Hybrid Metaheuristics for the Vehicle Routing Problem with Stochastic Demands

Leonora Bianchi[a] Mauro Birattari[b] Marco Chiarandini[c]
Max Manfrin[d] Monaldo Mastrolilli[e] Luis Paquete[f]
Olivia Rossi-Doria[g] Tommaso Schiavinotto[h]

[a]IDSIA, Lugano, Switzerland
[b]IRIDIA, Université Libre de Bruxelles, Belgium
[c]Intellectics Group, TU Darmstadt, Germany
[d]IRIDIA, Université Libre de Bruxelles, Belgium
[e]IDSIA, Lugano, Switzerland
[f]Intellectics Group, TU Darmstadt, Germany
[g]School of Computing, Napier University
[h]Intellectics Group, TU Darmstadt, Germany

Technical Report No. IDSIA-06-05
March 9, 2005

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

Abstract

This article analyses the performance of metaheuristics on the Vehicle Routing Problem with Stochastic Demands. The problem is known to have a computational demanding objective function, which could turn to be infeasible when large instances are considered. Therefore, fast approximations to the objective function would, at least, provide more time for searching for good quality solutions. Here, we explore the *hybridization* of the metaheuristic search process by interleaving the objective function with the one from a closely related problem (the traveling salesman problem) which can be computed in much less computation time. Moreover, we analyze several extensions to the metaheuristics which take this hybridization even further and report experimental results with respect to different types of instances. We show that for the instances tested, most metaheuristics perform better when hybridized with the traveling salesman objective function, and all metaheuristics find better solutions than state-of-the-art algorithms.

1 Introduction

Vehicle routing problems (VRPs) concern the transport of items between depots and customers by means of a fleet of vehicles. Solving a VRP means to find the best set of routes servicing all customers and respecting the operational constraints, such as vehicles capacity, time windows, driver's maximum working time. VRPs are a key issue in supply-chain and distribution systems today, and they are becoming increasingly complex. For this reason there is an ever increasing interest on routing models that are dynamic, stochastic, rich of constraints, and thus have objective functions more and more complex.

Models that focus particularly on the *stochasticity* of information are mainly known in the literature as Stochastic VRPs (SVRPs), and the problem we are addressing in this paper belongs to this class. In SVRPs elements of the problem such as the set of customers visited, the customers demands, or the travel times, are modeled as stochastic variables of known probability distributions, and the objective function is usually the expected cost of the planned routes. Note, however, that in SVRPs one needs to specify not only the concept of 'planned routes', but also the way planned routes are to be modified in response to the realization of the stochastic information.

One common feature of SVRPs is that they all have at least one deterministic counterpart, which is the VRP that one obtains by considering zero-variance probability distributions for the stochastic elements of the problem. SVRPs are thus NP-hard problems, like most VRPs. An important point that increases the difficulty of SVRPs is that they have an objective function (expected cost) which is much more computationally expensive than their deterministic counterparts. For this reason, a key issue in solving SVRPs by heuristics and metaheuristics is the use of fast and effective objective function approximations that may accelerate the search process. Due to the analogy between stochastic and deterministic VRPs, a reasonable choice for the objective function approximation of a given SVRP is the objective of the corresponding, or similar, deterministic problems.

In this paper, we investigate the use of objective function approximations derived from deterministic problems in the context of the VRPSD. This is an NP-hard problem, and despite the fact that it has a quite simple formulation, it arises in practice in many real world situations. One example is garbage collection, where it is indeed impossible to know a priori how much garbage has to be collected at each place. Another example where the demand is uncertain is the delivery of petrol to petrol stations. In fact, when a customer issues the order it is still unknown how much he will sell in the time between the order and the delivery.

In the VRPSD problem one vehicle of finite capacity is leaving from a depot with full load, and has to serve a set of customers whose exact demand is only known on arrival at the each customer location. A planned route in this context is very simple: a tour starting from the depot and visiting all customers exactly once; this is also called a priori tour, and it will be addressed as such in the remainder of the paper. The a priori tour is a sort of skeleton that fixes the order in which customers will be served, but the actual route the vehicle would travel would include return trips to the depot for replenishments when needed. The points at which return trips are performed are, in general, stochastic. The objective function to be minimized is the expected cost of the a priori tour.

Due to the nature of the a priori tour, a feasible solution for the VRPSD may also be seen as a feasible solution for a traveling salesman problem (TSP) on the set of customers (depot included). Moreover, if the vehicle has infinite capacity, the consequent VRPSD is, in fact, a TSP. Due to these analogies, a natural approximation of the VRPSD objective function is the length of the a priori tour.

In this paper we consider basic implementations of five metaheuristics: simulated annealing [23], tabu search [14], iterated local search [25], ant colony optimization [10] and evolutionary algorithms [1]. Our main goal is to test the impact on metaheuristics of interleaving the exact VRPSD objective function with the a priori tour length as an approximation of it. This mixture changes the *search landscape* during the search for good quality solutions and can be seen as innovative type of hybridization of metaheuristic's search process that has not been yet explored in the literature. In particular, we investigate two types of hybridization: first, we consider a local search algorithm (OrOpt) for which a quite good approximation for the exact VRPSD objective function is available, and we compare metaheuristics using this underlined local search by applying both VRPSD approximation and TSP approximation. Second, we further exploit the TSP analogy, by choosing the 3-opt local search operator, which is very good for the TSP, but for which there is no immediate VRPSD approximation.

