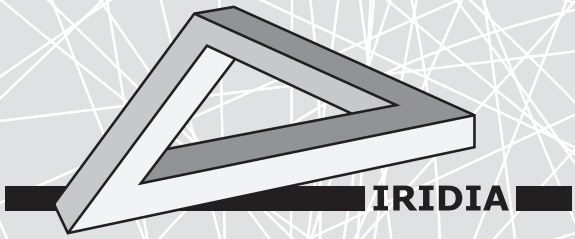

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Improvement Strategies for the F-Race algorithm: Sampling Design and Iterative Refinement

Prasanna Balaprakash, Mauro Birattari,
and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2007-011

May 2007

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/2007-011

Improvement Strategies for the F-Race algorithm: Sampling Design and Iterative Refinement

Prasanna Balaprakash pbalapra@ulb.ac.be
Mauro Birattari mbiro@ulb.ac.be
Thomas Stützle stuetzle@ulb.ac.be

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

May 29, 2007

Abstract

Finding appropriate values for the parameters of an algorithm is a challenging, important, and time consuming task. While typically parameters are tuned by hand, recent studies have shown that automatic tuning procedures can effectively handle this task and often find better parameter settings. `F-Race` has been proposed specifically for this purpose and it has proven to be very effective in a number of cases. `F-Race` is a racing algorithm that starts by considering a number of candidate parameter settings and eliminates inferior ones as soon as enough statistical evidence arises against them. In this paper, we propose two modifications to the usual way of applying `F-Race` that on the one hand, make it suitable for tuning tasks with a very large number of initial candidate parameter settings and, on the other hand, allow a significant reduction of the number of function evaluations without any major loss in solution quality. We evaluate the proposed modifications on a number of stochastic local search algorithms and we show their effectiveness.

1 Introduction

The full potential of a parameterized algorithm cannot be achieved unless its parameters are fine tuned. Often, practitioners tune the parameters using their personal experience guided by some rules of thumb. Usually, such a procedure is tedious and time consuming and, hence, it is not surprising that some authors say that 90% of the total time needed for developing an algorithm is dedicated to find the right parameter values [1]. Therefore, an effective automatic tuning procedure is an absolute must by which the computational time and the human intervention required for tuning can be significantly reduced. In fact, the selection of parameter values that drive heuristics is itself a scientific endeavor and deserves more attention than it has received in the operations research literature [2]. In this context, few procedures have been proposed in the literature.

F-Race [3, 4] is one among them and has proven to be successful and useful in a number of tuning tasks [5, 6].

Inspired by a class of racing algorithms proposed in the machine learning literature, F-Race evaluates a given set of parameter configurations sequentially on a number of problem instances. As soon as statistical evidence is obtained that a candidate configuration is worse than at least another one, the inferior candidate is discarded and not considered for further evaluation. In all previously published works using F-Race, the initial candidate configurations were obtained through a full factorial design. This design is primarily used to select the best parameter configuration from a relatively small set of promising configurations that the practitioner has already established. Nevertheless, the main difficulty of this design is that, if the practitioner is confronted with a large number of parameters and a wide range of possible values for each parameter, the number of initial configurations becomes quite large. In such cases, the adoption of the full factorial design within F-Race can become impractical and computationally prohibitive. In order to tackle this problem, we propose two modifications to the original F-Race approach. The first consists in generating configurations by random sampling. Notwithstanding the simplicity, the empirical results show that this approach can be more effective—in the context of tuning tasks—than the adoption of a full factorial design. However, if the number of parameters is large, this methodology might need a large number of configurations to achieve good results. We alleviate this problem taking inspiration from model-based search techniques [7]. The second procedure uses a probabilistic model defined on the set of all possible parameter configurations and at each iteration, a small set of parameter configurations is generated according to the model. Elite configurations selected by F-Race are then used to update the model in order to bias the search around the high quality parameter configurations.

The paper is organized as follows: In Section 2, we introduce the proposed approach and we present some empirical results in Section 3. We discuss some related work in Section 4, and conclude the paper in Section 5.

