The relevance of tuning the parameters of metaheuristics.

A case study: The vehicle routing problem with stochastic demand

Paola Pellegrini and Mauro Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2006-008

March 2006

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2006-008

The relevance of tuning the parameters of metaheuristics.

A case study: The vehicle routing problem with stochastic demand

Paola Pellegrini[†][‡] and Mauro Birattari[†]

[†]*IRIDIA, Université Libre de Bruxelles, Brussels, Belgium*
[‡]*Università Ca' Foscari, Venezia, Italy*
(e-mail: paolap@pellegrini.it, mbiro@ulb.ac.be)

March 28, 2006

Abstract

Metaheuristics are a class of promising algorithms for combinatorial optimization. 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. We say that, according to the specific context of applications, either a metaheuristic can be used *out-of-the-box*, or a *custom* implementation can be developed. This flexibility is one of the major strengths of metaheuristics but 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.

As a case study, this paper focuses 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 stochastic approaches to combinatorial optimization.

> *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, they 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 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.

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 if the versions considered are *out-of-the-box* or *custom* ones. Without this piece of information, the usefulness of these comparisons is somehow impaired.

The main goal of this paper is to illustrate this issue. 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. In particular, we wish to show that the relative performance of the above metaheuristics depends on whether *out-of-the-box* or *custom* implementations are 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.

In the current literature, the lack of information on the specific 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 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.

In this paper, in order to attenuate the above problems, we consider the five aforementioned metaheuristics in the implementations 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 Bianchi et al. [3].

In our analysis, these implementations play the role of *black-box* metaheuristics. Starting from them, we obtain the *out-of-the-box* and the *custom* versions. The former are obtained randomly drawing the parameters from a defined range. The latter are obtained by fine-tuning the parameters through an automatic process based on the F-Race algorithm [4, 5]. 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 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 arguments in favor of the validity and significance of our analysis should be sufficient in order to convince our reader. First, although many research papers fail

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

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 [6, 7, 5, 8]. Second, an accurate fine-tuning is indeed just one of the element that characterize a *custom* implementation of a metaheuristic. Other elements, as for example an advanced design and implementation of critical data structures, definitely play a major role. 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.

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

2 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 our analysis, and iii) the problem of fine-tuning metaheuristics.

2.1 The problem

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 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 [9] in 1969. Stewart and Golden [10], Dror and Trudeau [11], Laporte and Louveau [12] and Laporte et al. [13] use techniques from stochastic programming to solve optimally small instances. Bertsimas [14] and Bertsimas and Simchi-Levi [15] 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. Yang et al. [16] propose a strategy for splitting the *a priori* tour allowing the restocking before a stockout, when this is profitable. Secomandi [17, 18, 19] analyzes different possibilities for applying dynamic programming to this problem. Teodorović and Pavković [20] and Gendreau et al. [21] tackle the VRPSD using metaheuristic approaches. In particular, Teodorović and Pavković [20] adopt simulated annealing while Gendreau et al. [21] use tabu search. Finally, an extended analysis on the behavior of different metaheuristics is proposed by Bianchi et al. [3].

Two classical local search algorithms have been used for the VRPSD: the Or-opt and the 3-opt procedures. The first is proposed by Or [22] 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. [16] present an approximated way for computing the value of each move. This method is adopted also by Bianchi et al. [3], who propose also another approximation which is 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 extend this TSP-wise approximation also to the 3-opt local search [23].

2.2 Metaheuristics

Following Bianchi et al. [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

Tabu search has been introduced by Glover [1] in 1986, on the basis of early ideas formulated a decade before [24]. 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 [25]. The combination of the two elements prevents the search from tracing back its steps when moving away from local optima. The short term memory is represented by the *tabu list*, a first-in first-out queue of previously visited solutions.

Simulated annealing

Simulated annealing takes inspiration from the annealing process in crystals, which assume a low energy configuration when cooled with an appropriate cooling schedule [26, 27, 28, 29, 30]. 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 algorithm

Genetic computation is inspired by the ability shown by populations of living beings to evolve and adapt to changing conditions, under the pressure of natural selection [31]. This metaheuristic is based on the selection of individuals representing candidate solutions. Each individual is described by a *chromosome* which is a collection of *genes*, that is, a string of symbols. The *fitness* of an individual is a decreasing function of its cost. The procedure consists in allowing the evolution of the individuals, going from a generation to another through *recombination* and *crossover*, and using *mutation* or *modification* operators which lead to self-adaptation. Individuals with a higher *fitness* have a higher probability to be chosen as members of the population of the next iteration [32, 33, 34, 35, 36, 37].

