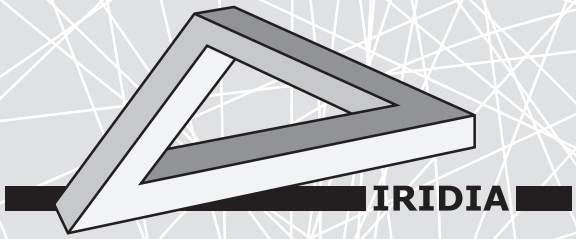


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

An Incremental Particle Swarm for Large-Scale Continuous Optimization Problems: An Example of Tuning-in-the-loop (Re)Design of Optimization Algorithms

Marco A. Montes de Oca, Doğan Aydin, and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2010-017

July 2010

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/2010-017

Revision history:
TR/IRIDIA/2010-017.001 July 2010

An Incremental Particle Swarm for Large-Scale Continuous Optimization Problems: An Example of Tuning-in-the-loop (Re)Design of Optimization Algorithms

Marco A. Montes de Oca[a] mmontes@ulb.ac.be
Doğan Aydın[b] dogan.aydin@ege.edu.tr
Thomas Stützle[a] stuetzle@ulb.ac.be

[a]IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
[b]Dept. of Computer Engineering, Ege University, Izmir, Turkey

July 2010

Abstract

The development cycle of high-performance optimization algorithms requires the designer to make several design decisions. These decisions range from implementation details to the setting of parameter values for testing intermediate designs. Proper parameter setting can be crucial for the effective assessment of algorithmic components because a bad parameter setting can make a good algorithmic component perform poorly. This situation may lead the designer to discard promising components that just happened to be tested with bad parameter settings. Automatic parameter tuning techniques are being used by practitioners to obtain peak performance from already designed algorithms. However, automatic parameter tuning also plays a crucial role during the development cycle of optimization algorithms. In this paper, we present a case study of a tuning-in-the-loop approach for redesigning a particle swarm-based optimization algorithm for tackling large-scale continuous optimization problems. Rather than just presenting the final algorithm, we describe the whole redesign process. Lastly, we study the scalability behavior of the final algorithm in the context of this special issue.

1 Introduction

The design of high-performance optimization algorithms has evolved from being almost an art to an engineering task [33,34]. Nowadays, optimization algorithms can be seen as entities made of a number of components that a designer integrates with one another in order to tackle an optimization problem. However, despite the progress made so far, there are still many decisions in the design process that are made by the designer based on intuition or limited information about the interaction between the chosen components and their behavior on the

target problem. For example, a designer may choose a mutation operator based on information gathered through some limited tests. What if the designer's choice of parameter settings during testing is unfortunate? Could one operator make the algorithm have a better average performance than another one once properly tuned?

Clearly, if the designer left all the possibilities open until a final parameter tuning phase, the design problem could be seen as a stochastic optimization problem. First steps in this direction have already been made with some very promising results, see e.g., [19, 20]. In these works, however, the high-level algorithm architecture that defines the component interactions remains fixed and is based on the extensive experience of the authors with the target problems. In this paper, we take a step back and explore the integration of automatic tuning as a decision-aid tool *during* the design loop of optimization algorithms. We see two advantages in such a tuning-in-the loop approach: (i) the parameter search space is kept to a reasonable size, and (ii) the result of tuning can help the designer to gain insight into the interactions between algorithmic components and their behavior on the optimization problem at hand. As a case study, we chose to redesign IPSOLS [23, 24], a particle swarm optimization (PSO) [18] algorithm in which the size of the population grows over time and the particles' *personal best* solutions go through a phase of improvement via a local search procedure. We describe the whole redesign process and study the final algorithm's performance scalability as the dimensionality of the problem increases. This scalability study is carried out following the protocol defined for the present special issue.

