Out-of-the-Box and Custom Implementation of Metaheuristics. A Case Study: The Vehicle Routing Problem with Stochastic Demand

Paola Pellegrini and Mauro Birattari

Abstract. Metaheuristics are a class of effective algorithms for optimization problems. A basic implementation of a metaheuristic typically requires rather little development effort. With a significantly larger investment in the design, implementation, and fine-tuning, metaheuristics can often produce state-of-the-art results. According to the amount of development effort, we say that an implementation of a metaheuristic is either an *out-of-the-box* version or a *custom* one. The possibility of implementing metaheuristics in such a flexible way is one of the major strengths of these algorithms. Nonetheless, it also hides some possible catches. In particular, it should be noticed that results obtained with *out-of-the-box* implementations cannot be always generalized to *custom* ones, and vice versa. The goal of this analysis is to stress that these two ways of using metaheuristics are different. As a case study, we focus on the vehicle routing problem with stochastic demand and on five among the most successful metaheuristics—namely, tabu search, simulated annealing, genetic algorithms, iterated local search, and ant colony optimization. We show that the relative performance of these algorithms strongly varies whether one considers *out-of-the-box* implementations or *custom* ones, in which the parameters are accurately fine-tuned. Moreover, we underline the relevance of clearly stating the framework in which the results reported in the literature have been obtained. To this aim, we consider also an implementation of the same algorithms as described in the literature.

Paola Pellegrini
Department of Applied Mathematics, Università Ca' Foscari Venezia,
Cannaregio 873, 30121, Venice, Italy
e-mail: paolap@unive.it

Mauro Birattari
IRIDIA, CoDE, Université Libre de Bruxelles, 50, Av. F. Roosevelt,
CP 194/6, B-1050 Brussels, Belgium
e-mail: mbiro@ulb.ac.be

M. Köppen et al. (Eds.): Intelligent Comput. Optimization in Eng., SCI 366, pp. 273–295.
springerlink.com

1 Introduction

The term *metaheuristics* [1] is nowadays widely adopted for designating a class of approaches to tackle optimization problems.

> *A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems.*
>
> Dorigo and Stützle, 2004 [2, p. 25]

The generality of metaheuristics and the ease with which they can be applied to the most diverse combinatorial optimization problems is definitely the main reason for their success. Indeed, compared to exact algorithms and problem-specific heuristics, metaheuristics typically require a much lower design and implementation effort. This is particularly true if one does not necessarily aim at state-of-the-art results but has the main goal of obtaining a fairly good performance, while minimizing the development costs. In these cases, an *out-of-the-box* implementation of a metaheuristic is typically the solution of choice for many practitioners. On the other hand, in a number of applications it has been shown that state-of-the-art performance can be obtained through metaheuristics, provided that a *custom* version is developed by taking extra care in the design, implementation, and fine-tuning. This, quite naturally, implies higher development costs.

This flexibility of metaheuristics is definitely one of their appealing traits: In practical applications, one can start with an *out-of-the-box* version of a metaheuristic for quickly having some preliminary results and for gaining a deeper understanding of the problem at hand. Then one can move to a *custom* version for obtaining a better performance without having to switch to a completely different technology.

Nonetheless, the fact that metaheuristics can be flexibly used either in their *out-of-the-box* or *custom* versions, can be reason of misunderstanding. Indeed, results obtained with *out-of-the-box* implementations do not always generalize to *custom* ones, and vice versa. In particular, it could well happen that, as we show in the case study proposed in this work, a metaheuristic M_1 performs better than a metaheuristic M_2 on a given problem when *out-of-the-box* versions of M_1 and M_2 are considered; whereas M_2 performs better that M_1 on the very same problem when *custom* versions are considered.

This issue is unfortunately overlooked in the literature: Many research papers propose comparisons of metaheuristics without providing any measure of the development effort devoted to the algorithms under analysis or, in other words, without clearly stating whether the metaheuristics considered are *out-of-the-box* versions or rather high-performing *custom* versions. Without this piece of information, the usefulness of these comparisons is somehow impaired[1].

The lack of specification about the context in which empirical studies are performed can be partially justified by the fact that, admittedly, measuring the amount of development effort is not a simple and well-defined task. Much of the ambiguity

[1] In a similar way the performance assessment method used may have an impact on the relative performance of metaheuristics. For the analysis of this impact see for example [3, 4, 5, 6, 7].

comes from the fact that there is no such thing as the *standard developer*: What costs a great effort to somebody with limited experience in the domain, might be effortless for a seasoned practitioner. The issue is further complicated by the fact that researchers and practitioners often specialize on one metaheuristic (or on few). For example, if an expert in genetic algorithms devotes the same time and attentions to the development of a genetic algorithm and of a tabu search, the resulting algorithms will have a relative performance that is expectedly much different from the one that would be obtained if the algorithms had been developed by a tabu search expert.

Following the preliminary analysis proposed by Pellegrini and Birattari [8], in this study we show the difference that may result between two experimental studies, one performed in the *out-of-the-box* context, and the other in the *custom* one. Moreover, we highlight the implications of using algorithms as they are described in the literature. In this case, the question is whether it makes sense to use parameters that have been selected as reasonably high performing in some context that is possibly different from the one under analysis. To this aim, we consider as a case study the vehicle routing problem with stochastic demand, and five of the most successful metaheuristics—namely, tabu search, simulated annealing, genetic algorithm, iterated local search, and ant colony optimization. The goal is to show that the relative performance of the above metaheuristics depends on the implementations considered. With this work, we wish to draw the attention of the research community on this issue and contribute to establish a better practice for the empirical analysis and comparison of metaheuristics.

What we wish to underline is that the scope of the studies should be made clear. Different researches may have different goals, that may justify the use of either *out-of-the-box* or *custom* versions of metaheuristics. Nonetheless, we should be aware of the fact that the results achieved may not be generalizable. If we use for our experiments an implementation that was built with a goal that is different from ours, we may be mislead. For supporting this conclusion, we will consider also implementations of the five metaheuristics as they are described in the literature. We will show that the relative performance of these versions is different from the one of both the *out-of-the-box* and the *custom* versions of the same metaheuristics.

In order to attenuate the problem concerning the different ability of a single designer in implementing various approaches, we consider the implementations of the five metaheuristics produced within the *Metaheuristics Network*,[2] a EU funded research project started in 2000 and accomplished in 2004. In the *Metaheuristics Network*, five academic groups and two companies, each specialized in the development and application of one or more of the above metaheuristics, joined their research efforts with the aim of gathering a deeper insight into the theory and practice of metaheuristics. For a detailed description of the metaheuristics developed by the *Metaheuristics Network* for the vehicle routing problem with stochastic demand, we refer the reader to Bianchi et al. [9]. The availability of this reference makes the

[2] `http://www.metaheuristics.net/`

vehicle routing problem with stochastic demand, and the five metaheuristics above mentioned, particularly suitable to our analysis.