Iterated local search

Iterated local search is one of the simplest metaheuristics, based on the reiteration of a local search procedure. It explores the neighborhoods of different solutions obtained via successive perturbations [38]. In order for this mechanism to be effective, the perturbation should not move the position of the new starting point too far from the previous local optimum, otherwise the search will loose the focus from the current explored area. At the same time, the perturbation should not be too feeble, otherwise the local search risks to converge back to the same local optimum already reached by the previous descent.

Ant colony optimization

Ant colony optimization is a metaheuristic based on the foraging behavior of ants. Ants randomly explore the area surrounding their nest. Once a food source is found, the ant evaluates the quantity and the quality of the food. Carrying some of it back to the nest, the ant deposits a pheromone trail on the ground. The amount of pheromone deposited may depend on the previous evaluation and will guide other ants to the food source. This indirect communication allows the ants to find the shortest path between the two points. The ant colony optimization metaheuristic takes inspiration from this process. 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 [2].

2.3 The tuning process

A metaheuristic is a general algorithmic framework which can be applied to different optimization problems. It 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]. 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 [6, 7, 39, 40, 5, 8]. 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 [41] 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. Parson and Johnson [42] and Breedam [43] use a similar approach. Xu et al. [44] describe 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. [8] and by Adenso-Díaz and Laguna [6] is based on the method that in the statistical literature is known as *response surface methodology.* Bartz-Beielstein and Markon [40] propose 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. Some procedures for tackling the tuning problem have been proposed by Birattari [5]. Among them, the F-Race method is the best performing one and has been used in a number of works on metaheuristics [45, 46, 47, 48, 49].

For the sake of completeness, we mention here another approach to tuning which 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 first influential descriptions of *on-line* adjustment of the parameters of an algorithm is given by Battiti and Tecchioli [50]. The authors introduce 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 paper. In particular, Section 3.1 describes the specific vehicle routing problem with stochastic demand that we consider. Moreover, it introduces the local search procedures and the approximations of the objective function that we adopt. Finally, a description of the class of instances considered in the study and of the instance generator is given. Section 3.2 describes the specific implementations we consider of the five metaheuristics under analysis. Section 3.3 describes F-Race, that is, the tuning algorithm that is adopted for fine-tuning the metaheuristics in our study.

3.1 The problem

As in most of the previously published works on VRPSD [14, 15, 3, 16], in this paper the problem is addressed by considering only one vehicle. This element was proved to give the best solution in absence of additional constraints [16]. 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. [16] and Bianchi et al. [3]. In particular, it is based on a dynamic programming recursion which 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. In the literature, when considering local search procedures, it is quite common to try to limit the cost for evaluating each move. This is done by calculating the difference of the value of two solutions (before and after a move) in an approximated way. Following this trend, for both Or-Opt and 3-opt, we have considered different ways of calculating the cost of a move:

- TSP-cost: only the variation in the cost of the *a priori* tour is considered [3]. This technique is known in the literature as *delta evaluation*;
- VRPSD-cost: only few steps of the dynamic programming recursion are calculated and compared with the original values [16, 3]. These steps are those that include the nodes that have been moved;
- EXACT-cost: the complete dynamic programming recursion is computed.

For the Or-Opt local search, all these variants are considered. For the 3-opt only the TSP-cost and the EXACT-cost are implemented. Since it is not evident which of these methods is the best performing for each metaheuristic, we consider the local search to be used as a parameter of the algorithms.

In order to reach some significant conclusion with our empirical analysis, a rather large set of instances is needed. The instances considered in Bianchi et al. [3] have been generated starting form standard benchmark instances for the TSP. Then, a demand distribution has been attached to each customer. The set obtained in this way by Bianchi et al. [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 Pellegrini and Birattari [51].

The created instances consist in nodes grouped in clusters. Each cluster represents a European city randomly selected in an available database. A city is represented as a set of concentric circles. The number of the circles depends on the population of the city considered: The higher the population, the higher the number of circles. We will refer to the areas obtained in this way as to *zones*. The nodes are uniformly distributed in each zone. The depot is located in the most external zone of a randomly selected city [51].

The traveling costs between nodes are functions of the travel time. A coefficient is associated to each zone. These values represent the allowed speed for going from one point to another. The Euclidean distance is multiplied for these coefficients for obtaining the traveling cost between nodes. The coefficient becomes smaller while moving from a zone to a more internal one. This models the characteristics of most European cities, where the actual speed decreases while one approaches the center. The velocities considered in order to compute the coefficients are 15 km/h for the most internal circle of each city and 40 km/h for the most external one. Proportional values are associated to the zones in between. The speed on the route connecting two cities is 80 km/h.

In this paper, we consider instances with either 50 or 60 nodes. They comprise three clusters representing three randomly selected Italian cities. Nine databases of this kind are used in order to represent different geographical structures.

Following [3], we consider instances in which the average demand and the spread are obtained as in Table 1.