2 Sampling F-Race and Iterative F-Race for tuning stochastic local search algorithms

For a formal definition of the problem of tuning SLS algorithms, we follow Birattari et al. [3]: the problem is defined as a 6 tuple $\langle \Theta, I, P_I, P_c, t, \mathcal{C} \rangle$, where Θ is the finite set of candidate configurations, I is the possibly infinite set of problem instances, P_I is a probability measure over the set I, t is a function associating to every instance the computation time that is allocated to it, P_C is a probability measure over the set of all possible values for the cost of the best solution found in a run of a configuration $\theta \in \Theta$ on an instance i, $\mathcal{C}(\theta)$ is the criterion that needs to be optimized with respect to θ: the solution of the

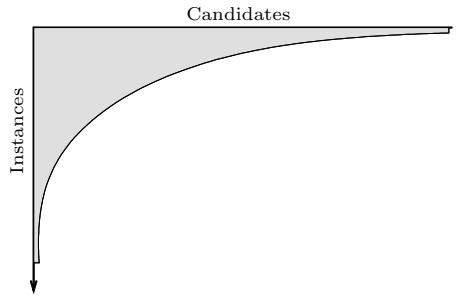

Figure 1: Visual representation of `F-Race`: a set of given candidate configurations are sequentially evaluated on a number of instances. As soon as sufficient evidence is gathered that a candidate configuration is worse than at least another one, the former is discarded from the race and is not further evaluated.

tuning problem consists in finding a configuration θ^* such that

$$\theta^* = \arg\min_{\theta} \mathcal{C}(\theta). \tag{1}$$

Typically, $\mathcal{C}(\theta)$ is an expected value where the expectation is considered with respect to both P_I and P_C. The main advantage of using expectation is that it can be effectively and reliably estimated with Monte Carlo procedures. In this paper, we focus on the minimization of the expected value of the solution cost and the criterion is given as:

$$\mathcal{C}(\theta) = \mathbb{E}_{I,C}\Big[c(\theta, i)\Big] = \int_I \int_C c_t(\theta, i)\ \mathrm{d}P_C(c_t|\theta, i)\ \mathrm{d}P_I(i), \tag{2}$$

where, $c_t(\theta, i)$ is a random variable that represents the cost of the best solution found by running configuration θ on instance i for t seconds. The integration is taken in the Lebesgue sense and the integrals are estimated in a Monte Carlo fashion on the basis of a so-called *tuning set* of instances. It is straightforward to use criteria other than the expected value such as inter-quartile range of the solution cost. In the case of decision problems, the practitioner might be interested in minimizing the run-time of an algorithm, a task that can be handled in a straightforward way by `F-Race`.

`F-Race` is inspired by a class of racing algorithms proposed in the machine learning literature for tackling the model selection problem [8, 9]. In `F-Race`, as in other racing algorithms, a set of given candidate configurations are sequentially evaluated on a number of tuning instances. As soon as sufficient evidence is gathered that a candidate configuration is worse than at least another one, the former is discarded from the race and is not further evaluated. The race terminates when either one single candidate configuration remains, or the available budget of computation time is used. The peculiarity of `F-Race` compared to other racing algorithms is the adoption of the *Friedman two-way analysis of*

variance by ranks [10], a nonparametric statistical test that appears particularly suitable in the context of racing algorithms for the tuning task. The `F-Race` procedure can be graphically illustrated as shown in Figure 2.

The main focus of this paper is the method by which the initial set of configurations is obtained in `F-Race`: while `F-Race` does not specify how Θ is defined, in most of the studies on `F-Race`, the configurations are defined using a full factorial design (`FFD`). In the simplest case, this is done as follows: Let $\mathrm{M} = \{\mathrm{M}_1, \ldots, \mathrm{M}_d\}$ be the set of parameters that need to be tuned whose ranges are given by (min_k, max_k), for $k = 1, \ldots, d$, where min_k and max_k are the minimum and maximum values of the parameter M_k, respectively. For each element in M, the practitioner has to choose a certain number of values; each possible combination of these parameter values leads to one unique configuration and the set of all possible combinations forms the initial set of configurations. If l_k values are chosen for M_k, then the number of initial configurations is $\prod_{k=1}^{d} l_k$. When each parameter takes l values, then $\prod_{k=1}^{d} l = l^d$: the number of configurations grows exponentially with respect to the number of parameters. As a consequence, even a reasonable number of possible values for each parameter makes the adoption of a full factorial design impractical and computationally prohibitive.

2.1 Sampling `F-Race`

A simple way to overcome the shortcomings of `FFD` is sampling. This means that the elements of Θ are sampled according to a given probability measure P_X defined on the space X of parameter values. If *a priori* knowledge is available on the effect of the parameters and on their interactions, this knowledge can be used to shape the probability measure P_X and therefore to suitably bias the sampling of the initial configurations. On the other hand, if no *a priori* knowledge on the parameter values is available, except the boundary constraints, then each possible value in the available range for each parameter should be given equal probability of being selected in sampling. In this case, P_X is a d-variate uniform distribution, which is factorized by a product of d univariate independent uniform distributions. A sample from the d-variate uniform distribution is a vector corresponding to a configuration θ such that a value x_k in the vector is sampled from the univariate independent uniform distribution parameterized by (min_k, max_k). We call this strategy *random sampling design* (`RSD`). The `F-Race` procedure is then applied to the set of sampled configurations. We denote this procedure as `RSD/F-Race`. It should be noted that the performance of the winning configuration is greatly determined by the number of sampled configurations, N_{max}.

