Implementation Effort and Performance

A Comparison of Custom and Out-of-the-Box Metaheuristics on the Vehicle Routing Problem with Stochastic Demand

Paola Pellegrini[1] and Mauro Birattari[2]

[1] Department of Applied Mathematics, Università Ca' Foscari, Venice, Italy
[2] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
paolap@pellegrini.it, mbiro@ulb.ac.be

Abstract. In practical applications, one can take advantage of metaheuristics in different ways: To simplify, we can say that metaheuristics can be either used *out-of-the-box* or a *custom* version can be developed. The former way requires a rather low effort, and in general allows to obtain fairly good results. The latter implies a larger investment in the design, implementation, and fine-tuning, and can often produce state-of-the-art results.

Unfortunately, most of the research works proposing an empirical analysis of metaheuristics do not even try to quantify the development effort devoted to the algorithms under consideration. In other words, they do not make clear whether they considered *out-of-the-box* or *custom* implementations of the metaheuristics under analysis. The lack of this information seriously undermines the generality and utility of these works.

The aim of the paper is to stress that results obtained with *out-of-the-box* implementations cannot be always generalized to *custom* ones, and vice versa. 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 algorithm, 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.

1 Introduction

The term *metaheuristics* [1] recently became widely adopted for designating a class of approaches used for tackling 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.*
>
> [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. A basic implementation of metaheuristic can be obtained

T. Stützle, M. Birattari, and H.H. Hoos (Eds.): SLS 2007, LNCS 4638, pp. 31–45, 2007.

with quite a low effort. Typically, with such an implementation, it is possible to achieve fairly good results. Nonetheless, it has been shown in the literature that metaheuristics can obtain state-of-the-art results with some larger implementation effort. To simplify, we can say that metaheuristics can be either used *out-of-the-box* or a *custom* version can be developed.

This flexibility of metaheuristics is definitely a positive feature: 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. One can then move to a *custom* version for obtaining better performance without having to switch to a completely different technology.

Nonetheless, this flexibility has a downside: The fact that metaheuristics achieve different results according to the development effort can be reason of misunderstanding. In particular, it could well happen that, as we show in the case study proposed in this paper, 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 concerned. In this sense, results obtained with *out-of-the-box* implementations do not always generalize to *custom* ones, and vice versa.

In the literature, this fact is often neglected and the development effort devoted to algorithms is rarely quantified. This can be partially justified by the fact that measuring development effort is not a simple and well-defined task. Nonetheless, without this piece of information, the usefulness of an empirical study is somehow impaired.

With this paper we wish to stress that two experimental studies, one performed in the *out-of-the-box* context, and the other in the *custom* one, may lead to different conclusions. 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. Our goal is to show that the relative performance of the above metaheuristics depends on the implementations considered.

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*,[1] 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 [3].

In our analysis, these implementations are considered as *black-box* metaheuristics: By modifying their parameters, we obtain the *out-of-the-box* and the *custom* versions. The first ones are obtained by randomly drawing the parameters from

[1] http://www.metaheuristics.net/

a defined range. The second ones are obtained by fine-tuning the parameters through an automatic procedure based on the F-Race algorithm [4,5].

Selecting the best values for the parameters, given the class of instances that are to be tackled, is definitely a sort of customization. By using this element as the only difference between *custom* and *out-of-the-box* implementations, we are neglecting the customization of most of the components of metaheuristics. Nonetheless, the goal of the paper 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.

It is interesting to note here that Bartz-Beielstein [6] has already addressed some of the issues discussed in the paper. In particular, he warns against performing a comparative analysis of algorithms that have not been properly fine-tuned. The results we present in the paper confirm the concerns raised in [6]. Nonetheless, our standpoint is somehow different from the one taken in [6]. Indeed, we think that analyzing the performance of *out-of-the-box* versions of metaheuristics can be in itself relevant since in some practical contexts the development of *custom* versions of a metaheuristic is not possible due to time and/or budget constraints.

The rest of the paper 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

The three main topics of interest of our analysis are introduced in this section. We first focus on the vehicle routing problem with stochastic demand. Then we sketch the five metaheuristics considered, and the problem of fine-tuning metaheuristics.