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

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	$U(20, 35)$	$U(5, 15)$

Table 2. Structure of the tabu search algorithm.

```
Determine initial candidate solution $s$
Initialize temperature $T$
While stopping criterion is not satisfied
      $s'$ =local_search(s) #A variable neighborhood is considered
      If $s'$ is non-tabu or if it satisfies the aspiration criterion
         then $s = s'$
      Update tabu-list based on $s'$
```

capacity of the vehicle and average customer demand are not very frequent in 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. [3].

3.2 Metaheuristics

The implementation of the metaheuristics we consider is based on the code written by Bianchi et al. [3], which is available at `http://iridia.ulb.ac.be/vrpsd.ppsn8`. In the following, we give a short description of each algorithm. In Tables 2, 3, 4, 5, and 6, the different structures are presented. The function *local_search(s)* is used to indicate the application of the local search procedure to solution s. The parameters of the algorithms are briefly explained. As reference algorithm, following Bianchi et al. [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.

Tabu search

The implementation of the tabu search algorithm follows the schema reported in Table 2. The tabu-list represents the short term memory and includes partial solutions. This results in a particularly efficient algorithm. Nonetheless, this approach has the drawback that good quality solutions might be forbidden for being partially equal to a recently visited solution. To avoid this counter effect, an *aspiration criterion* is used: forbidden moves are allowed if the new solution is the new best one. The solution components that would re-establish the partial order of two just modified nodes are forbidden. 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 $0 \le t \le 1$ 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, the 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 to the local search procedure.

Table 3. Structure of the simulated annealing algorithm.

```
Determine initial candidate solution $s$
Initialize temperature $T$
While stopping criterion is not satisfied
      $s'$ =local_search(s)
      If $s'$ satisfies the probabilistic acceptance criterion based on $T$
         then $s = s'$
      Update temperature $T$
```

Table 4. Structure of the genetic algorithm.

```
Determine an initial population $SP$ of solutions
While stopping criterion is not satisfied
      Generate a set of solutions $SP_r$ by edge recombination
      Generate a set of solutions $SP_m$ from $SP_r$ by mutation
      For each $s_i \in SP_m$, $s'_i$ =local_search($s_i$)
         If $Cost(s'_i) < Cost(s_i)$
            then $s_i = s'_i$
      Select population $SP$ from solutions in $SP$, $SP_r$, $SP_m$
```

Simulated annealing

The implementation of the simulated annealing algorithm is presented in Table 3. 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

$$p(s'|T_k, s) = exp\left(-\frac{Cost(s')Cost(s)}{T_k}\right). \tag{1}$$

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* [26] is applied: the temperature is increased by adding T_0 to the current temperature. Besides the local search procedure adopted, 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 1 are computed. In the TSP-cost, only the length of the *a priori* tour is considered.

Genetic algorithm

The genetic algorithm is implemented according to the framework reported in Table 4. Edge recombination [52] consists in generating a tour starting from two solutions 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 to the local search procedure adopted.

Table 5. Structure of the iterated local search algorithm.

```
Determine initial candidate solution $s$
Perform local search on $s$
While stopping criterion is not satisfied
      $s'$ =perturbation(s)
      $s'$ =local_search($s'$)
      If $Cost(s') < Cost(s)$
         then $s = s'$
```

Table 6. Structure of the ant colony optimization algorithm.

```
Initialize pheromone trails
While stopping criterion is not satisfied
      Generate a population $P$ of $p$ solutions
      For each $s_i \in P$
         $s'_i$ =local_search($s_i$)
         If $Cost(s'_i) < Cost(s_i)$
            then $s_i = s'_i$
      Adapt pheromone trails based on $P$
```

Iterated local search

The implementation of the iterated local search algorithm follows the schema described in Table 5. The function *perturbation(s)* performs a perturbation on s. 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 to the local search procedure adopted.

Ant colony optimization

The ant colony optimization algorithm considered is described in Table 6. 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 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

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

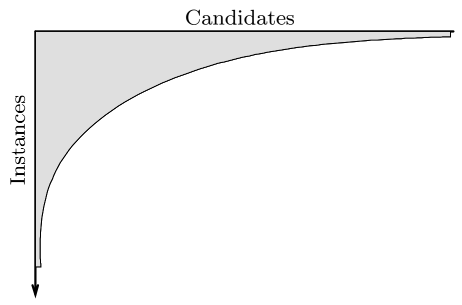


Figure 1. Graphical representation of computation performed by the racing approach. As the evaluation proceeds, the racing algorithm focuses more and more on the most promising candidates, discarding a configuration, as soon as sufficient evidence is gathered that it is suboptimal [5].

