Metaheuristics for the Vehicle Routing Problem with Stochastic Demands

Leonora Bianchi[1], Mauro Birattari[2], Marco Chiarandini[3], Max Manfrin[2], Monaldo Mastrolilli[1], Luis Paquete[3], Olivia Rossi-Doria[4], and Tommaso Schiavinotto[3]

[1] IDSIA, USI-SUPSI, Switzerland, leonora@idsia.ch
[2] IRIDIA, Université Libre de Bruxelles, Belgium
[3] Intellectics Group, TU Darmstadt, Germany
[4] School of Computing, Napier University, UK

Abstract. In the vehicle routing problem with stochastic demands a vehicle has to serve a set of customers whose exact demand is known only upon arrival at the customer's location. The objective is to find a permutation of the customers (an a priori tour) that minimizes the expected distance traveled by the vehicle. Since the objective function is computationally demanding, effective approximations of it could improve the algorithms' performance. We show that a good choice is using the length of the a priori tour as a fast approximation of the objective, to be used in the local search of the several metaheuristics analyzed. We also show that for the instances tested, our metaheuristics find better solutions with respect to a known effective heuristic and with respect to solving the problem as two related deterministic problems.

1 Introduction

The Vehicle Routing Problem with Stochastic Demands (VRPSD) is defined on a complete graph $G = (V, A, D)$, where $V = \{0, 1, ..., n\}$ is a set of nodes (customers) with node 0 denoting the depot, $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs joining the nodes, and $D = \{d_{ij} : i, j \in V, i \neq j\}$ are the travel costs (distances) between nodes. The cost matrix D is symmetric and satisfies the triangular inequality. One vehicle with capacity Q has to deliver goods to the customers according to their demands, minimizing the total expected distance traveled, and given that the following assumptions are made. Customers' demands are stochastic variables ξ_i, $i = 1, ..., n$ independently distributed with known distributions. The actual demand of each customer is only known when the vehicle arrives at the customer location. It is also assumed that ξ_i does not exceed the vehicle's capacity Q, and follows a discrete probability distribution $p_{ik} = \text{Prob}(\xi_i = k)$, $k = 0, 1, 2, ..., K \leq Q$. A feasible solution to the VRPSD is a permutation of the customers $s = (s(1), s(2), \ldots, s(n))$ starting at the depot (that is, $s(1) = 0$), and it is called a priori tour. The vehicle visits the customers in the order given by the a priori tour, and it has to choose, according to the actual customer's demand, whether to proceed to the next customer or to go to

X. Yao et al (Eds.): PPSN VIII, LNCS 3242, pp. 450–460, 2004.

depot for restocking. Sometimes the choice of restocking is the best one, even if the vehicle is not empty, or if its capacity is bigger than the expected demand of the next scheduled customer, this action is called 'preventive restocking'. The goal of preventive restocking is to avoid the bad situation when the vehicle has not enough load to serve a customer and thus it has to perform a back-and-forth trip to the depot for completing the delivery at the customer. In the VRPSD, the objective is to find an a priori tour that minimizes the expected distance traveled by the vehicle, which is computed as follows. Let $s = (0, 1, \ldots, n)$ be an a priori tour. After the service completion at customer j, suppose the vehicle has a remaining load q, and let $f_j(q)$ denote the total expected cost from node j onward. With this notation, the expected cost of the a priori tour is $f_0(Q)$. If L_j represents the set of all possible loads that a vehicle can have after service completion at customer j, then, $f_j(q)$ for $q \in L_j$ satisfies $f_j(q) = \text{Minimum}\{f_j^p(q), f_j^r(q)\}$, where

$$f_j^p(q) = d_{j,j+1} + \sum_{k:k\leq q} f_{j+1}(q-k)p_{j+1,k} + \sum_{k:k>q} [2d_{j+1,0} + f_{j+1}(q+Q-k)]p_{j+1,k}, \quad (1)$$