In our analysis, these implementations are considered as *black-box* metaheuristics: By modifying their parameters, we obtain the *out-of-the-box*, the *custom*, and what we call the *literature* versions. The first ones are obtained by randomly drawing the parameters from a defined range. The second ones are obtained by fine-tuning the parameters through an automatic procedure based on the F-Race algorithm [10, 11, 12]. This removes some of the ambiguity connected with the measure of the development effort and guarantees that equal attention is devoted to all metaheuristics under analysis. The last ones are exactly the implementations described in the literature: They are obtained considering the implementations and the values of the parameters reported by Bianchi et al. [9]. To the best of our knowledge this paper represents the most recent and successful application of metaheuristics to the problem considered.

The fact of reducing the difference between the *out-of-the-box* and the *custom* version of metaheuristics to the fine-tuning of the parameters is not free of implications and needs to be further justified. The following two observations in favor of the validity and significance of our analysis should be sufficient in order to convince our reader. Although many research papers fail to provide an exhaustive account on how the parameters of the algorithms under analysis are obtained, it is widely recognized that an accurate fine-tuning has a major impact on the performance of algorithms [11, 13, 3, 14]. Selecting the best values for the parameters, given the class of instances that are to be tackled, is definitely a sort of customization. Other elements, as for example an advanced design and implementation of critical data structures, clearly play a major role in the *custom* implementation of a metaheuristic. Nonetheless, the goal of the study is to show that an analysis based on *custom* implementations might produce radically different results from one based on *out-of-the-box* implementations. If we succeed to show this fact when even one single element characterizing *custom* implementations is considered, namely the fine-tuning of parameters, we have nevertheless reached our goal. The use of advanced data structures in the *custom* implementation could only enhance the difference observed.

The rest of the chapter is organized as follows. In Section 2, we present a panoramic view of the literature concerning the vehicle routing problem with stochastic demand, the metaheuristics considered, and the tuning problem. In Section 3, we describe the specific characteristics of these elements as they appear in our analysis. In Section 4, the experimental study is reported. Finally, in Section 5, we make some conclusions.

2 Literature Overview

In this section, we provide the reader with a general overview of the available literature concerning the three main topics of interest of our analysis: i) the vehicle routing problem with stochastic demand, ii) the five metaheuristics we consider in this study, and iii) the problem of fine-tuning metaheuristics.

The problem we consider in our analysis is the vehicle routing problem with stochastic demand (VRPSD). It can be described as follows: Given a fleet of vehicles with finite capacity, a set of customers has to be served at minimum cost. The peculiarity of this variant of the vehicle routing problem is that the demand of each customer is *a priori* unknown and only its probability distribution is available. The actual demand is revealed only when the customer is reached. In this probabilistic setting, the objective of the VRPSD is the minimization of the total expected traveling cost.

Optimal methods, heuristics, and metaheuristics have been proposed in the literature for tackling this problem. In particular, the problem is first addressed by Tillman [15] in 1969. Stewart and Golden [16], Dror and Trudeau [17], Laporte and Louveau [18] and Laporte et al. [19] used techniques from stochastic programming to solve optimally small instances. Bertsimas [20] and Bertsimas and Simchi-Levi [21] proposed different heuristics for solving the VRPSD. They considered the construction of an *a priori* TSP-wise tour. This tour is then split according to precise rules. Yang et al. [22] proposed a strategy for splitting the *a priori* tour allowing the restocking before a stockout, when this is profitable. Secomandi [23, 24, 25] analyzed different possibilities for applying dynamic programming to this problem. Teodorović and Pavković [26] and Gendreau et al. [27] tackled the VRPSD using metaheuristic approaches. In particular, Teodorović and Pavković [26] adopted simulated annealing while Gendreau et al. [27] used tabu search. Finally, an extended analysis on the behavior of different metaheuristics has been proposed by Bianchi et al. [9].

Two classical local search algorithms have been used for the VRPSD: the Or-opt and the 3-opt procedures. The first was proposed by Or [28] in 1976. It consists in the extraction of a string of consecutive nodes from the starting sequence representing a solution, and in its insertion at a different position. Yang et al. [22] presented an approximated way for computing the value of each move. In particular, the cost saving allowed by a move is the difference between the approximated saving obtained removing the string of consecutive customers from its original position, and the approximated cost of inserting it somewhere else in the tour: Instead of evaluating the complete solution before and after the move, the saving and cost are computed only with respect to the immediate neighbors of the customers shifted. This method has been adopted also by Bianchi et al. [9], who proposed also another approximation based on delta values calculated in a TSP-wise fashion, that is, considering the variation of the length of the *a priori* tour. Moreover, they extended this TSP-wise approximation also to the 3-opt local search [29]. In this case, three edges belonging to the starting solution are removed and replaced by three different ones.

Following Bianchi et al. [9], we focus on five of the most popular metaheuristics: tabu search (TS), simulated annealing (SA), genetic algorithm (GA), iterated local search (ILS), and ant colony optimization (ACO).

Tabu search has been introduced by Glover [1] in 1986, on the basis of early ideas formulated a decade before [30]. It consists in the exploration of the solution space via a local search procedure. Tabu search accepts non-improving moves and

uses a short term memory. The latter expedient is introduced to avoid sequences of moves that constantly repeat themselves [31].

Simulated annealing takes inspiration from the annealing process in crystals, which assume a low energy configuration when cooled with an appropriate cooling schedule [32, 33, 34, 35, 36]. The principal idea is the exploration of the search space via a local search procedure. Simulated annealing escapes from local minima by allowing moves to worsening solutions. The parameter that controls this mechanism is the *temperature*. It is slowly decreased during the search with the consequence that at the beginning the probability of accepting non-improving solutions is higher, and then it decreases over time. This technique helps in quitting the basin of attraction of high-cost local minima that might be encountered in the early stages of the search.

Genetic algorithms are inspired by the ability shown by populations of living beings to evolve and adapt to changing conditions, under the pressure of natural selection [37]. This metaheuristic is based on the selection of individuals representing candidate solutions. From generation to generation, individuals evolve through *recombination* and *crossover*, and using *mutation* or *modification* operators that lead to self-adaptation [38, 39, 40, 41, 42, 43].

Iterated local search is one of the simplest metaheuristics. It is based on the reiteration of a local search procedure: It explores the neighborhoods of a sequence of solutions obtained via successive perturbations [44].

Ant colony optimization is a metaheuristic based on the foraging behavior of ants [2, 45]. It constructs solutions using a pheromone model, that is, a parameterized probability distribution over the solution space. The solutions found are used to modify the pheromone values biasing the search toward high quality solutions [46].