The remainder of the paper is organized as follows. In section 2 we give the formal description of the VRPSD, we describe in detail the objective function, the state of the art about the VRPSD, and the relevant aspects of generating a benchmark of instances for this problem, taking into account the existing literature. Section 3 describes at high level the metaheuristics and the other algorithms analyzed. Section 4 reports details about tested instances, parameters used for the metaheuristics, computation times allowed. Sections 5 and 6 describe the computational experiments respectively on the first type of hybridization (the use of TSP objective function in OrOpt) and on the second type of hybridization (the use of the 3-opt local search with the TSP objective function). Section 7 summarizes the main conclusions that can be drawn from the experimental results.

2 The Vehicle Routing Problem with Stochastic Demand

The 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 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.

The expected distance traveled by the vehicle (that is, the objective function), 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

$$f_j(q) = \text{Minimum}\{f_j^p(q), f_j^r(q)\}, \tag{1}$$

where

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

$$f_j^r(q) = d_{j,0} + d_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)p_{j+1,k}, \tag{3}$$

with the boundary condition $f_n(q) = d_{n,0}$, $q \in L_n$. In (2-3), $f_j^p(q)$ 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 by Yang et al. in [33], 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 following algorithm is an implementation of the recursion (2-3) for the calculation of $f_0(Q)$ and of the thresholds.

Procedure 1 Computation of the VRPSD objective function $\mathbf{f_0(Q)}$

```
for ($q = Q, Q-1, ..., 0$) do
  $f_n(q) = d_{n,0}$
  for ($j = n-1, n-3, ..., 1$) do
    compute $f_j^r$ using $f_{j+1}(\cdot)$ (by means of equation (3))
    for ($q = Q, Q-1, ..., 0$) do
      compute $f_j^p(q)$ (by means of equation (2))
      compare $f_j^r$ and $f_j^p(q)$ for finding the threshold $h_j$
      compute $f_j(q)$ using $f_{j+1}(\cdot)$ (by means of equation (1))
    end for
  end for
end for
compute $f_0(Q)$
return $f_0(Q)$
```

The literature about VRPSD and SVRPs in general is quite rich. Formulations of SVRPs include the Traveling Salesman Problem with Stochastic Customers (TSPSC), the Traveling Salesman Problem with Stochastic Travel Times (TSPST), the Vehicle Routing Problem with Stochastic Customers (VRPSC), the Vehicle Routing Problem with Stochastic Customers and Demands (VRPSCD). For a survey on the early approaches to these problems, see [12] and [4]; for a more recent survey, especially on mathematical programming approaches used for SVRPs, see [22]. In the following we summarize the main contributions to solve VRPSD and similar problems, relevant to this paper.

- Jaillet [17, 18] and Jaillet-Odoni [19] derive analytic expressions for the computation of the expected length of a solution for the Probabilistic Traveling Salesman Problem and variations of it;

- Bertsimas [2] proposes the cyclic heuristic for the VRPSD, by adapting to a stochastic framework one of the heuristics presented by Haimovitch and Rinnooy Kan [16] in a deterministic context; later, Bertsimas et al. [3] improve this heuristic by applying dynamic programming, to supplement the a priori tour with rules for selecting returns trips to the depot, similarly to the preventive restocking strategy; their computational experience suggests that the two versions of the cyclic heuristic provide good quality solutions when customers are randomly distributed on a square region;

- Gendreau, Laporte and Séguin [11] present an exact stochastic integer programming method for VRPSCD (the same method can be applied to VRPSD as well); by means of the integer L-shaped method [24] they solve instances with up to 46 and 70 customers and 2 vehicles, for the VRPSCD and VRPSD, respectively; in [13], they also develop a tabu search algorithm called TABUSTOCH for the same problem; this algorithm is to be employed when instances become too large to be solved exactly by the L-shaped method;

- Teodorović and Pavković [31] propose a Simulated Annealing algorithm to the multi-vehicle VRPSD, with the assumption that no more than one route failure is allowed during the service of each vehicle.

- Gutjahr [15] applies S-ACO to the Traveling Salesman Problem with Time Windows, in case of stochastic service times. S-ACO is a simulation-based Ant Colony Optimization algorithm, that computes the expected cost of a solution (the objective function), by Monte Carlo sampling.

- Secomandi [27, 28] applies Neuro Dynamic Programming techniques to the VRPSD; he addresses the VRSPD with a re-optimization approach, where after each new exact information about customers demand is updated, the a priori tour planned on the not yet served customers is completely re-planned; this approach may find solutions with a lower expected value with respect to the preventive restocking strategy, but it is much more computationally expensive; moreover, the a priori tour may be completely different from the actual tour followed by the vehicle, and this situation is often seen as a disadvantage by companies;

- Yang et al. [33] investigate the single- and multi-vehicle VRPSD; the latter is obtained by imposing that the expected distance traveled by each vehicle does not exceed a given value; the authors test two heuristic algorithms, the route-first-cluster-next and the cluster-first-route-next, which separately solve the problem of clustering customers which must be served by different vehicles and the problem of finding the best route for each cluster; both algorithms seem to be efficient and robust for small size instances, as shown by comparisons with branch-and-bound solutions to instances with up to 15 customers; The authors also adapt to the stochastic case (both single- and multi-vehicle VRPSD) the OrOpt local search due to Or [26], by proposing a fast approximation computation for the change of the objective function value of a solution modified with a local search move.