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 [53]. 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

The 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 metaheuristics. What characterizes the *custom* versions in our analysis is the fact that the parameters are accurately fine-tuned with the automatic procedure known as F-Race. This algorithm, as mentioned in Section 3.3, selects the best values of the parameters out of a given set of candidate ones. On the other hand, for 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. Equal probability has been associated to each configuration and, for each instance considered in the analysis, a random selection has been performed. Beside the *out-of-the-box* and the *custom* versions, our analysis also comprises the versions studied by Bianchi et al. [3]. For convenience, we will refer to these versions as *literature* versions. They differ from the *out-of-the-box* and the *custom* versions solely in the fact that the parameters of the *literature* versions have been set mostly on the basis of the experience of the researchers that have implemented the algorithms [3]. For each of the metaheuristics, beside the methods adopted for setting the parameters, the implementations considered in the *out-of-the-box*, *custom*, and *literature* versions are identical. All experiments are run on a cluster of AMD Opteron$^{\mathrm{TM}}$ 244, and 1000 instances are considered. The computation time is used as a

Table 7. Parameters for tabu search.

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

Table 8. Parameters for simulated annealing.

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

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. 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 running the experiments, but the two sets of instances are disjoint [54]. 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. Tables 7, 8, 9, 10, 11, and 12 report, for each metaheuristic, the parameters that have been considered for optimization, the possible values, and those that have been selected.

The results of the experiments with *custom*, *out-of-the-box*, and *literature* versions are reported in Figures 2, 3, and 5, respectively. These graphics report the ranking obtained by the metaheuristics under analysis. 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 far 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 a 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. [3], the local search procedure is not considered as a parameter of the algorithms. For this reason, the metaheuristics are presented in

Table 9. Parameters for genetic algorithm.

parameter	range	selected value
pop. size	10, 12, 14, 16, 18, 20, 22, 24	20
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	0.7
adaptive	Yes, No	Yes
local_search	Or-Opt(TSP-cost), Or-Opt(VRPSD-cost), Or-Opt(EXACT-cost), 3-Opt(TSP-cost), 3-Opt(EXACT-cost)	Or-Opt(TSP-cost)
total number of candidates = 1360		

Table 10. Parameters for iterated local search.

parameter	range	selected value
ϵ	$\frac{n}{x}, x \in \{0.005, 0.01, 0.05, 0.1$, multiples of 0.5 up to $150.0\}$	$\frac{n}{1.5}$
local_search	Or-Opt(TSP-cost), Or-Opt(VRPSD-cost), Or-Opt(EXACT-cost), 3-Opt(TSP-cost), 3-Opt(EXACT-cost)	Or-Opt(TSP-cost)
total number of candidates = 1520		

Table 11. Parameters for ant colony optimization.

parameter	range	selected value
p	5,10, 20	5
ρ	0.1, 0.5, 0.7	0.7
r	100, 150, 200	150
Q	$10^5, 10^6, 10^7, 10^8, 10^9$	10^8
local_search	Or-Opt(TSP-cost), Or-Opt(VRPSD-cost), Or-Opt(EXACT-cost), 3-Opt(TSP-cost), 3-Opt(EXACT-cost)	3-Opt(TSP-cost)
total number of candidates = 1620		

Figure 5 with the name of the metaheuristic paired with the one of the variant of the local search used. In Bianchi et al. [3], the 3-opt local search has been used only in association with iterated local search and genetic algorithms.

The first issue on which we wish to focus is a comparison between the results achieved by the *custom* and by the *out-of-the-box* versions, as displayed in Figures 2 and 3, respectively. It is worth noticing immediately that all the *custom* versions behave significantly better than the random restart local search. On the contrary, this is not the case for the *out-of-the-box* versions. 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. Another element to be noticed in this comparison between *custom* and *out-of-the-box* versions, is that the ranking is visibly different in the two contexts. This shows that different metaheuristics are influenced in a different measure by their parameters.

The boxplots given in Figure 4 represent the difference between the costs of the solutions obtained in the two contexts by the metaheuristics. 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. This figure is proposed in order to allow the evaluation of the impact of the tuning procedure in absolute terms. In Figure 4(a), the whole distributions are shown. In Figure 4(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, 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 to the value of their parameters and therefore benefit more than others from an accurate fine-tuning.