Tuning is a critical issue when working with metaheuristics. Each metaheuristic can be seen as a modular structure coming with a set of components, each typically provided with a set of free parameters. The tuning problem is the problem of properly instantiating this algorithmic template by choosing the best among the set of possible components and by assigning specific values to all free parameters [12]. Although this problem is generally recognized to be very important when dealing with metaheuristics, only in recent years it has been the object of extensive studies [12, 13, 14, 47, 48, 49, 50]. Some authors adopt a methodology based on factorial design, which is characteristic of a *descriptive* analysis. Therefore, rather than solving directly the tuning problem, they pass through the possibly more complex intermediate problem of understanding the relative importance of each parameter of the algorithm. For example, Xu and Kelly [51] tried to identify the relative contribution of five different components of a tabu-search. Furthermore, the authors considered different values of the parameters of the most effective components and select the best one. Parson and Johnson [52] and Breedam [53] used a similar approach. Xu et al. [54] described a more general technique, which is nonetheless based on factorial analysis. Another approach to tuning that has been adopted for example by Coy et al. [14] and by Adenso-Díaz and Laguna [13] is based on the method that in the statistical literature is known as *response surface methodology*. Bartz-Beielstein and Markon [48] proposed a method to determine relevant parameter settings. It

is based on statistical design of experiments, classical regression analysis, tree based regression and design and analysis of computer experiments (a.k.a. DACE) models. Hutter et al. [49] provided methods for optimizing a target algorithms performance on a given class of problem instances by varying a set of ordinal and/or categorical parameters. The authors exploited a family of local-search-based algorithm configuration procedures and presented novel techniques for accelerating them by adaptively limiting the time spent for evaluating individual configurations. Some procedures for tackling the tuning problem have been proposed by Birattari [12]. Among them, the F-Race method is the best performing one and has been used in a number of works on metaheuristics [55, 56, 57, 58, 59].

For the sake of completeness, we mention here another approach to tuning that goes under the name of *on-line tuning*. The key idea behind this second family of techniques is to modify some parameters of the search algorithm while performing the search itself. This approach is particularly appealing when one is supposed to solve one single instance, typically large and complex. One of the most influential descriptions of *on-line* adjustment of the parameters of an algorithm has been given by Battiti and Tecchiolli [60]. The authors introduced a tabu search where the length of the tabu list is optimized on-line.

3 Main Elements of the Analysis

This section provides details on the three main elements of the case study considered in the work. In particular, Section 3.1 describes the algorithmic framework of the vehicle routing problem with stochastic demand and the local search procedures that we consider. Section 3.2 describes the specific implementations of the five metaheuristics under analysis. Section 3.3 describes F-Race, that is, the tuning algorithm that is used for fine-tuning the metaheuristics in our study.

3.1 The Problem

The vehicle routing problem (VRP) consists in finding the set of tours of minimum cost for visiting a given number of customers exactly once, starting and ending each tour at the depot. A fleet of identical vehicles with finite-capacity is to be used for delivering goods to customers, each having a predefined demand. Moving from a customer to another has a cost that is known *a priori*. Typically, problem instances are represented on graphs in which nodes correspond to customers and a cost is associated to each edge.

The VRP can be formulated using the following notation: Let $G = (V,E)$ be a complete undirected graph, with $V = \{0,...,n\}$ set of nodes. Each node $i \in V \setminus \{0\}$ represents a customer having a nonnegative demand q_i. Node 0 corresponds to the depot. A travel cost c_{ij} is associated to each edge $(i, j) \in E$. Let k be the number of identical vehicles available, each with capacity Q. Let $r(S)$ denote the minimum number of vehicles needed to serve the customers of a subset S of customers. A lower bound of $r(S)$ is $\lceil \sum_{i \in S} q_i/Q \rceil$. For each $s \subset V$ let $\delta(S) = \{(i, j) : i \in S, j \notin$

$S7 \vee i \notin S, j \in S\}$, i.e. the set of edges connecting a node belonging to S with one belonging to $V \setminus S$. Let x_{ij} be the integer variable indicating the number of times edge (i, j) is traversed in the solution. The formulation proposed by Laporte et al. [61] is then:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{(i,j)\in\delta(\{i\})} x_{ij} = 1, \qquad \forall i \in V \setminus \{0\}, \tag{2}$$

$$\sum_{(i,j)\in\delta(\{0\})} x_{ij} = 2k, \tag{3}$$

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq 2r(S), \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{4}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \notin \delta(\{0\}), \tag{5}$$

$$x_{ij} \in \{0,1,2\}, \qquad \forall (i,j) \in \delta(\{0\}). \tag{6}$$

$$\tag{7}$$

Constraints (2) impose that each customer is visited exactly once. Constraint (3) states that k routes are created. Capacity constraints (4) ensure both connectivity of the tours and respect of vehicles capacity. This is done by imposing a sufficient number of edges to enter each subset of nodes S. Constraints (5) and (6) imply that each edge connecting two customers is traversed at most once, while each edge connecting a customer to the depot is traversed at most twice.

In the vehicle routing problem with stochastic demand, the quantities demanded by customers are not known *a priori* but their probability distributions are given.

As in most of the previously published works on VRPSD [9, 20, 21, 22], in this work the problem is addressed by considering only one vehicle. This element was proved to give the best solution in absence of additional constraints [22]. The solution technique consists in constructing an *a priori* TSP-wise tour. This tour is then split according to the specific realizations of the random variables representing the demand of the customers. The objective is finding the *a priori* tour with minimum expected cost.

The computation of the expected cost of the solutions follows Yang et al. [22] and Bianchi et al. [9]. In particular, it is based on a dynamic programming recursion that moves backward from the last node of the sequence. At each node, the decision of restocking or proceeding is based on the expected cost-to-go in the two cases.

In this analysis, we consider the two local search procedures that can be found in the VRPSD literature: Or-opt and 3-opt [9, 22]. Different methods are used for computing the cost of a move in the local search. In this way, five procedures are obtained: Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost). For a detailed description of these techniques we refer the reader to Bianchi et al. [9].

In order to reach some significant conclusion with our empirical analysis, a rather large set of instances is needed. The set of instances considered in Bianchi et al. [9] is too small for the aim of our research. To the best of our knowledge, these are the only benchmark instances available for the vehicle routing problem with stochastic demand. For our experiments, we use instances created with the instance generator described in Pellegrini and Birattari [62]. We consider instances with either 50 or 60 nodes.

Following [9], we consider instances in which the demand of each customer is uniformly distributed. The average and the spread of these distributions are reported in Table 1.

Table 1 Parameters used for generating the instances. $U(min, max)$ means that the value is randomly extracted from a uniform distribution in the range between *min* to *max*.

Instance class	average demand	spread
I	$U(20,30)$	$U(5,10)$
II	$U(20,30)$	$U(5,15)$
III	$U(20,35)$	$U(5,10)$
IV	$U(20,35)$	$U(5,15)$

The capacity of the vehicle is 80. In this way the average number of customers that can be served before returning to the depot is about three. Analysis of cases with such a low ratio between capacity of the vehicle and average customer demand are not very frequent in the literature. On the other hand it is a situation that can be easily encountered in reality. A previous study of the performance of algorithms when tackling instances with this peculiarity can be found in Bianchi et al. [9]. Being this paper the main reference for our work, we decided to use instances generated according to the ratio used there.

3.2 Metaheuristics

The implementation of the metaheuristics we consider is based on the code written by Bianchi et al. [9][3]. In the following, we give a short description of the main element characterizing each algorithm. The parameters of the algorithms are briefly explained. As a reference algorithm, following Bianchi et al. [9], we considered a random restart local search (RR). It uses the randomized furthest insertion heuristic plus local search. It restarts every time a local optimum is found, until the stopping criterion is met—in our case, the elapsing of a fixed computational time.