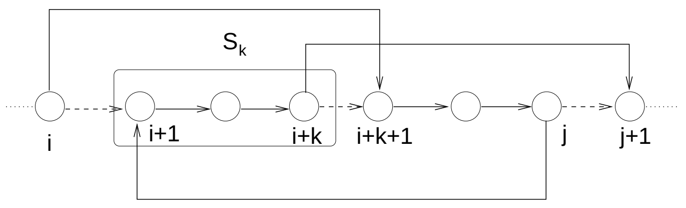

Figure 1: How an a priori tour is modified after performing an OrOpt move, where the set of consecutive customers S_k (here, $k = 3$) is moved forward in the tour.

The OrOpt local search and objective function approximation are used in the present paper as building blocks of our metaheuristics, therefore we will describe them in detail in the following subsection.

2.1 The OrOpt local search

A basic move of the OrOpt local search, as suggested by Yang et al. in [33], works as follows. Given a starting a priori tour, sets S_k of k consecutive customers with $k \in \{1, 2, 3\}$ are moved from one position to another in the tour, like in Figure 1. In the following we describe the two types of approximation schemes used for the computation of the move cost. The first one, that we call *VRPSD*, is the one proposed in [33], and the second one, that we call *TSP*, only computed the length change of the a priori tour.

VRPSD approximation scheme The move cost is 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. , as shown in Figure 1. 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

$$\Delta_{\text{VRPSD}} = \frac{\sum_{q=0}^{Q}[(f''_j(q) - f_j(q)) - (f'_i(q) - f_i(q))]}{Q + 1} . \tag{4}$$

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 equation (4). 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.(4), requiring $O(Q)$ time. Therefore, with the proposed *VRPSD* 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 [33].

TSP approximation scheme In the *TSP* approximation scheme the cost of an OrOpt move coincides with the difference between the length of the tour before the move and after the move:

$$\Delta_{\text{TSP}} = d_{i,i+k+1} + d_{j,i+1} + d_{i+k,j+1} - d_{i,i+1} - d_{i+k,i+k+1} - d_{j,j+1}, \tag{5}$$

where, as before, i and j are the extraction, respectively insertion point of a string of k consecutive customers (see Figure 1). Clearly, Δ_{TSP} is computable in constant time.

The OrOpt neighborhood examination follows the same scheme proposed in [33]. 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. (4) in the *VRPSD* approach and by eq. (5) in the *TSP* approach.

2.2 Benchmark

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, by controlling four factors in the generation of instances: customer position, capacity over demand ratio, variance of the stochastic demand, and number of customers.

Instances may be divided in two groups, uniform and clustered, according to the position of customers. In uniform instances, the position of customers is chosen uniformly at random on a square of fixed size. In clustered instances, coordinates are chosen randomly with normal distributions around a given number of centers. This results in clusters of nodes, a typical situation for companies serving customers positioned in different cities.

The ratio between the total (average) demand of customers and the vehicle's capacity is an important factor that influences the 'difficulty' of a VRPSD instance [11]. The bigger the ratio, the more 'difficult' the instance. Here, the vehicle capacity Q is 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.

Each customer's demand is an integer stochastic variable uniformly distributed on an interval. The demand interval for each customer i is 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 $[D_i - S_i, D_i + S_i]$. The spread is a measure of the variance of the demand of each customer.

The particular parameters used to generate the test instances for our experiments are reported in section 4.

3 The metaheuristics

We aim at an *unbiased* comparison of the performance of well-known five different metaheuristics to this problem. In order to obtain a fair and meaningful analysis of the results, we have restricted the metaheuristic approaches to the use of the common OrOpt local search. In the following we briefly and schematically describe the main principles of each metaheuristic and give the details of the implementations for the VRPSD.

- **Simulated Annealing (SA)**

```
Determine initial candidate solution $s$
Set initial temperature $T$ according to *annealing schedule*
While termination condition not satisfied:
    Probabilistically choose a neighbor $s'$ of $s$
    If $s'$ satisfies probabilistic acceptance criterion (depending on $T$):
        $s := s'$
    Update $T$ according to annealing schedule
```

The initial temperature T_i is given by the average cost of a sample of 100 solutions of the initial tour multiplied by μ; every $\psi \cdot n$ iterations the temperature is updated by $T \leftarrow \alpha \times T$ (standard geometric cooling); after $\rho{\cdot}\psi{\cdot}n$ without improvement, the temperature is increased by adding T_i to the current value; the solution considered for checking improvements is the best since the last re-heating.

- **Tabu Search (TS)**

```
Determine initial candidate solution $s$
While termination criterion is not satisfied:
    Probabilistically choose a neighbor $s'$ of $s$
    If $s'$ satisfies acceptance criterion (depending on its tabu status):
        $s := s'$
    Update tabu attributes based on $s'$
```

It follows a probabilistic acceptance criterion. The all neighborhood of a solution is explored, and the best *evaluated* neighbor is selected. Neighbors evaluation works as follows: if a neighbor is non tabu, its cost is evaluated according to a probability of p_{nt}; otherwise, its cost is evaluated with a probability of p_t and it may be set as new solution s' only if it leads to the best solution found so far. In this way, if all neighbors of a solution are non tabu, the new solution s' can be a worsening solution, since the best among neighbors is selected without comparing it with the best solution so far.