$$f_j^r(q) = d_{j,0} + d_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)p_{j+1,k}, \quad (2)$$

with the boundary condition $f_n(q) = d_{n,0},\ q \in L_n$. In (1-2), $f_j^p(q)$ is the expected cost corresponding to the choice of proceeding directly to the next customer, while $f_j^r(q)$ is the expected cost in case preventive restocking is chosen. As shown in [1], the optimal choice is of threshold type: given the a priori tour, for each customer j there is a load threshold h_j such that, if the residual load after serving j is greater than or equal to h_j, then it is better to proceed to the next planned customer, otherwise it is better to go back to the depot for preventive restocking. The computation of $f_0(Q)$ runs in $O(nKQ)$ time; the memory required is $O(nQ)$, if one is interested in memorizing all intermediate values $f_j(q)$, for $j = 1, 2, ..., n$ and $q = 0, 1, ..., Q$, and $O(Q)$ otherwise.

The preventive restocking strategy for the VRPSD has been applied in [2] and [1]. In [2], a simple but effective heuristic called cyclic heuristic is studied. [1] focuses on the single and multiple-vehicle VRPSD. The authors analyze several heuristics and compare them with an exact branch-and-bound approach for small instances up to 15 customers. They also adapt to the stochastic case the local search due to Or [3], by proposing a fast approximation computation for the change in the objective function when performing a local search move. Approaches different from the restocking policy exist in the VRPSD literature, for references and a brief review see [4].

In this paper we focus on an important aspect of designing metaheuristics for the VRPSD (and for stochastic combinatorial optimization problems in general): the objective function is computationally demanding, and effective approximations of it should be employed. Due to the analogies between the VRPSD and the traveling salesman problem (TSP), a natural approximation of the objective

function is the length of the a priori tour. In fact, if the vehicle has infinite capacity, the consequent VRPSD can be seen as a TSP. In this paper we consider basic implementations of five metaheuristics: iterated local search (ILS), tabu search (TS), simulated annealing (SA), ant colony optimization (ACO) and evolutionary algorithms (EA). Our main goal is to test the impact on the metaheuristics performance of using the TSP length of the a priori tour as fast approximation of the VRPSD objective function. The solution quality of our metaheuristics are also compared with the cyclic heuristic and with a priori tours obtained by solving the problems as a TSP and as a deterministic VRP where the demand is assumed to be equal to the average demand of the VRPSD.

The remainder of the paper is organized as follows. Section 2 describes the metaheuristics, pointing out the common elements and the way the VRPSD objective function has been approximated. Section 3 reports on computational experiments, and section 4 summarizes the conclusions that can be drawn from the experimental results.

2 The Metaheuristics

As already stated in the introduction, the main goal of this study was to see whether approximating the exact but computationally demanding objective with the fast computing length of the a priori tour is convenient or not. Our hypothesis was that the speedup due to the use of a fast approximation of the objective would be an advantage especially during the phase of local search, when many potential moves must be evaluated before one is chosen. In fact, the exact cost of a local search move is the expected cost difference between the a priori tour after the move and before the move; however, this computation takes $O(nKQ)$ time. Hence, we consider two different approximation schemes for the move cost: i) *VRPSDlike*, that was suggested in [1], and requires $O(KQ)$ time, and ii) *TSPlike*, that requires $O(1)$ time. Given that all metaheuristics use the same local search, we consider two versions for each metaheuristic according to the type of local search approximation scheme. In the remainder of the paper, when we want to specify a version of a metaheuristic, we add to its name the -VRPSDlike or -TSPlike label (as, for example, ILS-VRPSDlike or ILS-TSPlike). Notice also that, depending on the metaheuristic, the local search may be used either as a black-box, or not. In particular, ACO, EA and ILS use the local search as a black-box, but TS and SA do employ their own strategy for examining the neighborhood of a solution. In the following, we describe in more detail the local search with its two approximation schemes, the initialization criterion of metaheuristics, the metaheuristics, and the other algorithms that took part at the experimental comparisons.