In the **tabu search**, the tabu-list stores partial solutions. An *aspiration criterion* allows forbidden moves if the new solution is the new best one. The *tabu tenure*, that is, the length of the tabu list, is variable [9]: At each step it assumes a random value between $t(m-1)$ and $m-1$, where $0 \leq t \leq 1$ is a parameter of the algorithm. When

[3] Available at `http://iridia.ulb.ac.be/vrpsd.ppsn8`.

3-opt is used, m is equal to the number of customers. When Or-opt is used, m is equal to the number of customers minus the length of the string of consecutive nodes that are shifted in a move. During the exploration of the neighborhood, solutions that include forbidden components are evaluated with probability p_f and the others with probability p_a. The difference between the EXACT-cost, the VRPSD-cost, and the TSP-cost implementations concerns only the local search procedure.

Concerning the **simulated annealing**, the probabilistic acceptance criterion consists in accepting a solution s' either if it has a lower cost than the current solution s or, independently of its cost, with probability

$$p(s'|T_k, s) = exp\left(-\frac{Cost(s')Cost(s)}{T_k}\right). \tag{8}$$

The relevant parameters of the algorithm are related to the initial level of the *temperature* and to its evolution. The starting value T_0 is determined by considering one hundred solutions randomly chosen in the neighborhood of the first one, by computing the variation of the cost in this set, and by multiplying this result for the parameter f. At every iteration k, the *temperature* is decreased according to the formula $T_k = \alpha T_{k-1}$, where the parameter α, usually called *cooling rate*, is such that $0 < \alpha < 1$. If after $n \cdot q \cdot r$ iterations the quality of the best solution is not improved, the process known as *re-heating* [32] is applied: the temperature is increased by adding T_0 to the current temperature. Besides the local search procedure used, the difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations consists in the way $Cost(s')$ and $Cost(s)$ in Equation 8 are computed. In the TSP-cost, only the length of the *a priori* tour is considered.

In the implementation of the **genetic algorithm**, edge recombination [63] consists in generating a tour starting from two solutions by using edges present in both of them, whenever possible. Mutation swaps adjacent customers with probability p_m. If mutation is *adaptive*, p_m is equal to the product of the parameter *mr* (*mutation-rate*) and a similarity factor. The latter depends on the number of times the n-th element of the first parent is equal to the n-th element of the second one. If the mutation is not *adaptive*, p_m is simply equal to *mr*. The difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations concerns only the local search procedure adopted.

The **iterated local search** is characterized by a function that performs a perturbation on solutions. It returns a new solution obtained after a loop of n random moves (with n number of nodes of the graph) of a 2-exchange neighborhood. They consist in subtour inversions between two randomly chosen nodes. The loop is broken if a solution with quality comparable to the current one is found. We say that the quality of a solution is comparable to the quality of the current one if its objective function value is not greater than the objective function value of the current solution plus a certain value ε. The difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations concerns only the local search procedure adopted.

In this implementation of **ant colony optimization**, the pheromone trail is initialized to $\tau_0 = 0.5$ on every arc. The first population of solutions is generated and refined via the local search. Then, a *global pheromone update* is performed r times. At each following iteration, p new solutions are constructed by p artificial ants on the basis of the information stored in the pheromone matrix. After each step, the *local pheromone update* is performed on the arc just included in the route. Finally, the local search is applied to the p solutions and the *global pheromone update* is executed. Local and global pheromone updates are performed as follows:

Local pheromone update: the pheromone trail on the arc (i, j) is modified according to the following formula:

$$\tau_{ij} = (1 - \psi)\tau_{ij} + \psi\tau_0,$$

with ψ parameter such that $0 < \psi < 1$.

Global pheromone update: the pheromone trail on each arc (i, j) is modified according to the following formula:

$$\tau_{ij} = (1 - \rho)\tau + \rho\Delta\tau_{ij}^{bs}$$

where

$$\Delta\tau_{ij}^{bs} = \begin{cases} Q/Cost\_Solution\_bs & \text{if arc } (i,j) \in Solution\_bs \\ 0 & \text{otherwise,} \end{cases}$$

ρ is a parameter such that $0 < \rho < 1$ and *Solution_bs* is the best solution found so far.

3.3 The Tuning Process

The parameters of all algorithms considered in the study are tuned through the F-Race procedure [10, 11, 12]. F-Race is a racing algorithm for choosing a candidate configuration, that is, a combination of values of the parameters, out of predefined ranges.

F-race runs the optimization algorithm multiple times testing on several instances a given set of candidate configurations. On the basis of the results achieved, a configuration can be discarded if it appears suboptimal: For each instance (each representing one step of the race) the ranking of the results obtained using the different configurations is computed and a statistical test is performed for deciding whether to discard some candidates or not. The set of configurations considered at a specific step h contains all the candidates that survived after step $h - 1$ (Figure 1). F-Race is based on the Friedman two-way analysis of variance by ranks [64]. An important advantage offered by this statistical test is connected with the nonparametric nature of a test based on ranking, which does not require to formulate hypothesis on the distribution of the observations.

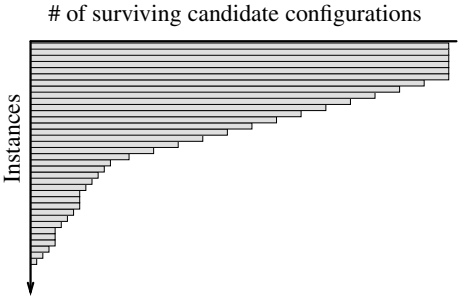


Fig. 1 Graphical representation of the computation performed by the racing approach. As the evaluation proceeds, the racing algorithm focuses more and more on the most promising candidates, discarding a configuration, as soon as sufficient evidence is gathered that it is suboptimal [11, 12].

4 Experimental Analysis

The main goal of the computational experiments proposed in this section is to show that a remarkable difference exists between the results obtained by *out-of-the-box* and *custom* versions of the metaheuristics. Moreover, we wish to study the performance of a *literature* version of the same approaches, that is, a version that reproduces as precisely as possible implementation described in the literature. We aim at showing that by using published versions that have been optimized for solving possibly different problem instances, the results that one might obtain are not necessarily state-of-the-art. In particular, they might significantly differ from those that can be obtained through *custom* implementations specifically tailored to the class of problem instances at hand.