- **Iterated Local Search (ILS)**

```
Determine initial candidate solution $s$
Perform local search on $s$
While termination criterion is not satisfied:
    $r := s$
    Perform perturbation on $s$
    Perform local search on $s$
    Based on acceptance criterion, keep $s$ or revert to $s := r$
```

The perturbation consists in a sampling of n neighbors according to the 2-opt exchange neighborhood [20]; each new solution is evaluated with by exact cost function (Procedure 1) and if a solution is found that has cost smaller than the best solution found so far plus ε, the sampling ends; otherwise, the best solution obtained during the sampling is returned; the acceptance criterion keeps s if it is the best solution found so far.

- **Ant Colony Optimisation (ACO)**

```
Initialise weights (pheromone trails)
While termination criterion is not satisfied:
    Generate a population $sp$ of solutions by a
                        randomised constructive heuristic
    Perform local search on $sp$
    Adapt weights based on $sp$
```

Pheromone trails are initialized to τ_0; *sp* solutions are generated by a constructive heuristic and refined by local search; a *global update rule* is applied r times; *sp* solutions are then constructed by using information stored in the pheromone matrix; 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 local search, weights are again updated by the global update rule (note that heuristic information is only used in the initialization phase).

- **Evolutionary Algorithm (EA)**

```
Determine initial population sp
While termination criterion is not satisfied:
    Generate a set spr of solutions by recombination
    Generate a set spm of solutions from spr by mutation
    Perform local search on spm
    Select population sp from solutions in sp, spr, and spm
```

At each iteration two solutions are chosen among the best ones to generate a new solution *spr* through Edge Recombination [32] (a tour is generated using edges present in both two other tours, whenever possible); The mutation swaps adjacent customers (without considering the depot) with probability p_m; finally, the solution improved by local search replaces the worst solution in the population.

The initial solution(s) for all metaheuristics are obtained by the constructive heuristic Farthest Insertion [21]; it builds a tour by choosing as next costumer the *not yet visited* customer which is farthest from the current one. Here, we consider a randomized version of this heuristic (RFI) which picks the first customer at random, and after the tour has been completed, shifts the starting customer to the depot.

In order to use a reference algorithm for comparison among metaheuristics (see section 4), we also implemented a simple random restart algorithm which uses the RFI heuristics *plus* local search and restarts everytime a local optimum is found, until the termination condition is reached; the best solution found among all the restarts is picked as the final solution; we call such algorithm RR.

4 Experimental Setup

In this article we report two computational experiments with two different goals: i) we analyse metaheuristics performance under the hypothesis of the relation between the TSP and VRPSD; ii) we study the performance of enhanced versions of the best algorithms found in the first set of experiments moreover, we relate these results with parameters of the instance.

We consider instances with 50, 100 and 200 customers and with customers uniformly or clustered distributed. In uniform instances, the position of customers is chosen uniformly at random in the square $[0, 99]^2$. Clustered instances are generated with two clusters, with centers randomly chosen in the square $[0, 99]^2$. The variance of the normal distribution used to generate customers coordinates around the centers is equal to $(0.8/\sqrt{n})\cdot$(max. coordinate). This corresponds to variance values of about 11, 8, and 6, respectively for instances with 50, 100, and 200 customers. The position of the depot is fixed at (1,1). The average number of customers served before restocking is maintained fixed to 4, thus yielding ratios for total demand over vehicle capacity in the range from 12 to 50. Typical values for this ratio in the VRPSD literature are below 3, however higher values are closer to the needs of real contexts [5]. In all instances, average demands D_i at each customer

i are taken with equal probability from $[1, 49]$ or $[50, 100]$. With regard to demand spread, instead, instances may be divided in two groups: low spread instances, in which each customer i has S_i chosen at random in $[1, 5]$, and high spread instances, in which each customer i has S_i chosen at random in $[10, 20]$. For each combination of size, distribution of customers and demand spread 75 instances were generated, totalizing 900 instances.

The parameters of each metaheuristic were fine-tuned with respect to the instances under study; some preliminary experiments indicated that the following settings would provide acceptable results:

SA: $\mu = 0.05$, $\alpha = 0.98$, $\psi = 1$, and $\rho = 20$;

TS: $p_{nt} = 0.8$ and $p_t = 0.3$;

ILS: $\varepsilon = \frac{n}{10}$;

ACO: $m = 5$, $\tau_0 = 0.5$, $\psi = 0.3$, $\rho = 0.1$, $q = 10^7$, and $r = 100$;

EA: $spm = 10$, $p_m = 0.5$.

Given the results reported in [6, 7], we decided to only perform one run for each metaheuristic on each instance. The termination criterion for each algorithm was set to a time equal to 60, 600 or 6000 seconds for instances respectively of 50, 100 or 200 customers. Experiments were performed on a cluster of 8 PCs with AMD Athlon(tm) XP 2800+ CPU running GNU/Linux Debian 3.0 OS, and all algorithms were coded in C++ under the same development framework.

In order to compare results among different instances, we normalized results with respect to the performance of RR. For a given instance, we denote as c_{MH} the cost of the final solution of a metaheuristic MH, c_{RFI} the cost of the solution provided by the RFI heuristic, and C_{RR} by cost of the final solution provided by RR; the normalized value is then defined as

$$\text{Normalized Value for MH} = \frac{c_{MH} - c_{RR}}{c_{RFI} - c_{RR}}. \tag{6}$$

Besides providing measure of performance independent from different instance hardness, this normalization method gives an immediate evaluation of the minimal requirement for a metaheuristic; it is reasonable to request that a metaheuristic performs at least better than RR within the computation time under consideration.