2.2 Iterative `F-Race`

`RSD/F-Race` can identify promising configurations in the search space. However, finding the best configuration from the promising regions is often a difficult task. In order to address this issue, we propose `iterative F-Race` (`I/F-Race`),

a supplementary mechanism to the original `F-Race` approach. It is an iterative procedure in which each iteration consists in first defining a probability measure over the parameter space using promising configurations obtained from the previous iteration, then generating configurations that are distributed according to the newly defined probability measure, and finally applying `F-Race` on the generated configurations. This approach falls under the general framework of model-based search [7].

The way in which the probability measure is defined at each iteration plays a crucial role in biasing the search towards regions containing high quality configurations. The main issues in the search bias are the choice of the distribution and search intensification. For what concerns the distribution, there exist a number of choices. Here, we adopt a d-variate normal distribution parameterized by mean vector and covariance matrix. In order to intensify the search around the promising configurations, a d-variate normal distribution is defined on each surviving configuration from the previous iteration such that the distribution is centered at the values of the corresponding configuration. Moreover, the spread of the normal densities given by the covariance matrix is gradually reduced at each iteration.

This paper focuses on a scenario in which the practitioner does not have any *a priori* knowledge on the parameter values. Hence, we assume that the values taken by the parameters are independent, that is, knowing a value for a particular parameter does not give any information on the values taken by the other parameters. Consequently, the d-variate normal distribution is factorized by a product of d univariate independent normal densities parameterized by $\mu = (\mu_1, \ldots, \mu_d)$ and $\sigma = (\sigma_1, \ldots, \sigma_d)$. At each iteration, the standard deviation vector σ of the normal densities is reduced heuristically using the idea of volume reduction: Suppose that N_s configurations survive after a given iteration; we denote the surviving configurations as $\theta_s = (x_1^s, \ldots, x_d^s)$, for $s = 1, \ldots, N_s$. At a given iteration r, let V_r be the total volume of the d-dimensional sampling region bounded by $(\mu_k^{s_r} \pm \sigma_k^{s_r})$, for $k = 1, \ldots, d$; for iteration $r+1$, in order to intensify the search, we reduce the volume of the sampling region by a factor equal to the number of sample configurations allowed for each iteration, N_{max}; therefore $\mathrm{V}_{r+1} = \mathrm{V}_r / N_{max}$, from which after some basic mathematical transformation, we have:

$$\sigma_k^s = R_k^{s_{prev}} \cdot \left(\frac{1}{N_{max}}\right)^{1/d} \qquad \text{for } k = 1, \ldots, d, \tag{3}$$

where $R_k^{s_{prev}}$ is set to standard deviation of the normal distribution component from which x_k^s has been sampled from the previous iteration. In simple terms, the adoption of Equation 3 allows `I/F-Race` to reduce the range of each parameter that falls around one standard deviation from the mean at a constant rate of $(1/N_{max})^{1/d}$ for each iteration—the larger the value of N_{max}, the higher the rate of volume reduction. Though one could use more advanced techniques to update the distribution as suggested by the model-based search framework [7], we have adopted the above described heuristic way of intensifying search due to its simplicity.

Note that in the first iteration, a d-variate uniform distribution is used as the probability measure, thus for the following iteration, $R_k^{s_{prev}}$ is set to the half of range, that is, $(max_k - min_k)/2$, where max_k and min_k are parameters of the uniform distribution component from which x_k^s has been sampled, respectively.

The proposed approach adopts a strategy in which the number of configurations drawn from a d-variate normal distribution defined on a surviving configuration is inversely proportional to the configurations' expected solution cost. Recall that we are faced with the minimization of the expected solution cost. To do so, a *selection probability* is defined: the surviving configurations are ranked according to their expected solution costs and the probability of selecting a d-variate normal distribution defined on a configuration with rank z is given by:

$$p_z = \frac{N_s - z + 1}{N_s \cdot (N_s + 1)/2}. \tag{4}$$

A configuration is obtained by first choosing a d-variate normal distribution according to Equation 4, and then sampling from the chosen distribution. This is repeated until N_{max} configurations are sampled.