During the redesign cycle of IPSOLS, we used iterated F-Race [7] to perform an ad-hoc tuning of the algorithm's parameters on the complete benchmark function set proposed for this special issue. Tuning was performed on problem versions of low dimensionality (10 dimensions), and under the hypothesis that some structure of the problems is maintained across versions of different dimensionality; testing was carried out on versions of medium size (100 dimensions), keeping in this sense also a separation between training set used for tuning and test set for evaluation. The scalability study of the final design was carried out on the problem versions of the dimensionality proposed for this special issue (50, 100, 200, 500, and 1000 dimensions). The main contributions of this paper are the following. First, we show how automated algorithm configuration techniques can be integrated into the redesign of a generic algorithm for a specific set of benchmark functions. In particular, we detail the single steps we have followed and the insights into the algorithm behavior that have guided the algorithm engineering process. A second contribution is the final, high performing algorithm itself; in fact, an experimental comparison to other algorithms has shown that the final IPSOLS algorithm reaches very competitive results and that it outperforms well known algorithms such as the real-coded CHC algorithm (CHC) [11] and G-CMA-ES [1] on the considered set of large-scale[1] benchmark functions. Finally, a third contribution is the scalability analysis of the redesigned IPSOLS algorithm with respect to the benchmark functions' dimensionality.

The rest of this paper is structured as follows. In Section 2, we describe the original IPSOLS algorithm and the iterated F-Race algorithm. In Section 3, we present the redesign process of IPSOLS. In Section 4, we present a scalability

[1] The terms "large-scale" and "high-dimensional" are used interchangeably in this paper.

study of the new IPSOLS algorithm. Finally, in Section 5, we briefly discuss related work and present some conclusions and directions for future work.

2 IPSOLS and Iterated F-Race

In this section, we describe the algorithms used in our study. We begin by describing the basic particle swarm optimization algorithm and the IPSOLS variant. Lastly, we describe the iterated F-Race algorithm, which is the tuning algorithm used throughout this paper.

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) [18] is a stochastic optimization technique primarily used to tackle continuous optimization problems. In a PSO algorithm, a population of *particles* (called *swarm*) moves in an n-dimensional space $\Theta \subseteq \mathbb{R}^n$. The position of a particle represents a candidate solution to a continuous optimization problem with an objective function f defined over Θ, that is, $f : \Theta \to \mathbb{R}$. The positions of the particles are updated using rules aimed at placing the particles on positions associated with the lowest possible values of the objective function, that is, they try to minimize the objective function.[2]

Three vectors are associated with a particle i. First, its current position vector $\mathbf{x}_i$, which represents the latest candidate solution generated by that particle; second, its velocity vector $\mathbf{v}_i$, which stores the variation of its current position between two consecutive iterations; and third its *previous best position*, also called *personal best* vector, $\mathbf{p}_i$, which denotes the particle's best-so-far position since the beginning of a run in the search space. Additionally, each particle has a *neighborhood*, which is a subset of the swarm (usually fixed, and defined before the algorithm is run). Neighborhood relations define what is called a *population topology*. At each iteration of the algorithm, the position of each particle is updated using information from its previous best position and a neighbor's previous best position, $\mathbf{l}_i$, which is usually the one associated with the lowest value of the objective function. The update process is done on a per-component basis as follows:

$$v_{i,j}^{t+1} = w v_{i,j}^{t} + \varphi_1 U_1 \left(p_{i,j}^{t} - x_{i,j}^{t}\right) + \varphi_2 U_2 \left(l_{i,j}^{t} - x_{i,j}^{t}\right), \tag{1}$$

and

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}, \tag{2}$$

where i is the index of the particle being updated, j is the index of the coordinate being updated, $v_{i,j}^{t}$ and $x_{i,j}^{t}$ are respectively the particle's velocity and position at iteration t, $p_{i,j}^{t}$ is the particle's previous best position, $l_{i,j}^{t}$ is the best position found by the particle's neighbors, φ_1 and φ_2 are two parameters (called *acceleration coefficients*), U_1 and U_2 are two pseudorandom numbers, uniformly distributed in $[0, 1)$, that are generated at every iteration for each dimension, and w is a parameter called *inertia weight*. The components of a particle's velocity are usually constrained within the range $[-V_{\max,j}, V_{\max,j}]$.

[2] In this paper, we focus on the minimization case.

2.2 Incremental Particle Swarm Optimizer with Local Search

In the incremental particle swarm optimizer with local search (IPSOLS) [23,24], the population size grows over time. Instead of a parameter that determines the population size, IPSOLS has a parameter, k, which is referred to as *growth period*. In IPSOLS, the optimization process begins with a small population and a new particle is added to the population every k iterations until a maximum population size is reached. Each time a new particle is added, it is initialized using information from particles that are already part of the population. The initialization rule modifies the initial random position of the new particle so that it ends up closer to the best particle. The initialization rule used in IPSOLS, as applied to a new particle's jth dimension, is

$$x'_{new,j} = x_{new,j} + U \cdot (p_{best,j} - x_{new,j}), \tag{3}$$

where $x'_{new,j}$ is the new particle's updated position, $x_{new,j}$ is the new particle's original random position, $p_{best,j}$ is the best particle's previous best position, and U is a uniformly distributed random number in the range $[0, 1)$, which is the same for all dimensions. Once the rule is applied for each dimension, the new particle's previous best position is initialized to the point $\boldsymbol{x}'_{new}$ and its velocity is set to zero. The final step is to place the new particle at random within the population's topology.