Figure 5 shows the ranking of the algorithms obtained in the third set of experiments in which the *literature* versions are considered [3]. As it can be noted by comparing this graph with Figures 2 and 3, 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 6 reports, for each metaheuristic, the comparison of the results obtained in the three sets of experiments. What clearly emerges is that all metaheuristics achieve the best results in their *custom* version. The difference is always statistically significant according to the

Table 12. Parameters for random restart.

parameter	range	selected value
local_search	Or-Opt(TSP-cost), Or-Opt(VRPSD-cost), Or-Opt(EXACT-cost), 3-Opt(TSP-cost), 3-Opt(EXACT-cost)	3-Opt(EXACT-cost)
total number of candidates = 5		

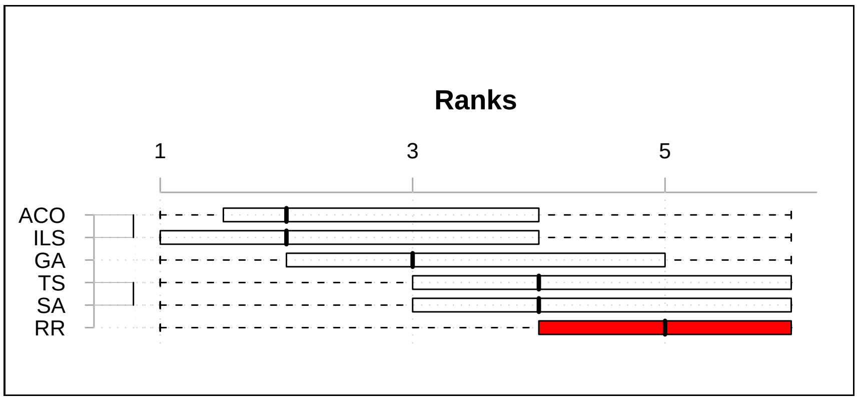


Figure 2. *Custom* versions: Results over 1000 instances of the metaheuristics when the values of the parameters are chosen through the F-Race procedure.

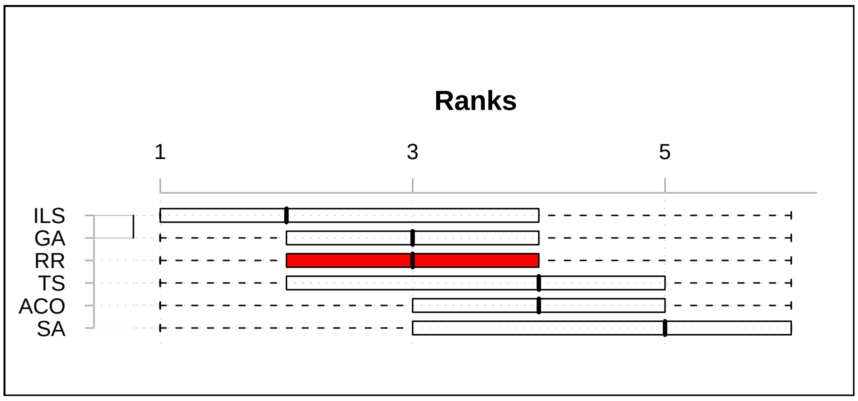
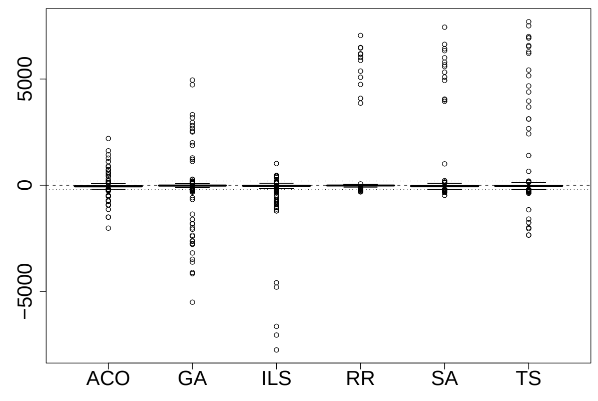


Figure 3. *Out-of-the-box* versions: Results over 1000 instances of all the metaheuristics when the values of the parameters are randomly chosen.

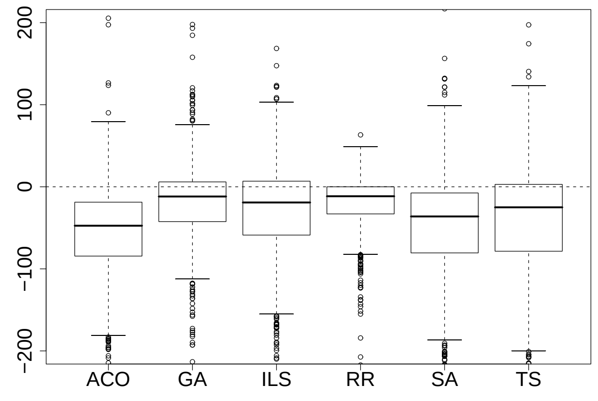
Friedman test. Moreover, it can be observed that *literature* versions, that is, those in which the parameters are set according to Bianchi et al. [3], obtain results that are comparable with those of the *out-of-the-box* versions. In particular, while for iterated local search and tabu search the values 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 adopted in Bianchi et al. [3] yield results that are significantly worse than those drawn at random. These results suggest that, by adopting the parameters that can be found in the literature, one should expect a performance comparable with what would be obtained by assigning random values to the parameters. A metaheuristic in which the parameters are set as reported in the literature should be *a priori* regarded as an *out-of-the-box* implementation.