2.2.1 Implementation specific details.

In order to guarantee that `I/F-Race` does a specific minimum number of iterations and that it has a minimum number of survivors, we have modified `F-Race` slightly to stop it prematurely. At each iteration, racing is stopped if one of the following conditions is true:

- when N_{min} configuration remains;
- when a certain amount of computational budget, CB_{min}, is used;
- when the configurations in the race are evaluated on at least I_{max} instances.

Though these modifications introduce 3 parameters, they are set in a reasonable and straightforward way with respect to the total computational budget CB when the algorithm starts: (i) CB_{min} is set to $CB/5$: this setting allows `I/F-Race` to perform at least five iterations; (ii) N_{min} is set to d: this setting enables `I/F-Race` to search in a number of promising regions rather than just concentrating on a single region; (iii) I_{max} is set to $2 \cdot (CB_{min}/N_{max})$: if none of the configurations is eliminated from the race then each configuration has been evaluated on CB_{min}/N_{max} instances; hence, twice this value seems to be a reasonable upper bound.

The maximum number N_{max} of configurations allowed for each race is kept constant throughout the procedure. Moreover, the N_s configurations that have survived the race are allowed to compete with the newly sampled configurations. Therefore, $N_{max} - N_s$ configurations are sampled anew at each iteration.

The order in which the instances are given to the race is randomly shuffled for each iteration. Since the surviving configurations of each race are allowed

to enter into the next race, their results could be reused if the configuration has already been evaluated on a particular instance. However, since we do not want to bias `I/F-Race` in the empirical study, we did not use this possibility here.

The boundary constraints are handled in an explicit way. We adopt a method that consists in assigning the boundary value if the sampled value is outside the boundary. The rationale behind this adoption is to allow the exploration of values that lay at the boundary. In the case of parameters that take integer values, the value assigned to each integer parameter in the entire procedure is rounded off to the nearest integer.

3 Experiments

In this section, we study the proposed `RSD/F-Race` and `I/F-Race` using three examples. Though any parameterized algorithm may be tuned, all three examples concern the tuning of stochastic local search algorithms [11]: (i) tuning $\mathcal{MAX}-\mathcal{MIN}$ ant system ($\mathcal{MM}$AS) [12], a particular ant colony optimization algorithm, for a class of instances of the TRAVELING SALESMAN PROBLEM (TSP), (ii) tuning estimation-based local search, a new local search algorithm for stochastic combinatorial optimization problems [13], for a class of instances of the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP), and (iii) tuning a simulated annealing algorithm for a class of instances of the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS (VRP-SD). The primary goal of these examples is to show that `RSD/F-Race` and `I/F-Race` can significantly reduce the computational budget required for tuning.

We compare `RSD/F-Race` and `I/F-Race` with an implementation of `F-Race` that uses a full factorial design (`FFD`). For `RSD/F-Race` and `I/F-Race` we make the assumption that the *a priori* knowledge on the parameter values is not available. In the case of `FFD`, we consider two variants:

1. `FFD` that uses *a priori* knowledge; a parameter M_k is allowed to take l_k values, for $k = 1, \ldots, d$, where l_k values are chosen according to the *a priori* knowledge available on the parameter values; we denote this variant by $\texttt{FFD}_{\mathrm{A}}$`/F-Race`.

2. `FFD` that uses random values: a parameter M_k is allowed to take l_k values, for $k = 1, \ldots, d$, where l_k values are chosen randomly; we denote this variant by $\texttt{FFD}_{\mathrm{R}}$`/F-Race`. Note that the number of configurations in this variant is the same as that of $\texttt{FFD}_{\mathrm{A}}$`/F-Race`. This serves as a yardstick to analyze the usefulness of the *a priori* knowledge. The rationale behind the adoption of this yardstick is that if one just takes random values for `FFD` and achieves better results then $\texttt{FFD}_{\mathrm{A}}$`/F-Race`, then we can conjecture that the available *a priori* knowledge is either not accurate or simply not useful, at least in the examples that we consider here.

The minimum number of steps allowed in `F-Race` for all algorithms before applying the *Friedman* test is set to 5 as proposed in [4].

The maximum computational budget of $\texttt{FFD}_\text{A}$/F-Race and $\texttt{FFD}_\text{R}$/F-Race are set to 10 times the number of initial configurations. This budget is also given for RSD/F-Race and I/F-Race. In order to force RSD/F-Race to use the entire computational budget, the number of configurations is set to one-sixth of the computation budget. Since I/F-Race needs to perform at least five F-races with the same budget as that of RSD/F-Race, the number of initial configurations in each F-Race run by I/F-Race is set to one-fifth of the number of configurations given to RSD/F-Race. Moreover, in order to study the effectiveness of RSD/F-Race and I/F-Race under strong budget constraints, the computational budget is reduced by a factor of two, four, and eight. Note that, in these cases, the number of configurations in RSD/F-Race and I/F-Race is set according to the allowed budget using the same rule as described before.