The vehicle routing problem with stochastic demand (VRPSD) 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 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. 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 [7] in 1969. [8], [9] and [10] use techniques from stochastic programming to solve optimally small instances. [11] and [12] propose different heuristics for solving the VRPSD. They consider the construction of an *a priori* TSP-wise tour. This tour is then split according to precise rules. [13] propose a strategy for splitting the *a priori* tour allowing the restocking before a stockout, when this is profitable. Secomandi [14,15] analyzes different possibilities for applying dynamic programming to this problem. [16] and [17] tackle the VRPSD using metaheuristic approaches. In particular, [16] adopt simulated annealing while [17] use tabu search. Finally, an extended analysis on the behavior of different metaheuristics is proposed by [3]. Two classical local search algorithms have been used for the VRPSD: the Or-opt [13] and the 3-opt [3] procedures.

Following [3], 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 consists in the exploration of the solution space via a local search procedure. Non-improving moves are accepted, and a short term memory is used. The latter expedient is introduced in order to avoid sequences of moves that constantly repeat themselves [18].

Simulated annealing takes inspiration from the annealing process in crystals [19]. The search space is explored via a local search procedure. Simulated annealing escapes from local minima by allowing moves to worsening solutions with a probability that decreases in time.

Genetic algorithms are inspired by natural selection. In this metaheuristic, candidate solutions are represented as individuals of a populations that evolve in time under the effect of a number of operators including *crossover* and *mutation*, which mimic the effects of their natural counterparts [20].

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 different solutions obtained via successive perturbations [21].

Ant colony optimization is inspired by the foraging behavior of ants [2]. Solutions are sampled based on a *pheromone* model and are used to modify the model itself biasing the search toward high quality solutions [22].

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 [5]. Only in recent years this problem has been the object of extensive studies [5,23,24,25,26], although it is generally recognized to be very important when dealing with metaheuristics. Some authors adopt a methodology based on factorial design, which is characteristic of a *descriptive* analysis. For example, [27] try to identify the relative contribution of five different components of a tabu-search. Furthermore, the authors consider different values of the parameters of the most effective components and select the best one. [28] and [29] use a similar approach. [30] describe a more general technique, which is nonetheless based on factorial analysis. Another approach to tuning that has been adopted for example by [26] and by [23] is based on the method that in the statistical literature is known as *response surface methodology*. [25] propose a method to determine relevant parameter settings. Some procedures for tackling the tuning problem have been proposed by [5].

3 Main Elements of the Analysis

The three main elements of the case study considered in the paper are presented in this section.

3.1 The Problem

The VRPSD is addressed by considering only one vehicle [3,11,12,13]. An *a priori* TSP-wise tour is constructed and is then split according to the specific realizations of the demand of the customers. The objective is finding the *a priori* tour with minimum expected cost. The computation of the expected cost of solutions 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 [13,3].

Two local search procedures are considered: Or-opt and 3-opt [13,3]. Five methods are used for computing the cost of a move in the local search: 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 [3].

A rather large set of instances is needed in order to reach some significant conclusion with our empirical analysis. The set of instances considered in [3] 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 [31]. We consider instances with either 50 or 60 nodes.

Following [3], we consider instances in which the demand of each customer is uniformly distributed. The average and the spread of these distributions are selected randomly extracted from a uniform distribution in the following ranges: $\{(20, 30), (20, 35)\}$ for the average, and $\{(5, 10), (5, 15)\}$ for the spread. The capacity of the vehicle is 80.

3.2 Metaheuristics

The implementation of the metaheuristics we consider is based on the code written for [3], which is available at `http://iridia.ulb.ac.be/vrpsd.ppsn8`. In the following, we give a short description of the main element characterizing each algorithm. More details can be found in [32]. The parameters of the algorithms are briefly explained. As a reference algorithm, following [3], 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 **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 [3]: 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-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 to 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 **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 $p(s'|T_k, s) = exp\left(Cost(s) - Cost(s')/T_k\right)$. 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* [33] 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 $Cost(s)$ are computed. In the TSP-cost, only the length of the *a priori* tour is considered.

In the implementation of **genetic algorithm**, edge recombination [34] 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.

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.