The particles' previous best positions are updated using the traditional PSO rules (Eqs. 1 and 2) and through a local search procedure. Specifically, a particle's personal best is used as the initial solution from which a local search procedure is called. If the final solution found through local search is better than the calling particle's previous best position, then the local search solution becomes the calling particle's new previous best position.

In IPSOLS, the local search procedure is not necessarily called at each iteration for every particle. This is done in an effort to save function evaluations. The local search procedure is called only when an improvement of the personal best position of the calling particle is expected. For example, a potentially wasteful call to the local search procedure could result if the initial solution is in a local optimum. However, since we are dealing with black-box optimization, it is not possible to be completely certain that a particle is already in a local optimum. Therefore, in IPSOLS a heuristic approach is used to decide whether to call the local search procedure from a particle's personal best position or not. The approach consists in making the local search procedure return a value to identify its exit condition. If the procedure finishes because the maximum number of function evaluations allocated to it is reached, the local search procedure is called again in the next iteration. In other situations, for example, when a minimum objective function value threshold has been reached, or when a very small difference between two consecutive solutions (not just two consecutive function evaluations) has been detected, the local search procedure is not called again. The rationale behind these decisions is that if a difference between two solutions is very small, it is probably because the procedure has reached the bottom of a valley in the search space's landscape. On the other hand, if the maximum number of function evaluations is reached, it may be because the procedure was descending through one of the sides of a deep valley in the search space landscape, and, thus, further improvement may still be achievable.

Once all particles have gone through a potential improvement phase using

local search, the particles' positions and velocities are updated using Eqs. 1 and 2. If after updating a particle's current position a new personal best position is found, that particle's personal best position is flagged as available for improvement through local search during the next iteration of the algorithm.

2.3 Tuning Algorithm: Iterated F-Race

F-Race [5, 6] is used for selecting the best algorithm configuration (a particular setting of numerical and categorical parameters of an algorithm) from of a set of candidate configurations under stochastic evaluations. In F-Race, candidate configurations are evaluated iteratively on a sequence of problem instances. As soon as sufficient statistical evidence is gathered against a candidate configuration, it is discarded from the race. The process continues until only one candidate configuration remains, or until the maximum number of evaluations or instances is reached.

The generation of candidate configurations is independent of F-Race. A method that combines F-Race with a process capable of generating promising candidate configurations is called iterated F-Race [3, 7]. It consists in executing the steps of configuration generation, selection and refinement iteratively. For numerical parameters, the configuration generation step involves sampling from Gaussian distributions centered at promising solutions and with standard deviations that vary over time in order to focus the search around the best-so-far solutions. For categorical parameters, the configuration generation procedure samples from a discrete distribution that gives the highest probability to the values that are found in the best configurations. The process is described in detail in [7].

3 Redesigning the IPSOLS algorithm

In this section, we present the redesign process of IPSOLS. Our goal is to design an IPSOLS variant that is especially suited for large-scale continuous optimization problems. The redesign process is composed of six stages. We start by describing the tuning setup, and then we present the main results of each stage.

3.1 Tuning Setup

While tuning, problem instances are usually fed into iterated F-Race as a stream. At each step of F-Race, all surviving candidate configurations are evaluated on one additional problem instance. Once each surviving candidate configuration has been evaluated, a statistical test is applied in order to determine whether there are configurations that have, up to that point, performed significantly worse than others. First, the Friedman test is applied and, if its null hypothesis of equal performance of all surviving candidate configurations is rejected, Friedman post-tests are used to eliminate configurations that perform worse than the best one [5, 6]. In our case, we only have a limited set of 19 benchmark functions as proposed by F. Herrera *et al.* [14]. Their mathematical definition can be found in Table A.1 in the appendix. Additionally, the set of benchmark functions may result in very varied performance as they include very easy functions such as Sphere (F_1) but also much more difficult ones such as Rastrigin

(F_4) or hybrid functions (F_{12}–F_{19}). We therefore used a modified version of F-race. To do so, we define a *block evaluation* as running all (surviving) candidate configurations once on each benchmark function in a *block* [8]; a block consists of all 19 benchmark functions. After each block evaluation, the usual statistical tests in F-race are applied and inferior candidate configurations are eliminated. A cumulative average solution value is then computed and this value is used as a rough indicator of quality to determine the best candidate configurations that go to the next iteration of iterated F-race.