Each tuning algorithm is allowed to perform 10 trials and the order in which the instances are given to an algorithm is randomly shuffled for each trial.

All tuning algorithms were implemented and run under R version 2.4[1] and we used a public domain implementation of F-Race in R which is freely available for download [14]. $\mathcal{MM}$AS[2] and estimation-based local search were implemented in C and compiled with gcc, version 3.4. Simulated annealing for VRP-SD is implemented in C++. Experiments were carried out on AMD Opteron$^{\text{TM}}$244 1.75 GHz processors with 1 MB L2-Cache and 2 GB RAM, running under the Rocks Cluster Distribution 4.2 GNU/Linux.

In order to quantify the effectiveness of each algorithm, we study the expected solution cost of the winning configuration $\mathcal{C}(\theta^*)$, where the expectation is taken with respect to the set of all trials and the set of all test instances. We report the expected solution cost of each algorithm, measured as the percentage deviation from a *reference cost*, which is given by the average over $\mathcal{C}(\theta^*)$ obtained by each algorithm. The adoption of *reference cost* allows us to compare the expected solution cost of different algorithms more directly.

In order to test whether the observed differences between the expected solution costs of different tuning algorithms are significant in a statistical sense, a random permutation test is adopted. The level of significance at which we reject the null hypothesis is 0.05; two sided p-value is computed for each comparison.

3.1 Tuning $\mathcal{MM}$AS for TSP

In this study, we tune 6 parameters of $\mathcal{MM}$AS:

1. relative influence of pheromone trails, α;
2. relative influence of heuristic information, β;
3. pheromone evaporation rate, ρ;

[1] R is a language and environment for statistical computing that is freely available under the GNU GPL license at http://www.r-project.org/

[2] We used the ACOTSP package, which is a public domain software that provides an implementation of various ant colony optimization algorithms applied to the symmetric TSP. The package available at: http://www.aco-metaheuristic.org/aco-code/

Table 1: Computational results for tuning $\mathcal{MM}$AS for TSP. The column entries with the label `per.dev` shows the percentage deviation of each algorithms' expected solution cost from the *reference cost*: $+x$ means that the expected solution cost of the algorithm is x% more than the *reference cost* and $-x$ means that the expected solution cost of the algorithm is x% less than the *reference cost*. The column entries with the label with `max.bud` shows the maximum number of evaluations given to each algorithm and the column with the label `usd.bud` shows the average number of evaluations used by each algorithm.

algo	per.dev	max.bud	usd.bud
$\texttt{FFD}_\text{R}$/F-Race	+13.45	7290	5954
$\texttt{FFD}_\text{A}$/F-Race	+11.13	7290	5233
RSD/F-Race	−2.69	7290	7232
I/F-Race	−3.92	7290	7181
RSD/F-Race	−2.55	3645	3275
I/F-Race	−3.84	3645	3564
RSD/F-Race	−2.51	1822	1699
I/F-Race	−3.66	1822	1793
RSD/F-Race	−2.17	911	823
I/F-Race	−3.23	911	894

4. parameter used in computing the minimum pheromone trail value τ_{min}, γ, which is given by $\tau_{max}/(\gamma * instance\_size)$;
5. number of ants, m;
6. number of neighbors used in the solution construction phase, nn.

In $\texttt{FFD}_\text{A}$/F-Race and $\texttt{FFD}_\text{R}$/F-Race, each parameter is allowed to take 3 values. The parameter values in $\texttt{FFD}_\text{A}$/F-Race are set as follows: $\alpha \in \{0.75, 1.00, 1.50\}$, $\beta \in \{1.00, 3.00, 5.00\}$, $\rho \in \{0.01, 0.02\ 0.03\}$, $\gamma \in \{1.00, 2.00, 3.00\}$, $m \in \{500, 750, 1000\}$, and $nn \in \{20, 30, 40\}$. These values are chosen reasonably close to the values as proposed in [15]. Note that the values are chosen from the version without the local search. The computational time allowed for evaluating a configuration on an instance is set to 20 seconds. Instances are generated with the DIMACS instance generator [16]. We used uniformly distributed Euclidean instances of size 750; 1000 instances were generated for tuning; 300 other instances were generated for evaluating the winning configuration. Table 1 shows the percentage deviation of each algorithms' expected solution cost from the *reference cost*, maximum budget allowed for each algorithm and the average number of evaluations used by each algorithm.