5 First hybridization: using approximate move costs in local search

The main goal of this first experiment is 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 is that the speedup due to the use of a fast approximation of the objective is an advantage especially during the phase of local search, when many potential moves must be evaluated before one is chosen.

In order to test the advantage of a speedup across the metaheuristics, we apply to each of them the OrOpt local search described in section 2.1, and we test two versions for each metaheuristic according to the type of approximation scheme used in the local search, *VRPSD-approximation* or *TSP-approximation*. This set up allows to use statistical techniques for a systematic experimental analysis. In particular, we consider a two-way factor analysis, where metaheuristics and type of objective function are factors and instances are considered as blocks [9].

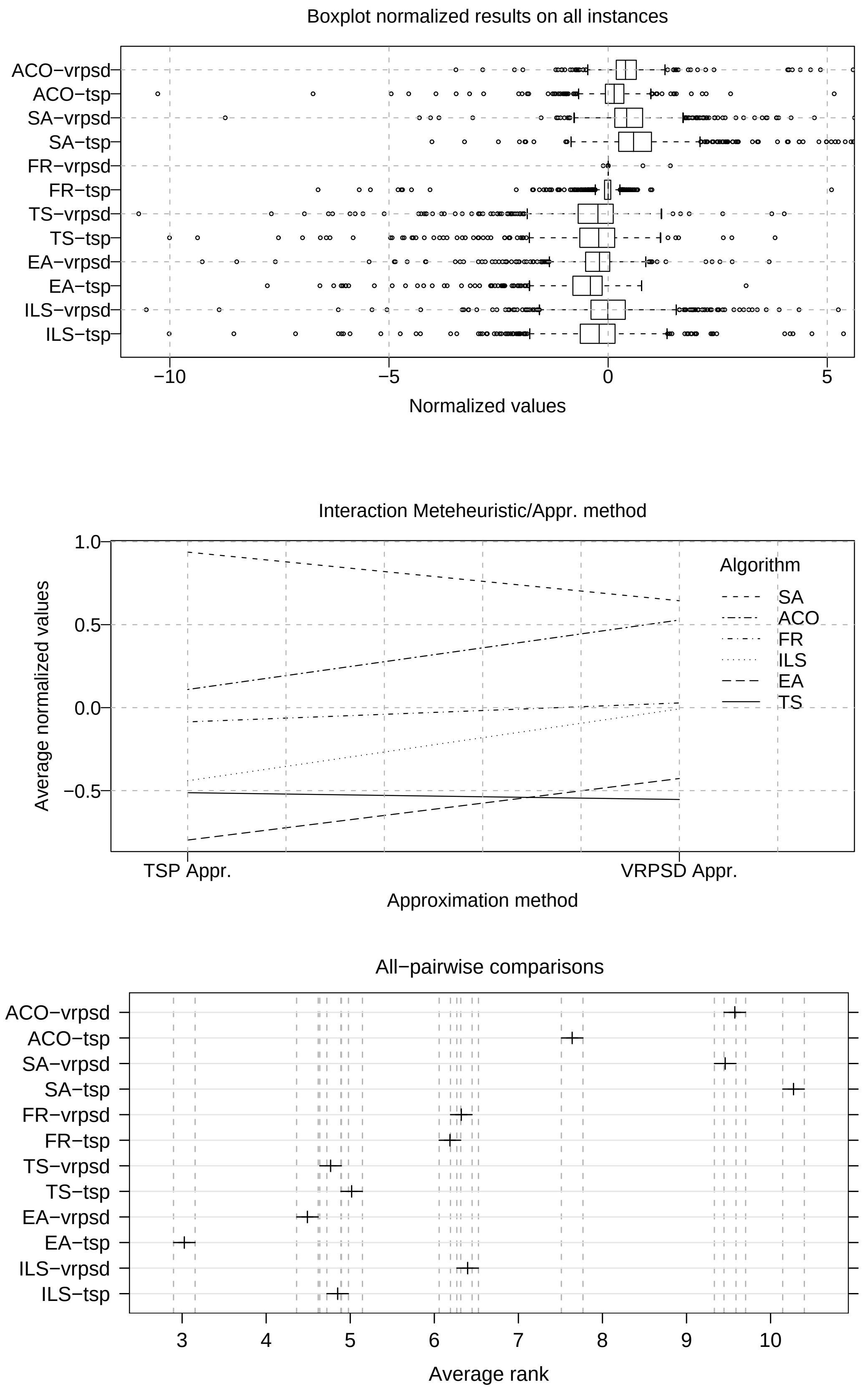

Figure 2: Aggregate results over all 900 instances. From top to bottom: boxplot of normalised results; interaction plot for the two factors: metaheuristic and objective function approximation scheme; error bars plot for simultaneous confidence intervals in the all-pairwise comparison. The boxplot is a restricted to values in $[-10.5, 5]$, few outliers fell outside this interval. In the error bar plot, ranks range from 1 to 12, the number of algorithms considered in the experiment.

The boxplot of normalized results in Figure 2 shows that ILS, EA and TS are able to achieve results which are even 5 or 10 times better than RR while SA and TS, in some cases, perform even worse. We checked that these results are also statistically significant. The assumptions for a parametric analysis are not met, hence we rely on non-parametric methods.