In all the tuning tasks described in this paper, we used the 10-dimensional versions of these functions. To have a higher chance of selecting a good candidate configuration, we launched 10 independent runs of the iterated F-Race algorithm and selected finally the one with the lowest average objective function value as the tuned configuration.

To use iterated F-Race, one needs to select the parameters to be tuned, their range or domain, and the set of instances on which tuning is to be performed. Since at each stage different parameters are tuned, we will mention them in the corresponding sections. Finally, iterated F-Race itself has a number of parameters that need to be set before it can be used. All the information regarding parameter ranges and settings used with iterated F-Race is shown in the appendix.

3.2 Stage I: Choosing a Local Search Method

When IPSOLS was originally proposed [23, 24], we used Powell's conjugate directions method [27]. In this paper, we explore the impact of the local search method on the performance of IPSOLS. In addition to Powell's conjugate directions method, we consider Powell's BOBYQA method [29]. These methods are briefly described below.

3.2.1 Conjugate Directions Method

This method tries to minimize an n-dimensional objective function by constructing a set of *conjugate directions* through a series of line searches. Directions $\boldsymbol{v}_i,\ \ i \in \{1 : n\}$, are said to be conjugate with respect to an $n \times n$ positive definite matrix A, if

$$\boldsymbol{v}_i^T A \boldsymbol{v}_j = 0, \qquad \forall i, j \in \{1 : n\},\ i \neq j\,.$$

Additionally, directions $\boldsymbol{v}_i,\ \ i \in \{1 : n\}$, must satisfy linear independence to be considered conjugate.

Conjugate search directions are attractive because if A is the Hessian matrix of the objective function, it can be minimized in exactly n line searches [30].

This method starts from an initial point $\mathbf{p}_0 \in \mathbb{R}^n$. It then performs n line searches using the unit vectors $\boldsymbol{e}_i$ as initial search directions $\boldsymbol{u}_i$. A parameter of the algorithm is the initial step size s of the search. At each step, the new initial point from which the next line search is carried out is the point where the previous line search found a relative minimum. A point $\mathbf{p}_n$ denotes the minimum discovered after all n line searches. Second, the method eliminates the first search direction by doing $\boldsymbol{u}_i = \boldsymbol{u}_{i+1},\ \forall i \in \{1 : n-1\}$, and replacing the last direction $\boldsymbol{u}_n$ for $\mathbf{p}_n - \mathbf{p}_0$. Next, a move to the minimum along the direction $\boldsymbol{u}_n$ is performed. We used an implementation of the method as described in [30].

The method terminates when the maximum number of iterations, MaxITER, is reached or when the tolerance FTol, that is, the relative change between solutions found in two consecutive iterations falls below a certain threshold.

3.2.2 Bound Constrained Optimization by Quadratic Approximation

The second local search method considered in this paper is called *bound constrained optimization by quadratic approximation* (BOBYQA) [29]. This is a derivative-free optimization algorithm based on the trust-region paradigm [10]. BOBYQA is an extension of the NEWUOA [28] algorithm that is able to deal with bound constraints. At each iteration, BOBYQA computes and minimizes a quadratic model that interpolates m points in the current trust region. These points are automatically generated by the algorithm starting from an initial guess provided by the user. Then, either the best-so-far solution, or the trust-region radius is updated. The recommended number of points to compute the quadratic model is $m = 2n + 1$ [29], where n is the dimensionality of the search space. Since the number of points BOBYQA needs to compute the model is linear with respect to the dimensionality of the search space, its author proposed BOBYQA as a method for tackling large-scale optimization problems [29]. This is the reason why we decided to study the performance of IPSOLS with BOBYQA as its local search method. NEWUOA, and by extension BOBYQA, are considered to be state-of-the-art continuous optimization techniques [2, 25].