The OrOpt local search We chose the *OrOpt insertion* as suggested in [1]. Given a starting tour, sets S_k of k consecutive customers with $k \in \{1, 2, 3\}$ are moved from one position to another in the tour. In the following we describe the two types of approximation schemes used for the computation of the move cost.

VRPSDlike approach The move cost may be computed in two stages: i) compute the saving from extracting the set of costumers from the tour; ii) compute the cost of inserting it back somewhere else in the tour. Let i and $i+k+1$ be the nodes immediately preceding, respectively following, S_k in the tour, and let j be the node immediately after which S_k is to be inserted. Here, we assume that j is *after* i in the a priori tour. Let $f_i(q)$ and $f_{i+k+1}(q)$ be the expected cost-to-go from nodes i, respectively $i+k+1$ onward before the extraction of S_k. Apply one dynamic programming recursion step starting with cost vector $f_{i+k+1}(\cdot)$ at node $i+k+1$ back to node i, without considering the sequence S_k. Let $f'_i(\cdot)$ be the resulting cost vector at node i, that is, after extracting S_k from the tour. Then, define the approximate extraction saving as a simple average over q of $f'_i(q) - f_i(q)$. The computation of the approximate insertion cost of S_k between nodes j and $j+1$ in the tour, is done analogously, if we assume that the insertion point (node j) is after the extraction point (node i). Let $f_j(q)$ be the cost-to-go at node j before inserting S_k, and $f''_j(q)$ be the cost-to-go at node j after inserting the S_k. The total approximate cost of an OrOpt move is computed by subtracting the approximate extraction saving form the approximate insertion cost, as follows

$$\textit{VRPSDlike-move-cost} = \frac{\sum_{q=0}^{Q}[(f''_j(q) - f_j(q)) - (f'_i(q) - f_i(q))]}{Q+1} \ . \qquad (3)$$

Note that the cost vectors are assumed to be already available from the computation of the expected cost for the starting tour, thus, they do not need to be computed when evaluating eq. (3). The only computations that must be done here are the evaluation of cost vectors $f'_{i+1}(\cdot)$ and $f''_j(\cdot)$, requiring $O(KQ)$ time, and the average of eq.(3), requiring $O(Q)$ time. Therefore, with the proposed *VRPSDlike* approximation, the cost of an OrOpt move can be computed in $O(KQ)$ time. Although it is possible that tours which are worsening with respect to the evaluation function are accepted because recognized as improving by the approximate evaluation, in practice this approximation scheme behave quite well. For a deeper discussion on the issues related with this scheme we refer the reader to the original paper [1].

TSPlike approach In the *TSPlike* approach the cost of an OrOpt move coincides with the difference between the length of the tour before the move and after the move:

$$\textit{TSPlike-move-cost} = d_{i,i+k+1} + d_{j,i+1} + d_{i+k,j+1} - d_{i,i+1} - d_{i+k,i+k+1} - d_{j,j+1}, \ (4)$$

where, as before, i and j are the extraction, respectively insertion point of a string of k consecutive customers. Clearly, eq. (4) is computable in constant time.

The OrOpt neighborhood examination follows the same scheme proposed in [1]. Briefly, all possible sequences of length $k \in \{1, 2, 3\}$ are considered for insertion in a random position of the tour after the extraction point. Then, only the 'best' move among those of length k is chosen. The 'best' move is the move

corresponding to the most negative move cost, which is computed by eq. (3) in the *VRPSDlike* approach and by eq. (4) in the *TSPlike* approach.

Randomized Farthest Insertion The Farthest Insertion is a tour construction heuristic originally designed for the TSP; it builds a tour by choosing as next customer the not-yet-visited customer which is furthest from the current one, of course the final solution depends on the starting customer. Here, we consider the Randomized Farthest Insertion heuristic (FR), that picks randomly the first customer, and after the tour has been completed, the starting customer is shifted to the depot. Like in all our metaheuristics, after the tour is built, the OrOpt local search is applied for further improving it. All metaheuristics use FR for generating a starting solution (ILS, TS, SA), or a set of starting solutions (ACO, EA).