As we mentioned in the introduction, the various versions differ one from the other in the values of the parameters. In the *literature* versions, the values of the parameters are those proposed by Bianchi et al. [9]. In the *custom* versions, the parameters are accurately fine-tuned with the F-Race automatic procedure. As mentioned in Section 3.3, F-Race selects the best values of the parameters out of a given set of candidate ones. Finally, in the *out-of-the-box* versions, the values of the parameters are randomly drawn from the same set of candidate values that is considered by F-Race for *custom* versions. Equal probability has been associated to each configuration and, for each instance considered in the analysis, a random selection has been performed. The choice of randomly drawing the values of the parameters is motivated by the observation that an *out-of-the-box* implementation is often based on an experience–based selection: Given a set of reasonable values, one makes decisions that will end up being more or less "lucky". In order to prescind from our fortune while testing the thesis at the basis of this work, a sort of average performance is sought. Such a selection criterion is anyway much different from the random picking of admissible values for the parameters: Following our experience in the field,

if we did not have the possibility of tuning parameters, we could have reasonably chosen any of the available combinations for running experiments.

For each of the metaheuristics, besides the methods used for setting the parameters, the implementations considered in the *out-of-the-box*, *custom*, and *literature* versions are identical.

The values of the parameters represent only one of the elements that may be customized when implementing a metaheuristic. By considering only this element in our study, we somehow underestimate the difference between *out-of-the-box* and *custom* implementations. Nonetheless, if we succeed to show that the results of an analysis performed on *out-of-the-box* implementations cannot be generalized to *custom* implementations when the difference between the two simply consists in the values of the parameters, we have reached our goal: Any other element that can be fine-tuned and customized would simply further reduce the possibility of generalizing results observed in one context to the other.

All experiments are run on a cluster of AMD Opteron^{TM} 244, and 1000 instances are considered. We run each algorithm once on each instance [65]. The computation time is used as a stopping criterion for all the algorithms and it is set to 30 seconds.

In order to obtain the *custom* versions of the metaheuristics through F-Race, a number of different configurations ranging from 1200 to about 1600 were considered for each of them. Table 2 reports, for each metaheuristic, the parameters considered for optimization, the range of values allowed, and the values selected. A set of 500 instances of the vehicle routing problem with stochastic demand was available for the tuning. These instances have the same characteristics of the ones used for the experimental analysis, but the two sets of instances are disjoint [5]. While tuning a metaheuristic, the F-Race procedure was allowed to run the metaheuristic under consideration for a maximum number of times equal to 15 times the number of configurations considered for that metaheuristic. Also for the random restart local search, a *custom* version has been considered. It has been obtained by selecting, through the F-Race procedure, the best performing local search. In other words, the parameter that has been optimized in this case is the underlying local search.

A first analysis of the performance of the algorithms in the two contexts, *custom* and *out-of-the-box*, consists in comparing the results achieved in terms of cost of the best solution returned. Figure 2 reports the distributions of the difference between the costs of the solutions obtained in the two contexts by each metaheuristic. To be precise, we report the distribution of the cost of the solutions found by each *custom* version minus the one of its *out-of-the-box* counterpart. In Figure 2(a), the whole distributions are shown. In Figure 2(b), the detail of the area around zero is reported. We can observe that, even if the tails of the distributions are sometimes very long[4], almost 75% of the observations fall below the zero line for all metaheuristics. This means that, in the strong majority of the cases, the difference is in favor of the *custom* version. Again, we can observe that some metaheuristics are more sensitive

[4] The long tails of the distributions do not appear to be dependent on any specific aspect of the instances solved. In the same way, it is not possible to find a correlation between the parameter configurations use in the *out-of-the-box* versions and the presence of outliers.

Table 2 Range of values considered for the parameters of the metaheuristics. The values reported in bold are the ones selected by F-Race for the *custom* versions.

Tabu search – total number of candidates = 1460	
parameter	range
p_f	0.1, **0.2**, 0.25, 0.3, 0.35, 0.4
p_a	0.5, **0.6**, 0.7, 0.75, 0.8, 0.85, 0.9
t	**0.3**, 0.4, 0.5, 0.7, 0.8, 0.9, 1
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), **3-opt(EXACT-cost)**

Simulated annealing – total number of candidates = 1200	
parameter	range
α	**0.3**, 0.5, 0.7, 0.9, 0.98
q	**1**, 5, 10
r	10, 20, 30, **40**
f	0.01, **0.03**, 0.05, 0.07
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

Genetic algorithm – total number of candidates = 1360	
parameter	range
pop. size	10, 12, 14, 16, 18, **20**, 22, 24
mr	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65,**0.7**, 0.75, 0.8, 0.85, 0.9
adaptive	**Yes**, No
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

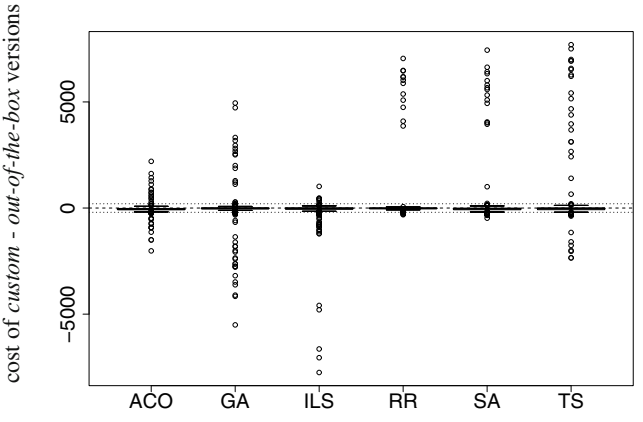
Iterated local search – total number of candidates = 1520	
parameter	range
ε	$n/x, x \in \{0.005,0.01,0.05,0.1,0.5, 1.0,$**1.5**$,2.0,$ all multiples of 0.5 up to 150.0$\}$
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

Ant colony optimization – total number of candidates = 1620	
parameter	range
p	**5**,10, 20
ρ	0.1, 0.5, **0.7**
r	100, **150**, 200
Q	$10^5, 10^6, 10^7, \mathbf{10^8}, 10^9$
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), **3-opt(TSP-cost)**, 3-opt(EXACT-cost)

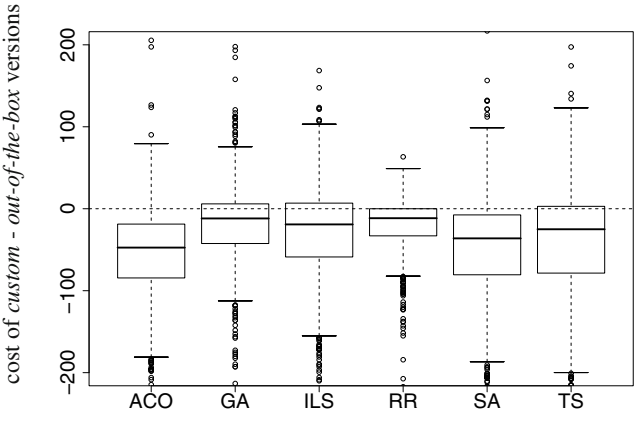
Random restart – total number of candidates = 5	
parameter	range
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), **3-opt(EXACT-cost)**

to the value of their parameters and therefore benefit more than others from an accurate fine-tuning. Observing these results, it is immediately clear that, as expected, the performance achieved by algorithms depend strongly on the values chosen for the parameters, and then on the contexts considered.