We used the implementation that comes with NLopt, a library for nonlinear optimization [17]. The main parameter that controls this method in NLopt is the initial trust-region radius, ρ_{start}. The stopping condition depends on the values assigned to FTol, defined in the same way as in Powell's conjugate directions method, and MaxFES, which is the maximum number of function evaluations allocated for the method.

3.2.3 IPSOLS Conjugate Directions *vs.* IPSOLS BOBYQA

To measure the performance differences due to the local search methods, we compared the default and tuned versions of IPSOLS with Powell's conjugate directions method and BOBYQA. The default and the best parameter settings found through iterated F-Race are shown in Table 1.[3] We refer to the variants introduced in each stage as IPSOLS-Stage-X, where X is the stage number in which it is described.

For the default configuration of IPSOLS-Stage-I as well as of the subsequent ones, the inertia weight, w, and the two acceleration coefficients, φ_1 and φ_2, were set according to Clerc and Kennedy's analytical analysis of the PSO algorithm [9]. For other parameters, values have been set by the designers of the algorithms based on experience and on commonly used parameter settings found in the PSO literature. In the fully connected topology, labeled FC, all particles are neighbors, and in the ring topology, labeled R, particles are arranged in a periodic array and particles are neighbors of the two particles that surround them. In our experiments, all particles were neighbors of themselves. The range for the parameters ρ_{start} and s is expressed as a percentage of the size

[3] For conciseness, we present here only the most relevant results. The complete set of results can be found in this paper's companion website [22].

Table 1: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-I in 10-dimensional instances[†]

Method	Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	s/ρ_{start}	MaxITER/FES
BOBYQA	Default	0.72	1.49	1.49	1	1	FC	-1	20%	$10(2n+1)^{\ddagger}$
	Tuned	0.25	0.0	2.9	50	6	R	-3.4	10%	210
Conjugate Directions	Default	0.72	1.49	1.49	1	1	FC	-1	20%	10
	Tuned	0.0	0.0	1.82	3	6	R	-2.92	15.75%	9

[†] FC stands for fully connected, R for ring. FTol and MaxITER/FES are parameters that determine the stopping condition of the local search method. The conjugate directions method's step size and BOBYQA's trust-region radius is specified as a percentage of the length of the search space in one dimension.

[‡] n is the dimensionality of the problem.

of the search range $[x_{\min}, x_{\max}]$ in each dimension. For example, if $x_{\min} = -100$, $x_{\max} = 100$, and $\rho_{\text{start}} = 20$, then the real value that BOBYQA will use as the initial trust-region radius will be $0.2 \cdot 200 = 40$. Before calling either local search method, the value of the parameter FTol is also transformed. The real value sent to the routine is 10^{-Ftol}.

The four configurations shown in Table 1 were run on the 100-dimensional version of the 19 benchmark functions suite. Each configuration was run 25 times on each function for up to $5000n$ function evaluations, where n is the dimensionality of the function. Configurations were stopped earlier if the objective function value of the best-so-far solution was lower than 1e−14 (we refer to this number as *0-threshold*). In this situation a value of 0e+00 is reported. In Figure 1, we show the distribution of the median objective function value obtained by the resulting four configurations.

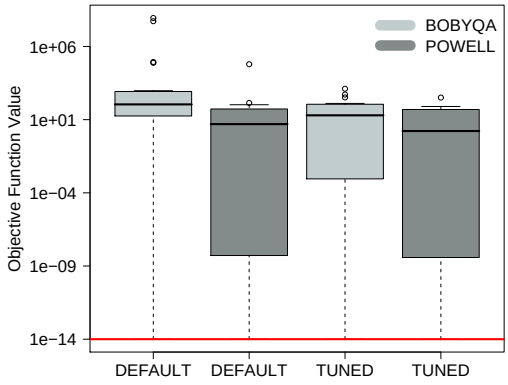


Figure 1: Box-plot of the median objective function values obtained by the four configurations listed in Table 1. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

The default as well as the tuned configurations of IPSOLS-Stage-I using Powell's conjugate directions method perform better than their BOBYQA-based counterparts. The differences between the two default and two tuned versions were found to be statistically significant at the 5% level using a Wilcoxon test ($p = 0.0008$, and $p = 0.01$, respectively). This result and the best parameter setting found for IPSOLS using Powell's conjugate directions method suggest

Table 2: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-II in 10-dimensional instances†

Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	MaxITER
Default	0.72	1.49	1.49	1	1	FC	-1	10
Tuned	0.02	2.51	1.38	85	9	FC	-11.85	10

† FC stands for fully connected, R for ring. FTol and MaxITER are parameters that determine the stopping condition of the local search method.

that line searches and a strong bias towards the best solutions (due to the setting $w = \varphi_1 = 0$) are well suited for optimizing the considered benchmark functions. In fact, these insights are exploited in the next section, where a new strategy for calling and controlling Powell's conjugate directions method is proposed.