Simulated Annealing The SA metaheuristic [5] uses the local search operators described before, but also accepts non improving neighbors according to a function of a temperature parameter which depends on the deterioration in the cost function. The initial temperature is given by the average cost (*VRPSDlike* or *TSPlike*) of a sample of 100 tours of the initial tour multiplied by a given factor μ. Every $TL = \psi \cdot n$ iterations the temperature is updated as $T_{n+1} = \alpha \times T_n$ (standard geometric cooling). After $\rho \cdot TL$ without improvement in the approximate cost of a tour the temperature is increased by adding T_i to the current value. The tour considered for checking improvements is the best since the last re-heating. From preliminary experiments we set: $\mu = 0.05$, $\alpha = 0.98$, $\psi = 1$ and $\rho = 20$.

Tabu Search The TS metaheuristic [6] is based on the idea of accepting also worsening neighbors during local search but controlling cycles by avoiding the repetition of visited tours. For the VRPSD, we defined the tabu mechanism as follows. After an Or-opt move on the current tour S (extracting the sub-sequence S_k for a given size k from position i and re-inserting it at position j) the moves that become tabu are all insertions such that: i) a customer at position $i+1$ of S becomes again the successor of customer at position i of S; or ii) a customer at position $i+k+1$ of S becomes again the successor of customer at position $i+k$ of S. The tabu tenure is randomly chosen in the interval $[0.8(n-k-1), (n-k-1)]$. We consider a probabilistic acceptance criterion as follows: if a move is non tabu its cost is evaluated with a probability of 0.8 and selected if it leads to an improving tour. If the move is tabu its cost is evaluated with a probability of 0.3 and the move is selected in spite of its tabu status if it leads to the best tour found so far.

In TS-VRPSDlike the selection and acceptance of a move is done according to the *VRPSDlike* approximation scheme. The exact cost of the new tour obtained is then computed. In TS-TSPlike, the selection depends on the *TSPlike* approximation scheme, but the selected move is performed only if it leads to a real improvement of the exact cost of the tour. If the move selected does not

lead to a real improvement no move is performed and the search continues with a different k.

Iterated Local Search The ILS metaheuristic is based on the idea of improving the local search procedure by providing new starting tour obtained from a perturbation of the current solution [7]. In our implementation the *perturbation* consists in a sampling of n neighboring tours according to the 2-opt exchange neighborhood [8]. Each new tour is evaluated with the exact cost function and if a tour is found that has cost smaller than the best tour found so far plus ε, the sampling ends. $\varepsilon = \frac{n}{10}$ was empirically the best value found on some preliminary runs. Otherwise, the best perturbed tour is returned.

The local search uses respectively the *VRPSDlike* or the *TSPlike* approximation scheme. Finally, the *acceptance criterion* evaluates each new local optima found with the exact VRPSD cost function and accept it as current solution if it is the best tour found so far.