Some further observations can be made considering the distribution of the ranking achieved by each algorithm. These results are reported in Figures 3(a), 3(b), and 3(c), for the *custom*, *out-of-the-box*, and *literature* versions respectively. On the left of each graph, the names of the algorithms are given. The order in which they appear reflects the average ranking: The lower the average ranking, the better the general behavior, and the higher the metaheuristic appears in the list. On the right, the boxplots represent the distributions of the ranks over the 1000 instances. Between the names and the boxplots, vertical lines indicate if the difference in the behavior of the



(a)



(b)

Fig. 2 Difference between the costs of the solutions obtained by the *custom* and the *out-of-the-box* versions of the metaheuristics under analysis. In Figure 2(a), the entire distribution is shown for each metaheuristic. Since the distributions are characterized by long tails, in Figure 2(b) the detail of the more interesting central area is given. For all metaheuristics, the median of the distribution is below the zero: Being the VRPSD a minimization problem, the results obtained by the *custom* versions are in general better than those obtained by their *out-of-the-box* counterpart.

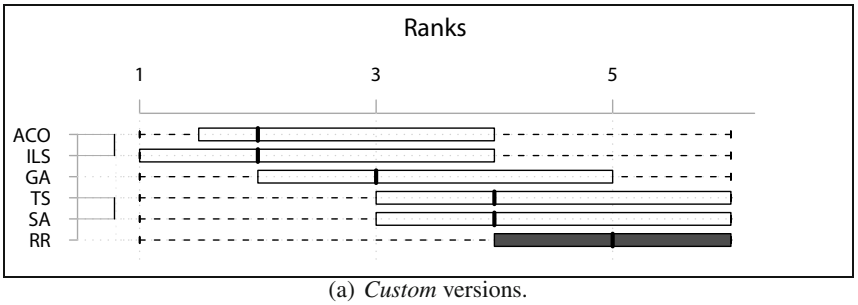
metaheuristics is significant according to the Friedman test: If two metaheuristics are not comprised by the same vertical line, their behavior is significantly different according to the statistical test considered, with a confidence of 95%. The difference in the denomination of the algorithms between the first two figures and the third one depends on the fact that in Bianchi et al. [9], the local search procedure is not considered as a parameter of the algorithms. For this reason, the metaheuristics are presented in Figure 3(c) with the name of the metaheuristic paired with the one of the variant of the local search used. In Bianchi et al. [9], the 3-opt local search has been used only in association with iterated local search and genetic algorithms.

As it can be observed, the ranking of algorithms varies in the three contexts. First of all, let us focus on the graphics representing the *out-of-the-box* and the *custom* versions. The two main differences concern RR and ACO. The former performs the worst in the *custom* context, while this is not the case in the *out-of-the-box* context. The case of a metaheuristic performing worse than the random restart local search is to be considered as a major failure for the metaheuristic itself. We consider this point as a remarkable difference between the two contexts: In the *out-of-the-box* context, three out of five metaheuristics perform significantly worse than the random restart local search; in the *custom* context, all metaheuristics achieve better results than the random restart local search.

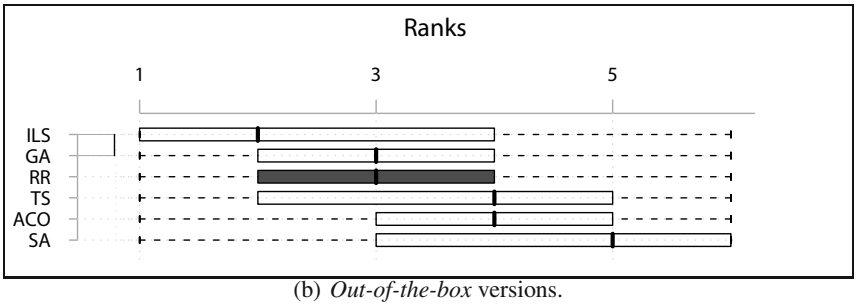
As far as ACO is concerned, we can observe that the relative performance is visibly different in the two contexts. The *out-of-the-box* version behaves significantly worse than RR, and is among the worst in the set. On the contrary, the *custom* version achieves the best average ranking. This difference shows that this metaheuristic is more sensitive than the others to variations of the parameters, possibly due to the large number of parameters of the algorithm. This might be seen as a drawback of ACO. Anyway, we think that this fact should be read in a different way: If one is interested in an *out-of-the-box* metaheuristics, a high sensitivity to the parameters is definitely an issue; on the other hand, if one wishes to implement a *custom* metaheuristic, the sensitivity is an opportunity that can be exploited in order to finely adapt the algorithm to the class of instances to be tackled.

Let us consider now Figure 3(c), where the performance of the *literature* version is reported. As it can be noted by comparing this graph with Figures 3(a) and 3(b), the general trend is very similar to the one obtained in the *out-of-the-box* context.

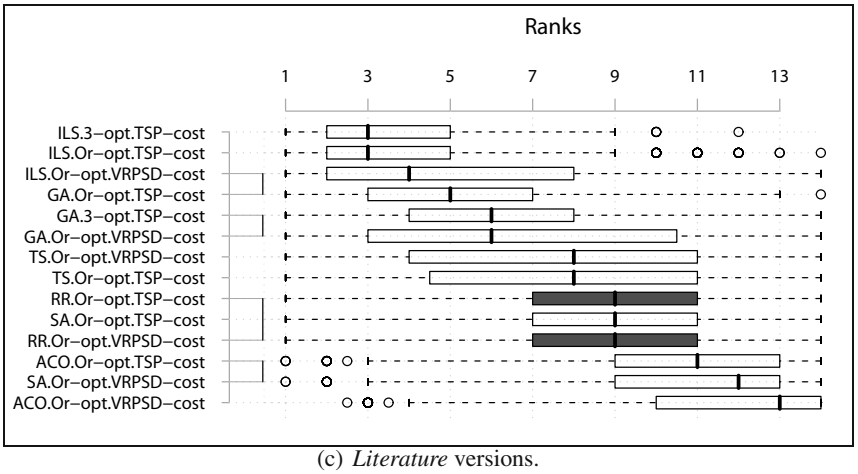
In order to provide a more precise picture of the sensitivity of each metaheuristics to its parameters, Figure 4 reports, for each metaheuristic, the comparison of the results obtained in the three contexts. What clearly emerges is that, as expected, all metaheuristics achieve the best results in their *custom* version. The difference is always statistically significant according to the Friedman test. Moreover, it can be observed that, less expectedly, *literature* versions, that is, those in which the parameters are set according to Bianchi et al. [9], obtain results that are comparable with those of the *out-of-the-box* versions. In particular, while for iterated local search and tabu search the values reported in the literature appear to be better than those drawn at random, for genetic algorithms and simulated annealing this is not always the case. Even more striking, in the case of ant colony optimization, the parameters used in Bianchi et al. [9] yield results that are significantly worse than those drawn



(a) *Custom* versions.



(b) *Out-of-the-box* versions.



(c) *Literature* versions.