The central plot of Figure 2 shows the interaction between the factor metaheuristic and approximation scheme. Interaction plots give an idea of how different combinations of metaheuristic and approximation method affect the average normalised result. The lines join the average value for each metaheuristic. If lines are not parallel it means that metaheuristics perform differently with different approximation methods. This effect between the two factors is called *interaction effect*. From the plot, the two factors interact, hence, it would be appropriate to report the effects of one factor separately for each level of the other. In the third plot of Figure 2 we report, however, together both VRPSD and TSP approximation schemes, but we distinguish the effects of this factor on each metaheuristic. The plot presents the simultaneous confidence intervals for the all-pairwise comparison of algorithms. Interval widths are obtained by the Friedman two-way analysis of variance by ranks [8], after rejecting the hypothesis that all algorithms perform the same. The difference between two algorithms is statistically significant at a level of confidence of 5% if their intervals do not overlap.

From the interaction and the all-pairwise comparison plots of Figure 2 we can conclude that:

- The improvements of EA, ILS, and TS over RR are statistically significant.
- The presence of an interaction between metaheuristic and approximation scheme shows that EA, ILS and ACO perform better with *TSP-approximation* while the opposite result holds for TS and SA. While EA, ILS and ACO use the local search as a black-box, TS and SA employ their own strategy for examining the neighborhood in the local search. This result indicates that, for becoming competitive, these two methods require a good approximation of the objective function.
- The metaheuristics which perform better are EA, ILS and TS. Furthermore, EA and ILS the TSP take significant advantage from *TSP-approximation* scheme.

6 Second Hybridization: further exploiting the TSP analogy

Given the results in the previous section, it is reasonable to investigate what happens if we exploit the hybridization based on the TSP objective function even further. For this purpose, we consider one of the best performing TSP state-of-the-art metaheuristics, and we observe that it is based on iterated local search with the 3-opt local search operator [30]. Thus, we also consider the better performing metaheuristics of the previous section (ILS, EA and TS) for *hybridization* with the 3-opt local search. Note however that TS does not need to be hybridised because it presents already better results with the *VRPSD-approximation* than with the *TSP-approximation.* This new algorithms are described as follows.

TSP-soa It is a state-of-the-art algorithm for TSP [30]: it uses the 3-opt exchange neighborhood, and a double bridge move as perturbation, i.e., it removes randomly four edges from the current tour and adds four other edges which close the tour also chosen at random; the coordinates of all customers (depot included) are used to define a corresponding TSP instance; the TSP solution found is shifted to start with the depot in order to be used as VRPSD solution, and eventually evaluated by the VRPSD objective function (Procedure 1). Note that If the solution quality of this algorithm was comparable to our metaheuristics, there would be no point in investigating algorithms specific for the VRPSD, because the problem would be practically equivalent to the TSP, at least for the instances of our benchmark.

ILS-hybrid It is like the *TSP-soa* but it uses a different acceptance criterion: it keeps the best according to Procedure 1.

EA-hybrid It also uses a 3-opt local search based on TSP objective function. The recombination and mutation operators are maintained unchanged, since these are supposed to be the components that determined the success of the algorithm in Figure 2. The selection operator of the evolutionary algorithm remains also unchanged, and is based on the VRPSD objective function computed by Procedure 1.

In order to compare our best performing algorithms with what is available in the literature for the VRPSD, we also include in the experiments the effective CYCLIC heuristic [2], that we implemented as follows:

- solve a TSP over the n customers, plus the depot, by using the TSP-soa algorithm;
- consider the n a priori tours obtained by the cyclic permutations $s_i = (0, i, i+1, ..., n, 1, ..., i-1), i = 1, 2, ..., n$;
- evaluate the n a priori tours by the VRPSD objective function (Procedure 1) and choose the best one.

For the comparison of these algorithms we designed an experimental analysis based on a single factor layout with blocks [9]. Since the assumption for parametric tests are again not met we use the Friedman two-way analysis of variance by ranks.

In Figure 3, 4, 5 we present the simultaneous confidence intervals after rejecting the hypothesis that all algorithms perform the same. We distinguish results by presenting the instances grouped according to the three main features defined in Section 4. Since we test the data collected three times, we adjust the "family-wise" level of confidence for each test dividing it by a factor of three [29].

The main indication arising from the results is that a mere TSP algorithm is not sufficient to produce high quality solutions for the VRPSD instances under any of the circumstances tested. VRPSD problem-specific algorithms, which take into account the stochasticity of the problem perform always better. Nevertheless, the hybridisation of TSP and VRPSD approaches leads to the best algorithms for VRPSD. *EA-hybrid* and *ILS-hybrid* in many cases perform statistically better than all other algorithms and differences are considerable. On the contrary, differences between these two approaches are never statistically significant. The comparison with the CYCLIC heuristic indicates that the algorithms we presented improve considerably the solution for VRPSD with respect to previous approaches in the literature.

Considering the characteristics of the instances, the position of customers seems not to have an impact on the rank of algorithms. On the contrary, the different rank of algorithms in relation to instance size indicates that the larger the instances are the higher is the importance of speed over precision in the evaluation of neighbors in local search. This expected result is exemplified by the worsening of the average rank of *EA-tsp* as size increases.

Finally, spread of demand has also a considerable influence. With small spread it is convenient to consider the costs of "preventive restockings" and thus the use of Procedure 1 is appropriate. With large spread, instead, the higher uncertainty of the right point in the tour for "preventive restocking" makes the evaluation of solution by means of Procedure 1 not far from that provided by a TSP evaluation, with the negative impact of a higher computational complexity.