Ant Colony Optimization In the ACO metaheuristic, a set of agents (ants) build solutions to the given problem cooperating through pheromone-mediated indirect and global communication. Here, we consider the Ant Colony System (ACS), an ACO variant proposed in [9]. At each iteration, m ants construct a tour by building a complete sequence of customers using information stored in a "pheromone matrix" τ: $V \times V \rightarrow \Re_{\geq 0}$. The pheromone values $\tau_{i,j}$ estimate of the utility of going from a customer i to a customer j in the a-priori tour. An ant which is at customer i chooses a not-yet-visited customer j as next customer with probability proportional to $\tau_{i,j}$. After each construction step a *local update rule* is applied to the element $\tau_{i,j}$ corresponding to the chosen customer pair: $\tau_{i,j} = (1 - \psi) \cdot \tau_{i,j} + \psi \cdot \tau_0$, with $\psi \in [0,1]$. After all the m ants have built their tours, the local search is applied to each of them. Then, the best tour S_{best} found since the beginning of the run according to the exact VRPSD cost function is determined, and it is used in the *global update rule* to change all the entries in the pheromone matrix as follows: if $(i, j) \in S_{best}$, $\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j} + \rho \cdot \frac{q}{f(S_{best})}$, otherwise $\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j}$, where $q, \rho \in [0, 1]$ are parameters. Pheromone is initialized in the following way. First, all pheromone values are set to τ_0, and S_{best} is initialized with the best among m tours generated by the FR heuristic refined by the OrOpt local search. Then, the global update rule is applied r times to the pheromone matrix. In our ACS implementation, heuristic information is only used by FR in the initialization phase, but not in the ants' tour construction process. From preliminary experiments the set of parameters that was chosen is: $m = 5$, $\tau_0 = 0.5$, $\psi = 0.3$, $\rho = 0.1$, $q = 10^7$, $r = 100$. In ACS every tour outside the local search is evaluated with the exact cost function. The *VRPSDlike* and *TSPlike* approximation schemes are only used in the local search.

Evolutionary Algorithm EA metaheuristics are based on the essence of natural evolution processes, which involve the reproduction, random variation, competition, and selection of contending individuals in a population [10]. Here, the

population is initialized with solutions produced by the FR heuristic. At each iteration two parent solutions are chosen among the best ones to generate a new child solution through the recombination operator. After some preliminary experiments, we chose the Edge Recombination operator [11]. It tries to build a child tour exclusively from the edges present in both parent tours, whenever possible. A mutation operator, consisting of swapping adjacent customers without considering the depot, is applied with probability 0.5. Then, local search is used to improve the solution. Finally, the improved solution replaces the worst solution in the population. We use a population of size 10. As for ACS, in EA the only difference between the *VRPSDlike* version and the *TSPlike* version is in the local search they use.

Other Algorithms In order to better understand the influence of the stochasticity in the problem, two state of the art algorithms to solve the VRPSD as a TSP [12] and as a capacitated VRP (CVRP) [13] have also been included in our study. In the first case, the coordinates of all customers (depot included) are used to define a corresponding TSP instance; the instance is solved by the state of the art TSP algorithm and the solution found is then shifted so to start with the depot. In the second case, the coordinates, average customers' demands and vehicle capacity are used to define a corresponding CVRP instance. A CVRP solution in general is not a single a priori tour visiting the depot once, but it is composed by a set of tours visiting different sets of customers (apart from the depot). Such a solution is transformed into an a priori tour by keeping the same order in which customers are visited and by deleting the intermediate multiple visits to the depot. In both cases (TSP and CVRP) the final solution, once transformed into a VRPSD a priori tour, is finally evaluated with the exact cost function of the VRPSD.

For comparison with the existing literature about the VRPSD, we have also implemented the simple but effective cyclic heuristic [2]. Following the description of [2], the cycle heuristic works as follows. First, heuristically solve a TSP over the n customers (depot excluded). Then, for each of the n cyclic permutations of the tour found, use the 2-opt local search to obtain a new tour. Evaluate with the VRPSD objective function the $2n$ tours obtained, and choose the best one. The computational experience in [2] suggests that the cyclic heuristic provides good quality solutions for instances with 50 to 100 customers, uniformly distributed on the unit square, and it is therefore an example of the effectiveness of the TSP analogy.

3 Experimental comparisons

Instances and experimental setup In the literature there is no commonly used benchmark for the VRPSD, therefore we have generated our own testbed. We have tried to consider instances which are 'interesting' from different points of view. First of all, the position of customers was not chosen uniformly at random, but randomly with normal distributions around two centers (so customers are

grouped in two clusters). This is done in order to consider instances nearer to the real world situations, where customers may be located for instance in two different cities. The clusters' centers have coordinate in [0,99], and customers' coordinates are all different. We considered a total of 120 instances, of these, 75 instances have 50 customers, 40 instances have 100 customers, and 5 instances have 200 customers.