Fig. 3 Results over 1000 instances of the metaheuristics in the three variants considered. Two main observations can be made: In the *out-of-the-box* and *literature* versions we can see that some metaheuristics are outperformed by the random restart local search. This represents a major failure. In the *custom* versions all metaheuristic achieve better results. Moreover, the relative behavior of metaheuristics changes. For example, ACO ranks first in Figure 3(a), the fifth in Figure 3(b), and last in Figure 3(c).

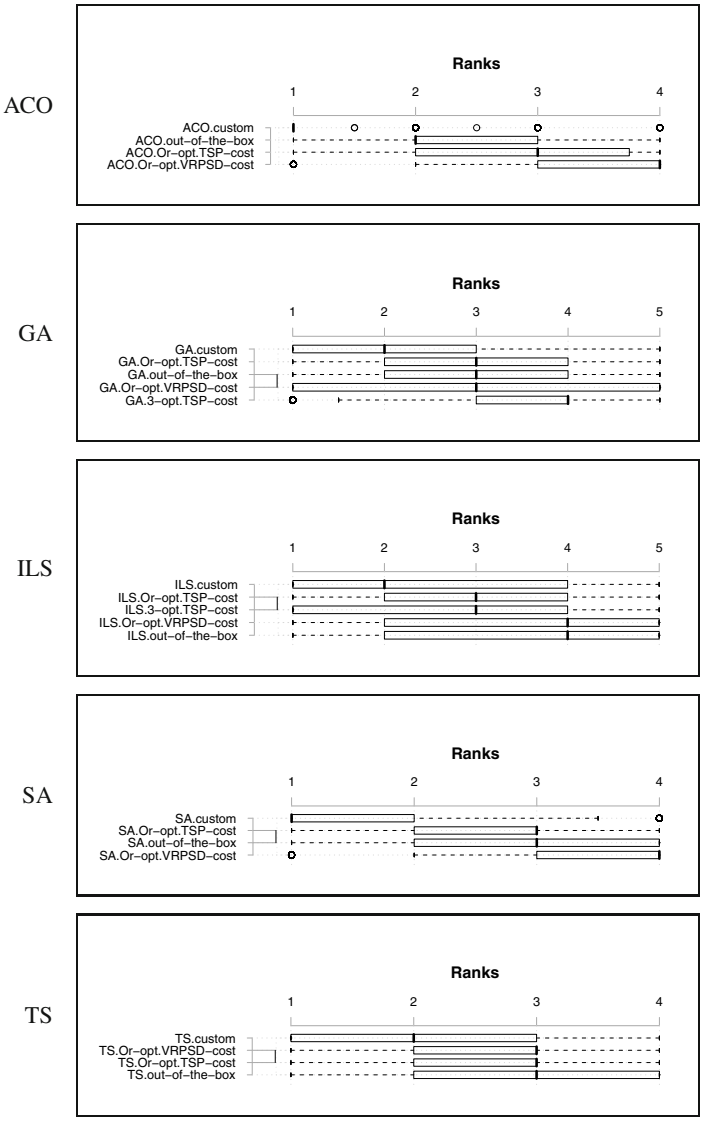


Fig. 4 Comparison of the results obtained in the three sets of experiments. As it can be observed, the *custom* versions are always significantly better than all the others. For iterated local search and tabu search, *literature* versions are better than their *out-of-the-box* counterparts, that is, the values chosen by Bianchi et al. [9] behave better than the random ones. On the contrary, the *out-of-the-box* ant colony optimization works better than the *literature* version. Finally, in genetic algorithms and simulated annealing the results are mixed: the *out-of-the-box* versions is better than one of the two *literature* versions and worse that the other one, or of the other two in the case of genetic algorithms for which three versions where proposed in Bianchi et al. [9].

at random. These results clearly support our claim according to which there is a strong difference between the performance of metaheuristics used *out-of-the-box* or in a *custom* way. Moreover, they show that the versions that can be found in the literature are not necessarily state-of-the-art, when applied to problem instances that differ from those considered in the original study.

5 Conclusions

In this chapter, five of the most successful metaheuristics, namely tabu search, simulated annealing, genetic algorithm, iterated local search, and ant colony optimization, have been compared on the vehicle routing problem with stochastic demand. These five metaheuristics applied to the vehicle routing problem with stochastic demand have been the focus of a research published by Bianchi et al. [9].

Our goal is to highlight that results obtained with *out-of-the-box* versions of metaheuristics cannot be directly generalized to *custom* versions. In our analysis, what differentiates a *custom* version of a metaheuristic from the corresponding *out-of-the-box* one, is that the parameters of the former are fine-tuned through the F-Race algorithm, while those of the latter are drawn at random.

As it could be expected, the empirical results show that the *custom* version of each metaheuristic achieves better results than the corresponding *out-of-the-box* one. The difference is always statistically significant according to the Friedman test. Moreover, the relative performance of algorithms differs greatly in the two contexts. This can be ascribed to the fact that different metaheuristics might be more or less sensitive to variations of their parameters.

A second element on which the analysis focuses is whether the results that are reported in the literature can be *a priori* associated with to one of the two contexts. From our experiments it appears clear that this is not the case.

On the basis of this case study, we can conclude that there may be a strong difference in the results achievable by using the *out-of-the-box* or the *custom* version of metaheuristics. This difference may concern both the quality of the solutions returned by an approach, and the relative performance of algorithms. As a consequence, one should clearly describe the implementation criteria followed in the design of an algorithm, in order to allow the readers to focus on the more suitable implementations, given their specific goals.

The lack of this piece of information cannot be filled by considering all the implementation studied in the literature as custom: as we show, they may refer alternatively to either context. This element confirms the relevance of our research. In this precise sense, the analysis presented in this work is strongly related to a subject that has an actual impact on the current research in the field of metaheuristics.

Acknowledgments. This work was partially supported by the ARC projects ANTS and META-X, funded by the Scientific Research Directorate of the French Community of Belgium. Mauro Birattari acknowledges support from the fund for scientific research F.R.S.-FNRS of Belgium's French Community, of which he is a research associate.

References

[1] Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13, 533–549 (1986)
[2] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
[3] Barr, R.S., Kelly, J.P., Resende, M.G.C., Stewart, W.R.: Designing and reporting computational experiments with heuristic methods. Journal of Heuristics 1(1), 9–32 (1995)
[4] Birattari, M., Dorigo, M.: How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? Optimization Letters 1(3), 309–311 (2006)
[5] Birattari, M., Zlochin, M., Dorigo, M.: Towards a theory of practice in metaheuristics design: A machine learning perspective. Theoretical Informatics and Applications 40(2), 353–369 (2006)
[6] Eiben, A.E., Jelasity, M.: A critical note on experimental research methodology in EC. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 582–587. IEEE Press, Los Alamitos (2002)
[7] Hooker, J.N.: Testing heuristics: We have it all wrong. Journal of Heuristics 1, 33–42 (1995)
[8] Pellegrini, P., Birattari, M.: Implementation effort and performance. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 31–45. Springer, Heidelberg (2007)
[9] Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T.: Hybrid metaheuristics for the vehicle routing problem with stochastic demands. Journal of Mathematical Modelling and Algorithms 5(1), 91–110 (2006)
[10] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
[11] Birattari, M.: The problem of tuning metaheuristics as seen from a machine learning perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2005)
[12] Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. Springer, Berlin (2009)
[13] Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)
[14] Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 7(1), 77–97 (2001)
[15] Tillman, F.: The multiple terminal delivery problem with probabilistic demands. Transportation Science 3, 192–204 (1969)
[16] Stewart, W., Golden, B.: Stochastic vehicle routing: a comprehensive approach. European Journal of Operational Research 14, 371–385 (1983)
[17] Dror, M., Trudeau, P.: Stochastic vehicle routing with modified saving algorithm. European Journal of Operational Research 23, 228–235 (1986)
[18] Laporte, G., Louveau, F.: Formulations and bounds for the stochastic capacitated vehicle routing problem with uncertain supplies. Technical Report G-87-23, Ecole des Hautes Etudes Commerciale, University of Montreal, Montreal, Canada (1987)
[19] Laporte, G., Louveau, F., Mercure, H.: Models and exact solutions for a class of stochastic location-routing problems. Technical Report G-87-14, Ecole des Hautes Etudes Commerciale, University of Montreal, Montreal, Canada (1987)