From the results, we can see that I/F-Race is very competitive: under equal computational budget, the expected solution cost of I/F-Race is approximately 17% and 15% less than that of $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race, respectively (the observed differences are significant according to the random permutation test). On the other hand, the expected solution cost of RSD/F-Race is also very low. However, I/F-Race reaches an expected cost that is about 1% less than

that of RSD/F-Race. Indeed, the observed difference is significant in a statistical sense. Regarding the budget, $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race use only 80% and 70% of the maximum budget. This early termination of the F-Race is attributed to the adoption of FFD: since, there are rather few possible values for each parameter, the inferior configurations are identified and discarded within few steps. However, the poor performance of $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race is not only attributable to the fact that they do not use the budget effectively: Given only half of the computational budget (a maximum budget of 3645), RSD/F-Race and I/F-Race achieve expected solution costs that are still 17% and 15% lower than $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race, respectively (the observed differences are significant according to the random permutation test). Another important observation is that, in the case of I/F-Race and RSD/F-Race, reducing the budget does not degrade the effectiveness to a large extent. Furthermore, in all these reduced budget cases, I/F-Race achieves an expected solution cost which is approximately 1% less than that of RSD/F-Race (the observed differences are significant according to the random permutation test).

3.2 Tuning estimation-based local search for PTSP

Estimation-based local search is an iterative improvement algorithm that makes use of the 2-exchange and node-insertion neighborhood relation, where the delta evaluation is performed using empirical estimation techniques [13]. In order to increase the effectiveness of this algorithm, a variance reduction technique called importance sampling has been adopted. Three parameters that need to be tuned in this algorithm are:

1. shift probability for 2-exchange moves, p_1;
2. number of nodes allowed for shift in 2-exchange moves, w;
3. shift probability for node-insertion moves, p_2.

Since this a recently developed algorithm, *a priori* knowledge is not available on the parameter values. Thus, in $\texttt{FFD}_\text{A}$/F-Race, the values are assigned by discretization: for each parameter, the range is discretized as follows: $p_1 = p_2 \in \{0.16, 0.33, 0.50, 0.66, 0.83\}$, and $w = \{8, 17, 25, 33, 42\}$. Estimation-based local search is allowed to run until it reaches a local optimum. Instances are generated as described in [13]: we used clustered Euclidean instances of size 1000; 800 instances were generated for tuning; 800 more instances were generated for evaluating the winning configuration.

The computational results show that the difference between the expected cost of the solutions obtained by different algorithms exhibits a trend similar to the one observed in the TSP experiments. However, the percentage deviations from the *reference cost* are relatively small: under equal computational budget, the expected solution cost of I/F-Race and RSD/F-Race are approximately 2% less than that of $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race, respectively. Note that this difference is significant according to a random permutation test. Though

Table 2: Computational results for tuning estimation-based local search for PTSP. The column entries with the label `per.dev` shows the percentage deviation of each algorithms' expected solution cost from the *reference cost*: $+x$ means that the expected solution cost of the algorithm is $x\%$ more than the *reference cost* and $-x$ means that the expected solution cost of the algorithm is $x\%$ less than the *reference cost*. The column entries with the label with `max.bud` shows the maximum number of evaluations given to each algorithm and the column with the label `usd.bud` shows the average number of evaluations used by each algorithm.

algo	per.dev	max.bud	usd.bud
$\texttt{FFD}_\text{R}$/F-Race	+1.45	1250	1196
$\texttt{FFD}_\text{A}$/F-Race	+1.52	1250	1247
RSD/F-Race	−0.62	1250	1140
I/F-Race	−0.53	1250	1232
RSD/F-Race	−0.17	625	615
I/F-Race	−0.52	625	618
RSD/F-Race	−0.06	312	307
I/F-Race	−0.58	312	278
RSD/F-Race	−0.37	156	154
I/F-Race	−0.11	156	150

RSD/F-Race obtains an expected solution cost which is 0.01% less than that of I/F-Race, the random permutation test cannot reject the null hypothesis. The overall low percentage deviation between algorithms is attributed to the fact that the estimation based local search is not extremely sensitive to the parameter values: there are only 3 parameters and interactions among them are quite low. As a consequence, the tuning task becomes relatively easy (as in the case of the previous task of tuning of $\mathcal{MM}$AS). This can be easily seen with the used budget of $\texttt{FFD}_\text{R}$/F-Race: if the task of finding a good configurations were difficult, the race would have terminated early. Yet, this is not the case and almost the entire computational budget has been used.