As it emerged from [14], an important factor that influences the 'difficulty' of a VRPSD instance is the ratio between the total (average) demand of customers and the vehicle's capacity. The bigger the ratio, the more 'difficult' the instance. Here, the vehicle capacity Q was chosen as $Q = \lceil \frac{\text{total average demand} \cdot r}{n} \rceil$, where the parameter r may be approximately interpreted as the average number of served customers before restocking. In our testbed, we have generated instances with $r = 4$, which corresponds to demand over capacity ratios from about 12 to 50, which is a much higher value than the values used in the VRPSD literature (typical values are below 3).

Each customer's demand is an integer stochastic variable uniformly distributed on an interval. The demand interval for each customer i was generated using two parameters: the average demand D_i, and the spread S_i, so that the possible demand values for customer i are the $2S_i + 1$ integers in the interval $[D_i - S_i, D_i + S_i]$. Average demands were chosen so that for each customer i, $D_i \in [1, 49]$ with probability 1/2, and $D_i \in [50, 100]$ with probability 1/2. Spreads were chosen so that for each customer i, $S_i \in \{1, 5\}$ with equal probability.

Each algorithm was tested once on each instance for a time equal to 60, 600 or 6000 seconds for instances respectively with 50, 100 or 200 customers. All algorithms, except the cyclic heuristic and the TSP state of the art algorithm, used all the available time for the computation. FR was restarted from a newly generated solution each time that the local search stopped in a local optimum. Experiments were performed on a cluster of 6 PCs with Athlon CPUs 1400MHz running GNU/Linux Debian OS, and all algorithms were coded in C++.

Results Results are summarized in the boxplots of Fig. 1. Each row of a boxplot shows the distribution of the quantity on the horizontal axis obtained by each metaheuristic on all the tested instances. The left plot of Fig. 1 reports on the horizontal axis the expected cost of the solutions found, normalized with respect to the range of improvement found by FR. So, for a given instance and a given metaheuristic MH, the normalized value reported on the boxplot is $(\text{MH}_{\text{ final value}} - \text{FR}_{\text{ final value}})/(\text{FR}_{\text{ starting value}} - \text{FR}_{\text{ final value}})$. The right plot of Fig. 1 shows the ranking of metaheuristics; the results of all executions on the same instance are ordered by quality of the solution (VRPSD expected cost) to determine the rank.

For the interpretation of the results, one could consider the performance of FR like a sort of minimal requirement for a metaheuristic. In fact, FR does essentially the simple iteration of the same local search for different starting solutions, until the available computation time is not over. Therefore, it is reasonable to

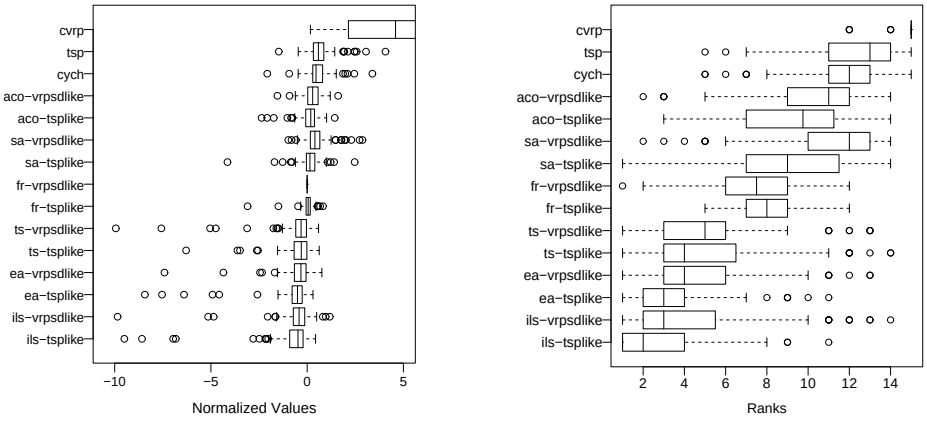

Fig. 1. Results over the 120 tested instances.

request that a good algorithm for the VRPSD perform significantly better than FR. From Fig. 1, it seems that only ILS, EA and TS perform significantly better than FR. This observation is confirmed by the one-tailed paired Wilcoxon test, confidence level 95% with Holmes corrections for multiple tests , that we have done for each couple MH-FR (with the same type of approximation scheme).