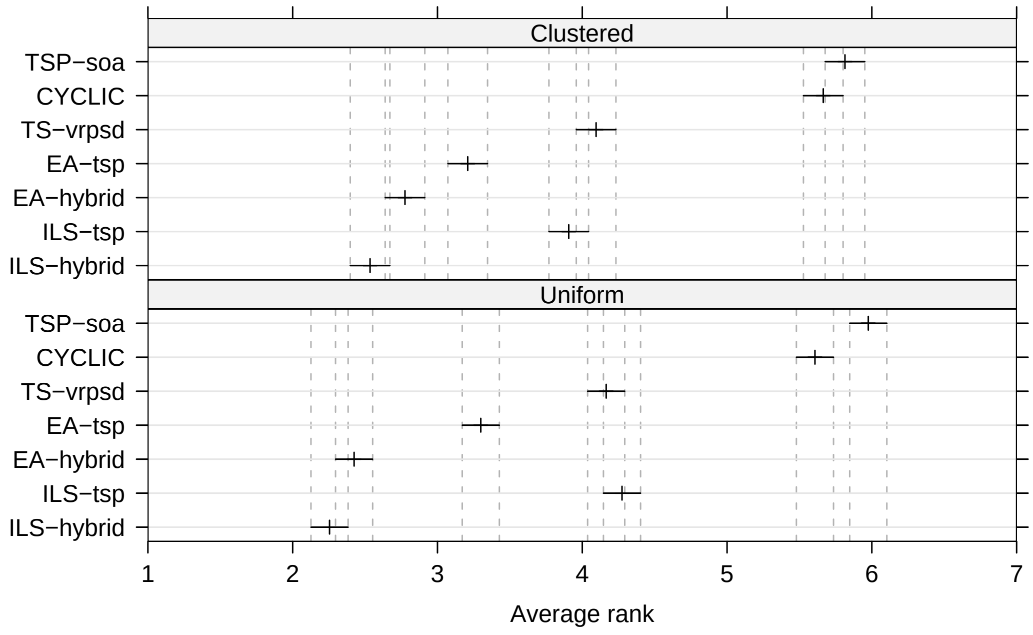

Figure 3: All-pairwise comparisons by means of simultaneous confidence intervals on uniform and clustered instances.

7 Conclusions

Our main goal in this paper was to test the impact on five well known metaheuristics of using the length of the a priori tour (*TSP-approximation*) as fast approximation of the exact but computationally demanding objective function.

For this purpose, we investigated two types of hybridization: first, we considered a local search algorithm (OrOpt) for which a quite good approximation for the exact VRPSD objective function is available, and we compared metaheuristics using this local search by applying both the *VRPSD-approximation* and the *TSP-approximation*. Second, we further exploited the TSP analogy, by choosing the 3-opt local search operator, which is very good for the TSP, but for which there is no immediate VRPSD approximation.

With the first type of hybridization, we have obtained that metaheuristics using the local search as a black-box (EA, ILS and ACO) perform better when using the *TSP-approximation*, while metaheuristics that use their own strategy for examining the neighborhood in the local search, perform better with the more precise but more computationally expensive *VRPSD-approximation*. With the second type of hybridization based on the use of the 3-opt local search, we considerably improved the performance of our metaheuristics, even though the difference between the best performing ones (EA and ILS) were not significant.

All the implemented metaheuristics found 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 TSP. This last point shows that it is not possible to neglect the stochasticity of the problem and the finite demand over capacity ratio, and it encourages

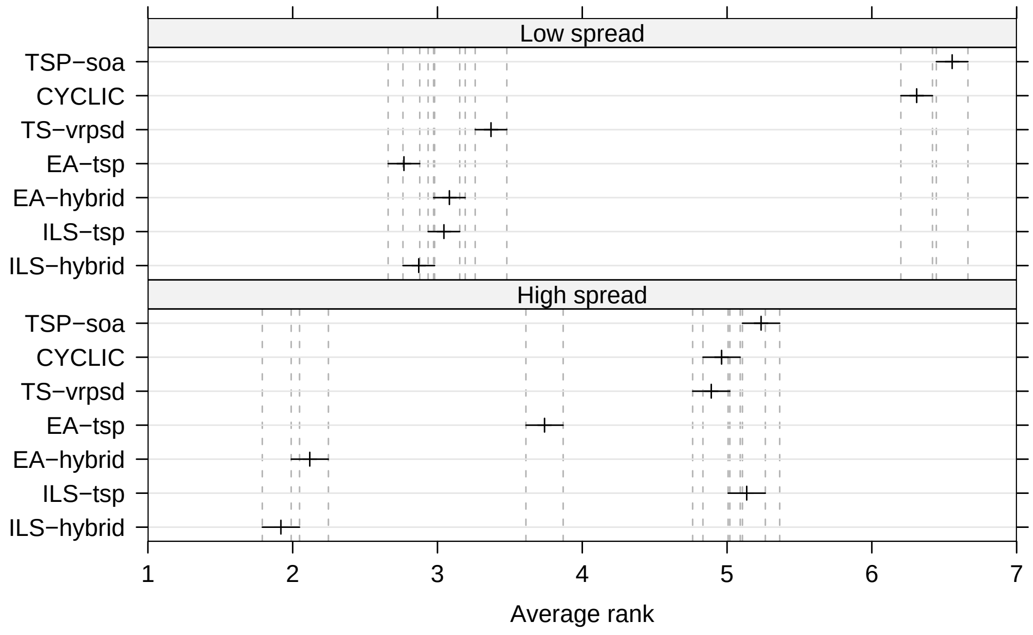

Figure 4: All-pairwise comparisons by means of simultaneous confidence intervals on instances with different demand spread.

the development of VRPSD-specific algorithms, even if the exploitation of the TSP fast objective function has shown to be very effective for improving the metaheuristics performance.

