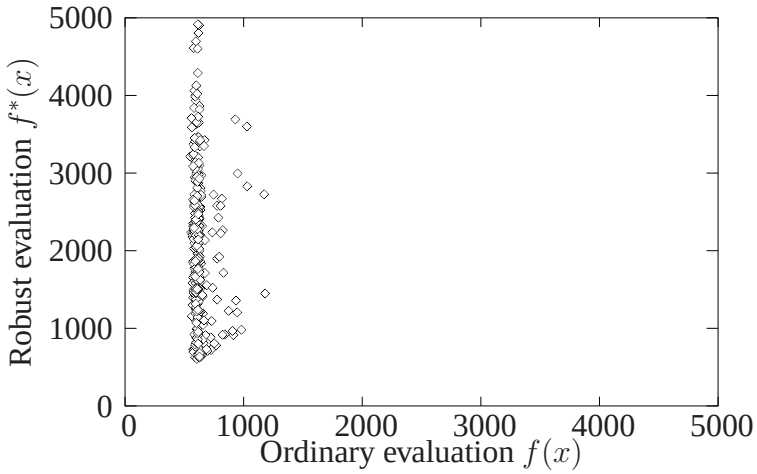

Figure 1: Ordinary evaluation and robust evaluation for vrpnc01

This is further illustrated in figure1, 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 on of the two robust solutions presented in table3 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 robust evaluation functions) generated by the MA|PM. These are not shown because of space restrictions.

4.2 Vehicle routing with stochastic customers and a repair function

In some situations, the vehicle dispatcher does not know in advance which customers will require service and which 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 repair 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. 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 table4.

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 robustness characteristics as the solution that optimizes one of the robust 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 robust evaluation function and computation times

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 robust evaluation function, should be taken with some care. Obviously, the robust evaluation function is used to evaluate the robustness properties of a solution. However, it can also be used to guide the search towards more robust solutions. In our MA|PM, the robust 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 robust 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 robust evaluation function is used. Indeed, it can be easily seen that 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-evaluating the entire solution, but they are no longer valid for a robust evaluation function.

By using the robust 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 2. 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.

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 robust 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 we mentioned before, a confidence interval can be calculated around the value of f^*. 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.

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

Table 2: Computing times (200 generations)

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 repair 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.

References

R. Bellman. On a routing problem. *Quarterly Journal of Applied Mathematics*, 16(1): 87–90, 1958.

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

J.R. Birge. Stochastic programming computation and applications. *INFORMS Journal on Computing*, 9:111–133, 1997.

J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, Boston, 2001.

N. Christofides, A. Mingozzi, P. Toth, and C.Sandi. *Combinatorial optimization*. John Wiley, Chichester, 1979.

L.C. Dias and J. Clímaco. On computing ELECTRE's credibility indices under partial information. *Journal of Multi-Criteria Decision Analysis*, 8:74–92, 1999.

M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88:3–12, 1996.

M. Goetschalckx, S. Ahmed, A. Shapiro, and T. Santoso. Designing flexible and robust supply chains. In A. Artiba, editor, *Proceedings of the International Conference on Industrial Engineering and Production Management*, pages 539–551, Quebec, Canada, 2001.

J.L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage stochastic linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

G. Infanger. *Planning Under Uncertainty: Solving Large Scale Stochastic Linear Programs*. Boyd and Fraser Publishing Co., Danvers, MA, 1994.

A.J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.

P. Kouvelis and G. Yu. *Robust Discrete Optimisation and its Applications*, volume 14 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 1997.

V. Lambert, G. Laporte, and F.V. Louveaux. Designing collection routes through bank branches. *Computers and Operations Research*, 20:783–791, 1993.

A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, London, 1999.

J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43:264–281, 1995.

V.I. Norkin, Y.M. Ermoliev, and A. Ruszczyński. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46:381–395, 1998a.

V.I. Norkin, G.Ch. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998b.

C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31(12):1985–20021, 2004. URL `doi:10.1016/S0305-0548(03)00158-8`.

B. Roy. A missing link in OR-DA: Robustness analysis. *Foundations of Computing and Decision Sciences*, 23:141–160, 1998.

K. Sörensen. *A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design*. PhD thesis, University of Antwerp, Belgium, 2003.

K. Sörensen and M. Sevaux. MA|PM: memetic algorithms with population management. *Computers and Operations Research*, 33(5):1214–1225, 2005. URL `doi:10.1016/j.cor.2004.09.011`.

W.R. Stewart and B.L. Golden. Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research*, 14:371–385, 1983.