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 also of a research recently published by Bianchi et al. [3]. In Bianchi et al. [3], the main goal was to study the impact of different ways of approximating the objective function, which in the vehicle routing problem with stochastic demand is rather



(a)



(b)

Figure 4. 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 4(a), the entire distribution is shown for each metaheuristic. Since the distributions are characterized by long tails, in Figure 4(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.

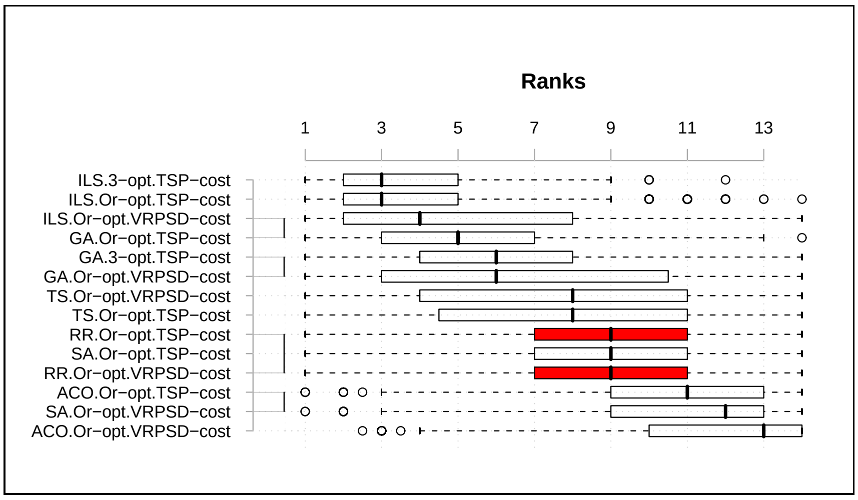


Figure 5. *Literature* versions: Results over 1000 instances of all the metaheuristics when the values of the parameters are set as in Bianchi et al. [3].

expensive to be computed. An accurate fine-tuning of the parameters of the metaheuristics under analysis has been considered out of the scope of the above research.

On the contrary, in this paper the fine-tuning of the metaheuristics under analysis plays a central role. Here, the goal is to highlight the fact that results obtained with *out-of-the-box* versions of metaheuristics might differ from those obtained with *custom* versions. The central role played by the fine-tuning derives from the fact that, 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. A possibly less expected result is that, when *out-of-the-box* implementations are considered, three metaheuristics out of five perform worse than the random restart local search, which is taken as a yardstick in our analysis. Only iterated local search and genetic algorithms perform better than the random restart local search. This is to be considered as a major failure for the other three algorithms. On the contrary, when *custom* versions are considered, all metaheuristics outperform the random restart local search. By itself, this result should be regarded as a strong point in favor of the systematic adoption of a fine-tuning algorithm like, for example, the F-Race algorithm considered in this study.

A second important element that emerges from the empirical analysis is an evidence that results obtained with *out-of-the-box* versions of metaheuristics should not be extended to *custom* versions. In particular, 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. For example, our results show that ant colony optimization, which in the *out-of-the-box* version ranked fourth out of five, ranked first in the *custom* version, together with iterated local search.

Finally, our analysis shows that, in average, by adopting the parameters that have been used in Bianchi et al. [3], the metaheuristics do not achieve performances that are significantly better than those obtained by our *out-of-the-box* versions. In Bianchi et al. [3], the parameters had been set on the basis of few non-systematic tests and mostly on the basis of previous experience of the researchers that implemented the different metaheuristics. Our analysis shows that, at least in this case study, setting the parameters on the basis of previous experience or, equivalently, on the basis of what published in the literature, is not significantly better than drawing them at random.