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] T. Baeck, D. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing, Bristol, UK, 2000.

[2] D. J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[3] D. J. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation Science*, 29(4):342–352, 1995.

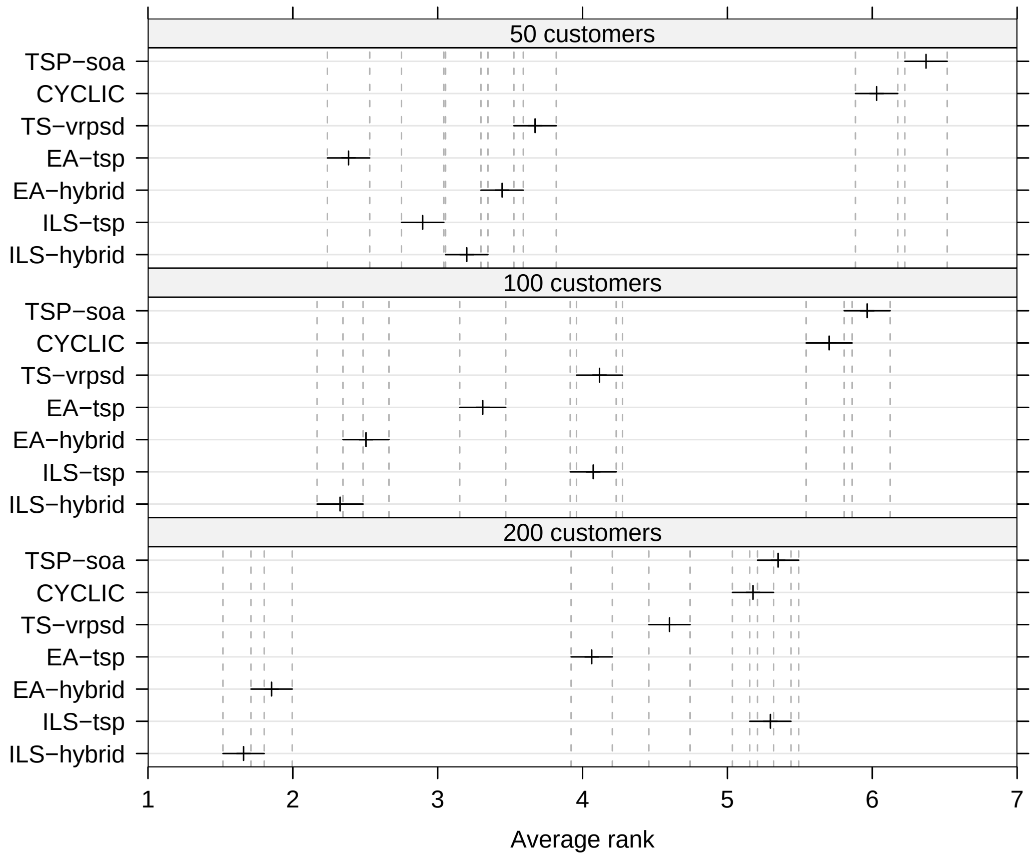

Figure 5: All-pairwise comparisons by means of simultaneous confidence intervals on instances with different number of customers.

[4] D. J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(2):216–304, 1996.

[5] L. Bianchi, M. Birattari, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Research on the vehicle routing problem with stochastic demand. Technical Report IDSIA-07-04, IDSIA, March 2004.

[6] M. Birattari. On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Technical Report TR/IRIDIA/2004-01.2, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[7] M. Birattari. *The Problem of Tuning Metaheuristics, as seen from a machine learning perspective.* PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[8] W. J. Conover. *Practical Nonparametric Statistics.* John Wiley & Sons, New York, NY, 1999.

[9] A. Dean and D. Voss. *Design and Analysis of experiments.* Springer-Verlag New York, 1999.

[10] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[11] 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.

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

[13] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3), 1996.

[14] F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[15] W. Gutjahr. S-ACO: An ant-based approach to combinatorial optimization under uncertainty. In *Proceedings of ANTS 2004 – Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 238–249. Springer Verlag, Heidelberg, Germany, 2004.

[16] M. Haimovitch and A. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10:527–542, 1985.

[17] P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, MIT, Cambridge, MA, 1985.

[18] P. Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.

[19] P. Jaillet and A. Odoni. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, chapter The probabilistic vehicle routing problems. Elsevier, Amsterdam, The Netherlands, 1988.

[20] 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.

[21] D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[22] A. Kenyon and D. P. Morton. A survey on stochastic location and routing problems. *Central European Journal of Operations Research*, 9:277–328, 2002.

[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, (4598):671–680, 1983.

[24] G. Laporte and F. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 33:133–142, 1993.

[25] 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.

[26] 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.

[27] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers and Operations Research*, 27(5):1171–1200, 2000.

[28] N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[29] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures.* Chapman & Hall, second ed., 2000.

[30] 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.

[31] D. Teodorović and G. Pavković. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. *Transportation Planning and Technology*, 16:261–273, 1992.

[32] 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.

[33] W. Yang, K. Mathur, and R. H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, 2000.