B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

P. Vincke. Robust solutions and methods in decision aid. *Journal of Multi-Criteria Decision Analysis*, 8:181–187, 1999.

R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

Table 3: 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_1^*	604.76	605.01	13.21	876.66	
		f_2^*	629.12	629.28	10.61	661.48	
vrpnc02	75	f	891.86	8335.36	4986.75	30836.52	349
		f_1^*	956.57	1134.44	617.56	6509.91	
		f_2^*	956.57	1134.44	617.56	6509.91	
vrpnc03	100	f	887.97	2904.39	2753.68	16150.60	541
		f_1^*	971.38	977.51	72.99	2236.30	
		f_2^*	971.38	977.51	72.99	2236.30	
vrpnc04	150	f	1149.53	5313.22	3869.58	23450.44	1893
		f_1^*	1208.36	1596.03	1063.23	9598.45	
		f_2^*	1208.36	1596.03	1063.23	9598.45	
vrpnc05	199	f	1512.77	10112.13	5737.19	33182.36	4153
		f_1^*	1687.18	3040.09	2118.64	15630.12	
		f_2^*	1687.18	3040.09	2118.64	15630.12	
vrpnc06	50	f	559.87	567.68	122.64	3350.03	98
		f_1^*	559.87	567.68	122.64	3350.03	
		f_2^*	579.51	579.42	9.58	605.79	
vrpnc07	75	f	995.55	2583.40	2285.52	11686.45	415
		f_1^*	1070.72	1077.22	79.39	2512.38	
		f_2^*	1070.72	1077.22	79.39	2512.38	
vrpnc08	100	f	945.33	946.03	38.49	2106.94	648
		f_1^*	945.33	946.03	38.49	2106.94	
		f_2^*	965.35	965.21	11.81	999.43	
vrpnc09	150	f	1352.35	1506.98	659.95	6979.06	2265
		f_1^*	1404.10	1404.68	16.24	1460.92	
		f_2^*	1404.10	1404.68	16.24	1460.92	
vrpnc10	199	f	1597.51	3683.01	2822.85	18401.49	5117
		f_1^*	1697.23	1698.18	31.61	2415.77	
		f_2^*	1784.14	1783.08	18.10	1837.73	
vrpnc11	120	f	1260.55	4775.36	3331.22	21620.35	1229
		f_1^*	1492.17	1504.12	212.15	6115.15	
		f_2^*	1507.27	1507.52	26.71	1601.84	
vrpnc12	100	f	883.55	8004.79	5372.87	30825.31	698
		f_1^*	1138.66	1638.14	1359.21	9534.97	
		f_2^*	1138.66	1638.14	1359.21	9534.97	
vrpnc13	120	f	1368.33	4629.15	3227.07	18598.48	1079
		f_1^*	1565.28	1611.20	315.01	6819.88	
		f_2^*	1727.20	1734.02	73.03	3117.71	
vrpnc14	100	f	952.46	7706.49	5126.85	29321.32	719
		f_1^*	1033.58	1775.53	1707.77	13073.67	
		f_2^*	1125.71	2192.62	1891.78	12757.21	

Table 4: Vehicle routing with stochastic customers and repair 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^*_1	528.22	423.92	28.85	488.88	
		f^*_2	528.22	423.92	28.85	488.88	
vrpnc02	75	f	905.22	736.65	45.95	854.83	408.59
		f^*_1	942.20	721.21	43.89	846.89	
		f^*_2	927.41	728.53	47.00	836.21	
vrpnc03	100	f	852.05	703.72	34.09	788.00	573.17
		f^*_1	852.14	696.88	32.00	776.41	
		f^*_2	852.14	696.88	32.00	776.41	
vrpnc04	150	f	1130.42	977.25	32.34	1067.67	1856.12
		f^*_1	1130.42	977.25	32.34	1067.67	
		f^*_2	1134.48	979.88	33.22	1065.78	
vrpnc05	199	f	1466.17	1286.28	32.03	1366.59	4266.21
		f^*_1	1483.99	1282.70	39.68	1384.81	
		f^*_2	1466.17	1286.28	32.03	1366.59	
vrpnc06	50	f	567.92	457.43	32.55	533.51	97.52
		f^*_1	577.01	456.09	34.14	553.61	
		f^*_2	569.51	459.30	32.16	528.00	
vrpnc07	75	f	990.71	809.90	51.78	919.41	423.80
		f^*_1	1000.02	806.21	56.00	945.15	
		f^*_2	990.71	809.90	51.78	919.41	
vrpnc08	100	f	933.72	786.23	36.69	872.24	655.88
		f^*_1	942.00	779.18	40.24	890.61	
		f^*_2	933.72	786.23	36.69	872.24	
vrpnc09	150	f	1351.54	1152.77	40.75	1257.38	2329.98
		f^*_1	1351.54	1152.77	40.75	1257.38	
		f^*_2	1351.54	1152.77	40.75	1257.38	
vrpnc10	199	f	1574.00	1379.91	41.69	1497.29	5102.75
		f^*_1	1577.60	1377.88	42.73	1483.68	
		f^*_2	1577.60	1377.88	42.73	1483.68	
vrpnc11	120	f	1122.20	1019.96	21.55	1083.10	1215.21
		f^*_1	1123.32	1019.26	22.61	1081.70	
		f^*_2	1125.26	1020.72	21.19	1074.73	
vrpnc12	100	f	861.27	774.77	24.27	839.91	698.23
		f^*_1	862.98	771.46	24.77	820.59	
		f^*_2	862.98	771.46	24.77	820.59	
vrpnc13	120	f	1239.06	1121.11	22.49	1177.47	1008.80
		f^*_1	1250.98	1117.50	23.57	1177.95	
		f^*_2	1239.06	1121.11	22.49	1177.47	
vrpnc14	100	f	878.46	768.03	23.92	833.47	722.35
		f^*_1	878.46	768.03	23.92	833.47	
		f^*_2	878.46	768.03	23.92	833.47	