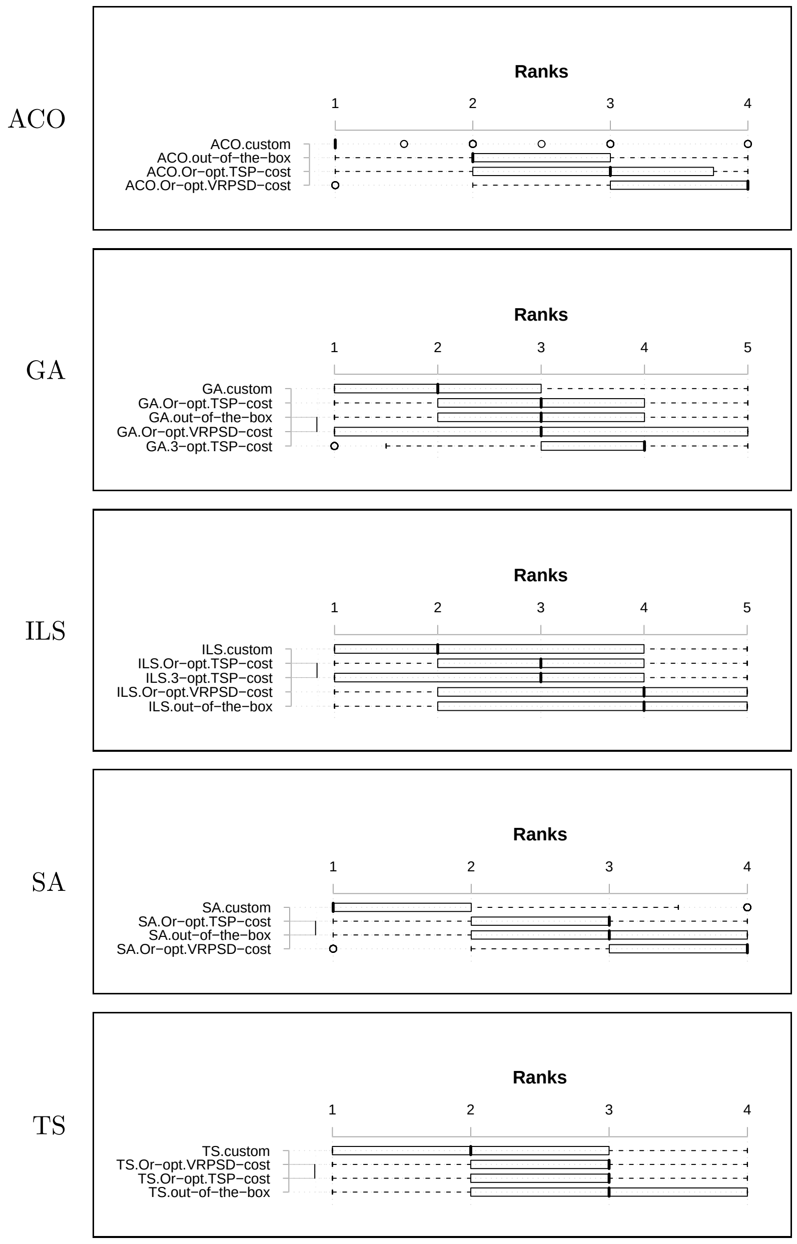


Figure 6. Comparison of the results obtained in the three sets of experiments. As it can be observed, the *custom* versions are always significantly better than all the others. For iterated local search and tabu search, *literature* versions are better that their *out-of-the-box* counterparts, that is, the values chosen by Bianchi et al. [3] behave better than the random ones. On the contrary, the *out-of-the-box* ant colony optimization works better that the *literature* version. Finally, in genetic algorithms and simulated annealing the results are mixed: the *out-of-the-box* versions is better that 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. [3].

This is indeed another strong argument in favor of a systematic fine-tuning of metaheuristics.

To sum up, the results presented in the paper show that the behavior of metaheuristics in their *out-of-the-box* version does not necessarily generalize to the *custom* versions, and vice versa. The impact of a proper customization—the fine-tuning of the parameters in this specific case—is detectable both when considering the performance of metaheuristics with respect to each other, and when comparing their behavior with some reference algorithm such as a random restart local search.

The fact that the results achieved by the *out-of-the-box* versions of the metaheuristics under analysis mimic so closely those obtained by using the values proposed by the literature, confirms the relevance of our research. In this precise sense, the analysis presented in this paper is strongly related to a subject which has an actual impact on the current research in the field of metaheuristics. In particular, this paper should convince that, when publishing the results of an empirical analysis, one should always be very clear about the specific context in which the study is performed. Namely, one should provide some measure of the development effort needed for obtaining the specific implementations under analysis. In general, the results obtained cannot be safely extended to other contexts. Clearly, if the information on the experimental context is missing, the results are of little value and can be greatly misleading.

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] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[2] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, USA, 2004.

[3] L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demand. *Journal of Mathematical Modelling and Algorithms*, 2006. In press. available as `http://springerlink.metapress.com/link.asp?id=y630h66728446j63`.

[4] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W.B. Langdon, editor, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers.

[5] M. Birattari. *The problem of tuning metaheuristics as seen from a machine learning perspective.* PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2005.

[6] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

[7] R.S. Barr, J.P. Kelly, M.G.C. Resende, and W.R. Stewart. Designing and reporting computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.

[8] S.P. Coy, B.L. Golden, G.C. Runger, and E.A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

[9] F. Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3:192–204, 1969.