We have also verified if the TSPlike version of the metaheuristics is significantly better than the VRPSDlike version, and the answer is positive.We refer the reader interested in the details of the performed statistical tests to the webpage `http://iridia.ulb.ac.be/vrpsd.ppsn8/experiments.html`.

Another point to note is that the cyclic heuristic performs worse than all metaheuristics. This is an interesting results, since the cyclic heuristic was performing very well [2] for different types of instances (customers uniformly distributed on the unit square and low demand over capacity ratio). The worst algorithms in our tests are the two state of the art heuristics for the TSP, respectively CVRP; this is a point which encourages the development of VRPSD problem-specific algorithms, and let us conclude that, for the type of tested instances, the stochasticity of the problem is not negligible.

4 Conclusions

Our main goal in this paper was to test the impact on the metaheuristics' performance of using the length of the a priori tour as fast approximation of the exact but computationally demanding objective function. For this porpose, we have considered two different approximation schemes for evaluating the cost of a local search move: the *VRPSDlike* approximation (as also suggested in [1]) and the *TSPlike* approximation. We show experimentally that using the *TSPlike* approximation leads to better performing metaheuristics with respect to using the *VRPSDlike* approximation. We also show that for the tested instances, our metaheuristics find better solutions with respect to the cyclic heuristic (which is known from the literature to perform well on different types of instances)

and with respect to solving the problem as a traveling salesman problem and as a capacitated vehicle routing problem, which are related classical deterministic problems. Additionaly, we remark that the test instances for which we obtained the above results are characterized by high demand over capacity ratios. This feature makes our instances nearer to many real-world vehicle routing problem instances.

Acknowledgments

This work was supported by the *Metaheuristics Network,* a Research Training Network funded by the IHP programme of the EC, grant HPRN-CT-1999-00106. M. Manfrin also acknowledges support by "COMP2SYS", a Marie Curie Early Stage Training Site, funded by the the EC through the HRM programme. The information provided is the sole responsibility of the authors and does not reflect the EC's opinion. The EC is not responsible for any use that might be made of data appearing in this publication. This research was also funded by the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

1. W. Yang, K. Mathur, and R. H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, 2000.
2. D. J. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation Science*, 29(4):342–352, 1995.
3. I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking.* PhD thesis, Department of Industrial Engineering and Management Sciences, Nortwestern University, Evanston, IL, 1976.
4. L. Bianchi, M. Birattari, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the vehicle routing problem with stochastic demands. Technical Report IDSIA-06-04, IDSIA, April 2004.
5. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, (4598):671–680, 1983.
6. F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
7. H.R. Lourenço, O. Martin, and T. Stützle. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management*, chapter Iterated Local Search, pages 321–353. Kluwer Academic Publishers, Boston, U.S.A., 2002.
8. D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, Ltd., New York, U.S.A., 1997.
9. M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions On Evolutionary Computation*, 1(1):53–66, 1997.
10. T. Baeck, D. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing, Bristol, UK, 2000.

11. D. Whitley, T. Starkweather, and D. Shaner. The travelling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 350–372. Van Nostrand Reinhold, New York, U.S.A., 1991.
12. T. Stützle and H. Hoos. In P. Hansen and C. Ribeiro, editors, *Essays and Surveys on Metaheuristics*, chapter Analyzing the Run-time Behaviour of Iterated Local Search for the TSP, pages 589–612. Kluwer Academic Publishers, Boston, U.S.A., 2002.
13. L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, 1999.
14. M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Sciences*, 29(2):143–155, 1995.