[20] Bertsimas, D.J.: A vehicle routing problem with stochastic demand. Operations Research 40(3), 574–585 (1992)
[21] Bertsimas, D.J., Simchi-Levi, D.: A new generation of vehicle routing research: robust algorithms, addressing uncertainty. Operations Research 44(3), 286–304 (1996)
[22] Yang, W.H., Mathur, K., Ballou, R.H.: Stochastic vehicle routing problem with restocking. Transportation Science 34(1), 99–112 (2000)
[23] Secomandi, N.: Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers & Operations Research 27, 1201–1225 (2000)
[24] Secomandi, N.: A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49, 796–802 (2001)
[25] Secomandi, N.: Analysis of a rollout approach to sequencing problems with stochastic routing applications. Journal of Heuristics 9, 321–352 (2003)
[26] Teodorović, D., Pavković, G.: A simulated annealing technique approach to the VRP in the case of stochastic demand. Transportation Planning and Technology 16, 261–273 (1992)
[27] Gendreau, M., Laporte, G., Séguin, R.: A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Working paper, CRT, University of Montreal, Montreal, Canada (1994)
[28] Or, I.: Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. PhD thesis, Northwestern University, Evanston, IL, USA (1976)
[29] Lin, S.: Computer solutions of the traveling salesman problem. Bell System Tech. Journal 44, 2245–2269 (1965)
[30] Glover, F.: Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166 (1977)
[31] Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
[32] Aarts, E.H.L., Korst, J.H.M., van Laarhoven, P.J.M.: Simulated annealing. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 91–120. John Wiley & Sons, Inc., New York (1997)
[33] Cerny, V.: A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 45, 41–51 (1985)
[34] Fleischer, M.: Simulated annealing: past, present and future. In: Lilegdon, W.R., Alexopoulos, C.L., Kang, K., Goldsam, G. (eds.) Proceedings of the 1995 Winter Simulation Conference, pp. 155–161 (1995)
[35] Ingber, L.: Adaptive simulated annealing (ASA): lessons learned. Control and Cybernetics 26(1), 33–54 (1996)
[36] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
[37] Darwin, C.R.: On the Origin of Species by Means of Natural Selection. Or the preservation of favoured races in the struggle for life. John Murray, London (1859)
[38] Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol (1997)
[39] Fogel, L.J.: Toward inductive inference automata. In: Proceedings of the International Federation for Information Processing Congress, Munich, Germany, pp. 395–399 (1962)
[40] Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligent through Simulated Evolution. John Wiley & Sons, New York (1966)

[41] Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading (1989)
[42] Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975)
[43] Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
[44] Laurenço, H.R., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 321–353. Kluwer Academic Publishers, Norwell (2002)
[45] Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1(4), 28–39 (2006)
[46] Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research 131(1-4), 373–395 (2004)
[47] Bartz-Beielstein, T., Preuss, M., Reinholz, A.: Evolutionary algorithms for optimization practitioners. Technical Report CI-151/03, Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence, Universität Dortmund, Dortmund, Germany (2003)
[48] Bartz-Beielstein, T., Markon, S.: Tuning search algorithms for real-world applications: A regression tree based approach. In: Greenwood, G.W. (ed.) Proc. 2004 Congress on Evolutionary Computation (CEC 2004), pp. 1111–1118. IEEE Press, Piscataway (2004)
[49] Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
[50] Favaretto, D., Moretti, E., Pellegrini, P.: On the explorative behavior of MAX–MIN ant system. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2009. LNCS, vol. 5752, pp. 115–119. Springer, Heidelberg (2009)
[51] Xu, J., Kelly, J.: A network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science 30, 379–393 (1996)
[52] Parson, R., Johnson, M.: A case study in experimental design applied to genetic algorithms with applications to DNA sequence assembly. American Journal of Mathematical and Management Sciences 17, 369–396 (1997)
[53] Van Breedam, A.: An analysis od the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. Technical Report TR 96/14, Faculty of Applied Economics, University of Antwerp, Antwerp, Belgium (1996)
[54] Xu, J., Chiu, S.Y., Glover, F.: Fine-tuning a tabu search algorithm with statistical tests. International Transactions on Operational Research 5(3), 233–244 (1998)
[55] Chiarandini, M.: Stochastic local search for overconstrained problems. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany (2005)
[56] Chiarandini, M., Stützle, T.: Experimental evaluation of course timetabling algorithms. Technical Report AIDA-02-05, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany (2002)
[57] den Besten, M.L.: Simple metaheuristics for scheduling. An empirical investigation into the application of iterated local search to deterministic scheduling problemns with tardiness penalities. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany (2004)

[58] Schiavinotto, T., Stützle, T.: The linear ordering problem: instances, search space analysis and algorithms. Journal of Mathematical Modelling and Algorithms 3, 367–402 (2004)
[59] Yuan, B., Gallagher, M.: Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tino, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 172–181. Springer, Heidelberg (2004)
[60] Battiti, R., Tecchiolli, G.: The reactive tabu search. ORSA Journal on Computing 6, 126–585 (1994)
[61] Laporte, G., Nobert, Y., Desrochers, M.: Optimal routing under capacity and distance restrictions. Operations Research 33, 1050–1073 (1985)
[62] Pellegrini, P., Birattari, M.: Instances generator for the vehicle routing problem with stochastic demand. Technical Report TR/IRIDIA/2005-10, Iridia, Université Libre de Bruxelles, Brussels, Belgium (2005)
[63] Whitley, D., Starkweather, T., Shaner, D.: The traveling salesman problem and sequence scheduling: quality solutions using genetic edge recombination. In: Davis, L. (ed.) Handbook of Genetic Algorithms, pp. 350–372. Van Nostrand Reinhold, New York (1991)
[64] Friedman, J.H.: Multivariate adaptive regression splines. The Annals of Statistics 19, 1–141 (1991)
[65] Birattari, M.: 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, Iridia, Université Libre de Bruxelles, Brussels, Belgium (2004)