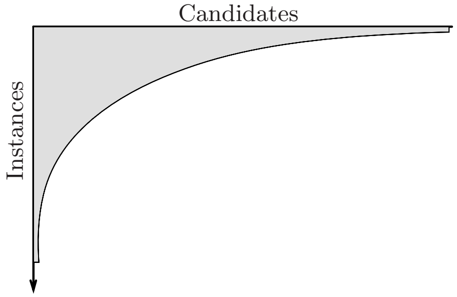


Fig. 1. Graphical representation of computation performed by the racing approach. As the evaluation proceeds, the racing algorithm focuses more and more on the most promising candidates, discarding a configuration, as soon as sufficient evidence is gathered that it is suboptimal [5].

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 pheromone update: the pheromone trail on the arc (i, j) is modified according to $\tau_{ij} = (1 - \psi)\tau_{ij} + \psi\tau_0$, with ψ parameter such that $0 < \psi < 1$.

Global pheromone update: the pheromone trail on each arc (i, j) is modified according to $\tau_{ij} = (1 - \rho)\tau + \rho\Delta\tau_{ij}^{bs}$ where $\Delta\tau_{ij}^{bs} = Q/Cost\_Solution\_bs$ if arc (i, j) belongs to $Solution\_bs$, and $\Delta\tau_{ij}^{bs} = 0$ otherwise. 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 paper are tuned through the F-Race procedure [5,4]. F-Race is a racing algorithm for choosing a candidate configuration, that is, a combination of values of the parameters, out of predefined ranges. A racing algorithm consists in generating a sequence of nested sets of candidate configurations to be considered at each step (Figure 1). The set considered at a specific step h is obtained by possibly discarding from the set considered at step $h - 1$, some configurations that appear to be suboptimal on the basis of the available information. This cumulated knowledge is represented by the behavior of the algorithm for which the tuning is performed, when using different candidates configurations. 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 from the following experiments (in case they appear suboptimal) or not. F-Race is based on the Friedman two-way analysis of variance by ranks [35]. 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.

4 Experimental Analysis

With the computational experiments proposed in this section we wish to show that a remarkable difference exists between the results obtained by *out-of-the-box* and *custom* versions of metaheuristics.

As mentioned in the introduction, the two versions of the metaheuristics are obtained by applying different procedures for setting the values of the parameters. The values of the parameters represent only one of the elements that may be customized when implementing a metaheuristic. In this sense, here the difference between *out-of-the-box* and *custom* implementations may be underestimated. Then, our goal is reached by showing that the results achieved with *out-of-the-box* implementations cannot be generalized to *custom* ones even when the difference between the two simply consists in this element: 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.

In the *custom* versions, the parameters are accurately fine-tuned with the F-Race automatic procedure. 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.

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

All experiments are run on a cluster of AMD Opteron™ 244, and 1000 instances are considered. A computation time of 30 seconds is used as a stopping criterion for all the algorithms.

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 1 reports, for each metaheuristic, the parameters that have been considered for optimization, the range of values allowed, and the values that have been 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 [36]. 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.

Table 1. 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)

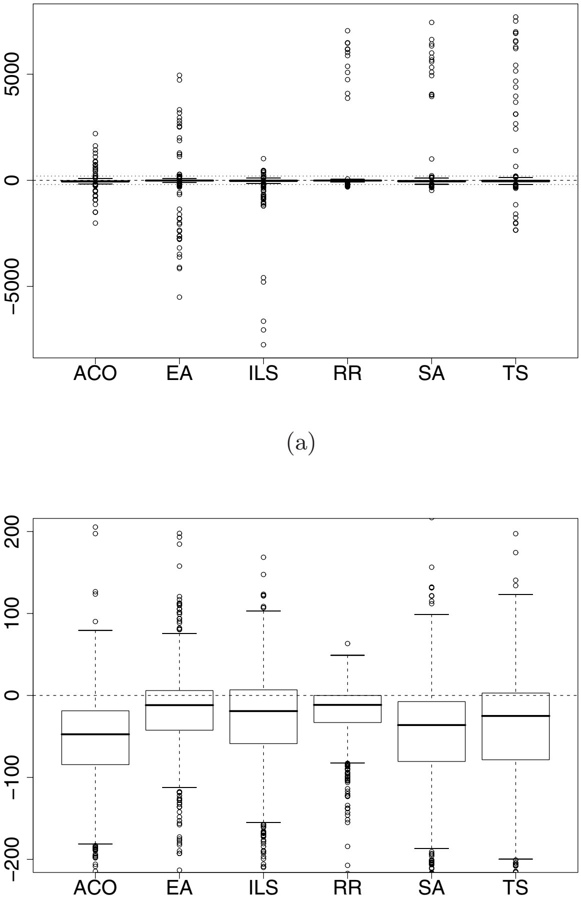
Genetic algorithm – total number of candidates = 1360	
parameter	range
pop. size	10, 12, 14, 16, 18, **20**, 22, 24
mr	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65,**0.7**, 0.75, 0.8, 0.85, 0.9
adaptive	**Yes**, No
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost),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