The numerical results on the budget constraints show that both RSD/F-Race and I/F-Race are indeed effective. Given only one-eighth of the computational budget (a maximum budget of 156), RSD/F-Race and I/F-Race achieve expected solution costs which are approximately 1.4% less than that of $\texttt{FFD}_\text{R}$/F-Race and $\texttt{FFD}_\text{A}$/F-Race. This observed difference is significant according to the random permutation test. However, in this case, the random permutation test cannot reject the null hypothesis that RSD/F-Race and I/F-Race achieve expected solution costs that are equivalent. On the other hand, given one-half and one-fourth of the computational budget, I/F-Race achieves expected solution cost that is approximately 0.4% less that of RSD/F-Race (observed differences are significant according to the random permutation test).

Table 3: Computational results for tuning a simulated annealing algorithm for VRP-SD. The column entries with the label `per.dev` shows the percentage deviation of each algorithms' expected solution cost from the *reference cost*: $+x$ means that the expected solution cost of the algorithm is x% more than the *reference cost* and $-x$ means that the expected solution cost of the algorithm is x% less than the *reference cost*. The column entries with the label with `max.bud` shows the maximum number of evaluations given to each algorithm and the column with the label `usd.bud` shows the average number of evaluations used by each algorithm.

algo	per.dev	max.bud	usd.bud
$\texttt{FFD}_\text{R}$/F-Race	+0.02	810	775
$\texttt{FFD}_\text{A}$/F-Race	+0.11	810	807
RSD/F-Race	−0.05	810	804
I/F-Race	−0.03	810	797
RSD/F-Race	−0.03	405	399
I/F-Race	−0.05	405	399
RSD/F-Race	+0.02	202	200
I/F-Race	−0.01	202	200
RSD/F-Race	+0.02	101	101
I/F-Race	+0.02	101	100

3.3 Tuning a simulated annealing algorithm for VRP-SD

In this study, 4 parameters of a simulated annealing algorithm have been tuned:

1. cooling rate, α;
2. a parameter used to compute the number of iterations after which the process of reheating can be applied, q;
3. another parameter used to compute the number of iterations after which the process of reheating can be applied, r;
4. parameter used in computing the starting temperature value, f;

In $\texttt{FFD}_\text{A}$/F-Race and $\texttt{FFD}_\text{R}$/F-Race, each parameter is allowed to take 3 values and in the former, the values are chosen close to the values adopted in [6]: $\alpha \in \{0.25, 0.50, 0.75\}$, $q \in \{1, 5, 10\}$, $r \in \{20, 30, 40\}$, $f \in \{0.01, 0.03, 0.05\}$. In all algorithms, the computational time allowed for evaluating a configuration on an instance is set to 10 seconds. Instances are generated as described in [6]; 400 instances were generated for tuning; 200 more instances were generated for evaluating the winning configuration.

The computational results show that, similar to the previous example, the tuning task is rather easy. Concerning the expected solution cost, the randomized permutation test cannot reject the null hypothesis that the different algorithms produce equivalent results. However, it should be noted that the main

advantage of `RSD/F-Race` and `I/F-Race` is their effectiveness under strong budget constraints: `RSD/F-Race` and `I/F-Race`, given only one-eighth of the computational budget, achieve an expected solution costs which are not significantly different from $\texttt{FFD}_{\text{R}}$`/F-Race` and $\texttt{FFD}_{\text{A}}$`/F-Race`.

4 Related works

The problem of tuning SLS algorithm is essentially a mixed variable stochastic optimization problem. Even though a number of algorithms exist for mixed variable stochastic optimization, it is quite difficult to adopt them for tuning. The primary obstacle is that, since these algorithms have parameters, tuning them is indeed paradoxical. Few procedures have been developed specifically for tuning algorithms: Kohavi and John [17] proposed an algorithm that makes use of best-first search and cross-validation for automatic parameter selection. Boyan and Moore [18] introduced a tuning algorithm based on machine learning techniques. The main emphasis of these two works is given only to the parameter value selection; there is no empirical analysis of these algorithms when applied to large number of parameters that have wide range of possible values. Audet and Orban [19] proposed a pattern search technique called mesh adaptive direct search that uses surrogate models for algorithmic tuning. In this approach, a conceptual mesh is constructed around a solution and the search for better solutions is carried around this mesh. The surrogates are used to reduce the computation time by providing an approximation to the original response surface. Nevertheless, this approach has certain number of parameters and it has never been used for tuning SLS algorithms. Adenso-Diaz and Laguna [1] designed an algorithm CALIBRA specifically for fine tuning SLS algorithms. It uses Taguchi's fractional factorial experimental designs coupled with local search. In this work, the authors explicitly mention that tuning wide range of possible values for parameters is feasible with their algorithm. However, a major limitation of this algorithm is that one cannot use it for tuning SLS algorithms with more than five parameters. Recently, Hutter et al. [20] proposed an iterated local search algorithm for parameter tuning called paramILS. This algorithm is shown to be very effective and importantly, it can be used to tune algorithms with large number of parameters.