[10] W. Stewart and B. Golden. Stochastic vehicle routing: a comprehensive approach. *European Journal of Operational Research*, 14:371–385, 1983.

[11] M. Dror and P. Trudeau. Stochastic vehicle routing with modified saving algorithm. *European Journal of Operational Research*, 23:228–235, 1986.

[12] G. Laporte and F. Louveau. 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.

[13] G. Laporte, F. Louveau, and H. Mercure. 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.

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

[15] D.J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(3):286–304, 1996.

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

[17] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27:1201–1225, 2000.

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

[19] N. Secomandi. Analysis of a rollout approach to sequencing problems with stochastic routing applications. *Journal of Heuristics*, 9:321–352, 2003.

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

[21] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Working paper, CRT, University of Montreal, Montreal, Canada, 1994.

[22] I. Or. *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking.* PhD thesis, Northwestern University, Evanston, IL, USA, 1976.

[23] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Tech. Journal*, 44:2245–2269, 1965.

[24] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.

[25] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[26] E.H.L. Aarts, J.H.M. Korst, and P.J.M. van Laarhoven. Simulated annealing. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 91–120. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[27] V. Cerny. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[28] M. Fleischer. Simulated annealing: past, present and future. In Lilegdon W.R. Alexopoulos C.L., Kang K. and Goldsam G., editors, *Proceedings of the 1995 Winter Simulation Conference*, pages 155–161, 1995.

[29] L. Ingber. Adaptive simulated annealing (ASA): lessons learned. *Control and Cybernetics*, 26(1):33–54, 1996.

[30] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[31] C.R. Darwin. *On the Origin of Species by Means of Natural Selection. Or the preservation of favoured races in the struggle for life.* John Murray, London, UK, 1859.

[32] T. Back, D.B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation.* IOP Publishing Ltd., Bristol, UK, 1997.

[33] L.J. Fogel. Toward inductive inference automata. In *Proceedings of the International Federation for Information Processing Congress*, pages 395–399, Munich, Germany, 1962.

[34] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligent through Simulated Evolution.* John Wiley & Sons, New York, NY, USA, 1966.

[35] D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley Publishing Company, Reading, MA, USA, 1989.

[36] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Harbor, MI, USA, 1975.

[37] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, Germany, 1973.

[38] H.R. Laurenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[39] T. Bartz-Beielstein, M. Preuss, and A. Reinholz. Evolutionary algorithms for optimization practitioners. Technical Report CI-151/03, Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence, Universität Dortmund, Dortmund, Germany, 2003.

[40] T. Bartz-Beielstein and S. Markon. Tuning search algorithms for real-world applications: A regression tree based approach. In G.W. Greenwood, editor, *Proc. 2004 Congress on Evolutionary Computation (CEC'04)*, pages 1111–1118, Piscataway, NJ, USA, 2004. IEEE Press.

[41] J. Xu and J. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.

[42] R. Parson and M. Johnson. 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.

[43] A. Van Breedam. 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.

[44] J. Xu, S.Y. Chiu, and F. Glover. Fine-tuning a tabu search algorithm with statistical tests. *International Transactions on Operational Research*, 5(3):233–244, 1998.

[45] M. Chiarandini. *Stochastic local search for overconstrained problems.* PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2005.

[46] M. Chiarandini and T. Stützle. Experimental evaluation of course timetabling algorithms. Technical Report AIDA-02-05, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany, 2002.

[47] M.L. den Besten. *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.

[48] T. Schiavinotto and T. Stützle. The linear ordering problem: instances, search space analysis and algorithms. *Journal of Mathematical Modelling and Algorithms*, 3:367–402, 2004.

[49] B. Yuan and M. Gallagher. Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In X. Yao, E. Burke, J.A. Lonzano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tino, A. Kabán, and H.P. Schwefel, editors, *Parallel Problem Solvinf from Nature, 8th International Conference, PPSN VIII*, volume 3242 of LNCS, pages 172–181, Berlin, Germany, 2004. Springer-Verlag.

[50] R. Battiti and G. Tecchioli. The reactive tabu search. *ORSA Journal on Computing*, 6: 126–585, 1994.

[51] P. Pellegrini and M. Birattari. Instances generator for the vehicle routing problem with stochastic demand. Technical Report TR/IRIDIA/2005-10, Iridia, Université Libre de Bruxelles, Brussels, Belgium, 2005.

[52] D. Whitley, T. Starkweather, and D. Shaner. The traveling salesman problem and sequence scheduling: quality solutions using genetic edge recombination. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 350–372. Van Nostrand Reinhold, New York, NY, USA, 1991.

[53] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.

[54] M. Birattari, M. Zlochin, and M. Dorigo. Towards a theory of practice in metaheuristics design: A machine learning perspective. *Theoretical Informatics and Applications*, 2006. Accepted for publication.