Iterated local search – total number of candidates = 1520	
parameter	range
ϵ	$n/x, x \in \{0.005, 0.01, 0.05, 0.1, 0.5, 1.0, \mathbf{1.5}, 2.0,$ 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)**

First of all, let us compare the results achieved by the metaheuristics in the two contexts in terms of cost of the best solution returned. The whole distribution of the difference of the results is reported in Figure 2 for each metaheuristic. The detail of the region around 0 is presented in Figure 2(b). The cost of the solutions found by each *custom* version minus the one of its *out-of-the-box* counterpart is considered for each instance. Even if the tails of the distributions are sometimes very long, it can be observed that almost 75% of the observations fall below the zero line for all metaheuristics: the difference is in favor of the *custom* version in the strong majority of the cases. Moreover, it can be noted that, as it can be expected, the various metaheuristics are sensitive in different measure to the value of their parameters. Therefore, they may benefit in different measure from an accurate fine-tuning. Observing these results, it is immediately clear that, as



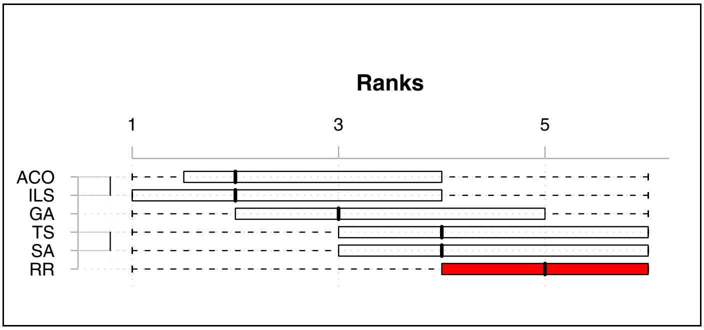
(a)

(b)

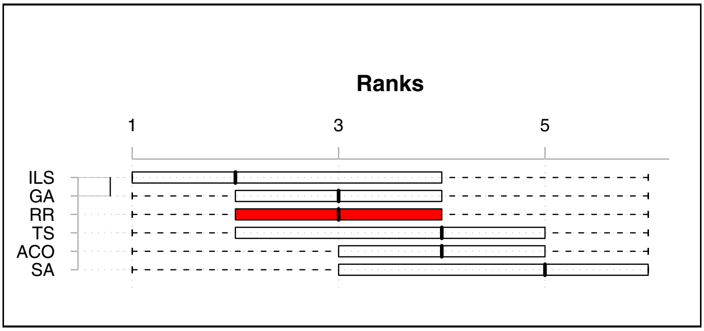
Fig. 2. Difference between the costs of the solutions obtained by the *custom* and the *out-of-the-box* versions of the metaheuristics under analysis. In Figure 2(a), the entire distribution is shown for each metaheuristic. Since the distributions are characterized by long tails, in Figure 2(b) the detail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

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. Figures 3(a) and 3(b) report the results achieved by the *custom* and *out-of-the-box* versions, respectively. On the left of each graph, the names of the algorithms are given. The order in which they



(a) *Custom* versions



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

Fig. 3. Results over 1000 instances of the metaheuristics in the two variants considered

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 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%.

As it can be observed, the ranking of algorithms varies in the two contexts: 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 draw a conclusion that was suggested by the representation proposed in Figure 2 and 2(b). In fact, it 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.

Another point that can be observed concerns the comparison of the ranking obtained by the metaheuristics in the two contexts considered, and the one proposed in [3] on similar instances. Even if, certainly, more experiments are necessary before drawing conclusion, the general trend reported in [3] appears very similar to the one obtained in the *out-of-the-box* context.

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 make us to wonder whether the versions that can be found in the literature are necessarily to be considered *custom*, when applied to problem instances that differ from those considered in the original study.

5 Conclusions

In the paper, 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 and this same optimization problem have been the focus of a research recently published by [3].