5 Conclusions and future work

We proposed two supplementary procedures for `F-Race` that are based on random sampling, `RSD/F-Race`, and model-based search techniques, `I/F-Race`. While the adoption of full factorial design in the `F-Race` framework is impractical and computationally prohibitive when used to identify the best from a large number of parameter configurations, `RSD/F-Race` and `I/F-Race` are useful in such cases. Since the proposed approaches are quite effective under strong budget constraints, they can reduce significantly the computational time required

for tuning. However, based on the case studies, we conjecture that the expected solution cost obtained by `RSD/F-Race` and `I/F-Race` is mainly attributed to the difficulty of the tuning task.

Concerning the future research, we will extend our approach to include catogerical variables. Regarding `I/F-Race`, we will also investigate the adoption of distributions like Cauchy and some advanced techniques for updating the distribution. Finally, from the case studies that were made in the paper, we speculate that the difficulty of the tuning task depends on a number of factors such as the sensitivity of the parameters that need to be tuned and problem instances that need to be tackled. In this context, search space analysis on the parameter values is an area to investigate further.

Acknowledgments. This research has been supported by COMP^{2}SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme under contract number MEST-CT-2004-505079, and by the ANTS project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Prasanna Balaprakash and Thomas Stützle acknowledge support from the Belgian FNRS of which they are an Aspirant and a Research Associate, respectively. The information provided is the sole responsibility of the authors and does not reflect the opinion of the sponsors. The European Community is not responsible for any use that might be made of data appearing in this publication.

References

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

[2] Barr, R., Golden, B., Kelly, J., Rescende, M., Stewart, W.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics **1**(1) (1995) 9–32

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

[4] Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2004)

[5] Chiarandini, M., Birattari, M., Socha, K., Rossi-Doria, O.: An effective hybrid algorithm for university course timetabling. Journal of Scheduling **9**(5) (2006) 403–432

[6] 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 (2006)

[7] Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research **131** (2004) 373–395

[8] Maron, O., Moore, A.: Hoeffding races: Accelerating model selection search for classification and function approximation. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: Advances in Neural Information Processing Systems. Volume 6., San Francisco, CA, USA, Morgan Kaufmann (1994) 59–66

[9] Moore, A., Lee, M.: Efficient algorithms for minimizing cross validation error. In: Proceedings of the Eleventh International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann (1994) 190–198

[10] Conover, W.J.: Practical Nonparametric Statistics. Third edn. John Wiley & Sons, New York, NY, USA (1999)

[11] Hoos, H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann (2005)

[12] Stützle, T., Hoos, H.: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Future Generation Computer System **16**(8) (2000) 889–914

[13] Birattari, M., Balaprakash, P., Stützle, T., Dorigo, M.: Estimation-based local search for stochastic combinatorial optimization. Technical Report TR/IRIDIA/2007-003, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)

[14] Birattari, M.: The race package for R. Racing methods for the selection of the best. Technical Report TR/IRIDIA/2003-37, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2003) Package available at:

`http://cran.r-project.org/src/contrib/Descriptions/race.html`.

[15] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)

[16] Johnson, D.S., McGeoch, L.A., Rego, C., Glover, F.: 8th DIMACS implementation challenge (2001)

[17] Kohavi, R., John, G.: Automatic parameter selection by minimizing estimated error. In Prieditis, A., Russell, S., eds.: Proceedings of the Twelfth International Conference on Machine Learning. (1995) 304–312

[18] Boyan, J., Moore, A.: Using prediction to improve combinatorial optimization search. In: Sixth International Workshop on Artificial Intelligence and Statistics. (1997)

[19] Audet, C., Orban, D.: Finding optimal algorithmic parameters using the mesh adaptive direct search algorithm. SIAM Journal on Optimization **17**(3) (2006) 642–664

[20] Hutter, F., Hoos, H., Stützle, T.: Automatic algorithm configuration based on local search. In: AAAI-07 (to appear). (2007)