Each approach has been considered both in an *out-of-the-box* and in a *custom* version. The procedure used for choosing the values of the parameters is the element that differentiates a *custom* version of a metaheuristic from the corresponding *out-of-the-box* one: In the former, the parameters are fine-tuned through the F-Race algorithm. In the latter, they are drawn at random. Our goal is to highlight that results obtained in one context cannot be directly generalized to the other.

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.

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 experimental analysis raises doubts on the possibility of *a priori* ascribing the results that are reported in the literature to one of the two contexts. The computations reported suggest that this is not the case. A further analysis needs to be devoted to this point.

Acknowledgments. This work was supported by the ANTS project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium.

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, MA, USA (2004)
3. 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)
4. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W. (ed.) GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
5. Birattari, M.: The problem of tuning metaheuristics as seen from a machine learning perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2005)
6. Bartz-Beielstein, T.: Experimental analysis of evolution strategies - overview and comprehensive introduction. Technical Report CI-157/03, Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence, Universität Dortmund, Dortmund, Germany (2003)
7. Tillman, F.: The multiple terminal delivery problem with probabilistic demands. Transportation Science 3, 192–204 (1969)
8. Stewart, W., Golden, B.: Stochastic vehicle routing: a comprehensive approach. European Journal of Operational Research 14, 371–385 (1983)
9. Dror, M., Trudeau, P.: Stochastic vehicle routing with modified saving algorithm. European Journal of Operational Research 23, 228–235 (1986)
10. 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)
11. Bertsimas, D.: A vehicle routing problem with stochastic demand. Operations Research 40(3), 574–585 (1992)

12. Bertsimas, D., Simchi-Levi, D.: A new generation of vehicle routing research: robust algorithms, addressing uncertainty. Operations Research 44(3), 286–304 (1996)
13. Yang, W., Mathur, K., Ballou, R.: Stochastic vehicle routing problem with restocking. Transportation Science 34(1), 99–112 (2000)
14. Secomandi, N.: A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49, 796–802 (2001)
15. Secomandi, N.: Analysis of a rollout approach to sequencing problems with stochastic routing applications. Journal of Heuristics 9, 321–352 (2003)
16. 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)
17. 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)
18. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell, MA, USA (1997)
19. Ingber, L.: Adaptive simulated annealing (ASA): lessons learned. Control and Cybernetics 26(1), 33–54 (1996)
20. Bäck, T., Fogel, D., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd. Bristol, UK (1997)
21. Laurenço, H., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 321–353. Kluwer Academic Publishers, Norwell, MA, USA (2002)
22. 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)
23. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)
24. Barr, R., Kelly, J., Resende, M., Stewart, W.: Designing and reporting computational experiments with heuristic methods. Journal of Heuristics 1(1), 9–32 (1995)
25. Bartz-Beielstein, T., Markon, S.: Tuning search algorithms for real-world applications: A regression tree based approach. In: Greenwood, G. (ed.) Proc. 2004 Congress on Evolutionary Computation (CEC'04), Piscataway, NJ, USA, pp. 1111–1118. IEEE Computer Society Press, Los Alamitos (2004)
26. Coy, S., Golden, B., Runger, G., Wasil, E.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 7(1), 77–97 (2001)
27. Xu, J., Kelly, J.: A network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science 30, 379–393 (1996)
28. 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)
29. Breedam, A.V.: 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)
30. Xu, J., Chiu, S., Glover, F.: Fine-tuning a tabu search algorithm with statistical tests. International Transactions on Operational Research 5(3), 233–244 (1998)
31. 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)

32. Pellegrini, P., Birattari, M.: The relevance of tuning the parameters of metaheuristics. A case study: The vehicle routing problem with stochastic demand. Technical Report TR/IRIDIA/2006-008, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (submitted for journal publication, 2006)
33. Aarts, E., Korst, J., van Laarhoven, P.: Simulated annealing. In: Aarts, E., Lenstra, J. (eds.) Local Search in Combinatorial Optimization, pp. 91–120. John Wiley & Sons, Inc. New York, USA (1997)
34. 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, USA (1991)
35. Friedman, J.: Multivariate adaptive regression splines. The Annals of Statistics 19, 1–141 (1991)
36. Birattari, M., Zlochin, M., Dorigo, M.: Towards a theory of practice in metaheuristics design: A machine learning perspective. Theoretical Informatics and Applications, Accepted for publication (2006)