3.3 Stage II: Changing the Strategy for Calling and Controlling the Local Search Method

The results of the tuning process presented in the previous section are very interesting because they are counterintuitive at first sight. Let us recall that the parameter w in Eq. 1 controls the influence of the velocity of particles in the computation of new candidate solutions. Setting $w = 0$ removes all influence of the particles' previous moves. The parameter φ_1 in Eq. 1 controls the strength of the attraction of a particle toward the best solution it has ever found. Setting $\varphi_1 = 0$ again removes all influence of the particles' memory on the generation of new solutions. Setting $\varphi_2 > 0$, as was the case in the experiments of the previous section, accelerates particles toward their neighbor's best position. It seems that a ring topology provided the necessary search diversification to avoid the premature convergence that a fully-connected topology would have induced.

In this second stage of the redesign process, we integrate some of these insights into the algorithm itself. We do this by changing the strategy for calling and controlling the local search method. This new strategy seeks to enhance the search around the best-so-far-solution. Two changes with respect to the original version are introduced. First, the local search procedure is no longer called from each particle's previous best position. It is now called from the best-so-far solution only. Second, the step size that controls the granularity of the local search procedure is no longer fixed, but it changes according to the state of the search. This "adaptive" step size control is implemented as follows: a particle, different from the best particle, is picked at random. The maximum norm ($||\cdot||_\infty$) of the vector that separates this random particle from the best particle is used as the local search step size. At the beginning, if the swarm size is equal to one, the step size is a random number in the range $[0, |X|)$, where $|X| = x_{max} - x_{min}$, x_{min} and x_{max} are the minimum and maximum limits of the search range. With these changes, we focus the search around the best-so-far solution with a further local search enhancement through step sizes that tend to decrease over time due to the swarm's convergence.

In Table 2, we show the default and best configurations for IPSOLS-Stage-II, which was described above.

A comparison of the results obtained with IPSOLS-Stage-I and IPSOLS-Stage-II is shown in Figure 2. Statistically significant differences between the results obtained with IPSOLS-Stage-I and IPSOLS-Stage-II are detected through

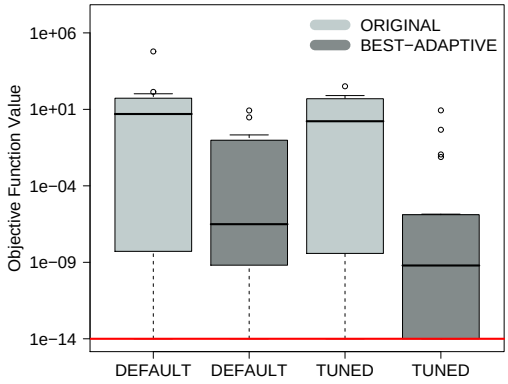


Figure 2: Box-plot of the median objective function values obtained by the two configurations listed in Table 1 of IPSOLS-Stage-I using Powell's conjugate directions method, and the two configurations listed in Table 2 of IPSOLS-Stage-II. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

the application of Wilcoxon's test. The default and tuned configurations of IPSOLS-Stage-II are better than those of IPSOLS-Stage-I ($p = 0.0006$ and $p = 0.01$, respectively).

The parameter values that correspond to the tuned configuration of IPSOLS-Stage-II are now different from those found for IPSOLS-Stage-I. The inertia weight is still very small, but the acceleration coefficient φ_1 is not. Moreover, the initial population size increased from three to 85. This is interesting because given that local search is only applied to the best particle, increasing the initial population size increases the chances of selecting a good initial solution from which to call the local search method. The increment of the growth period (k) seems to indicate that IPSOLS-Stage-II needs a greater number of iterations with a constant population size than the previous version. This may be due to the fact that the step size depends now on the spatial spread of the swarm. By having a slower population growth rate, particles have more time to converge, which allows the local search step size to decrease to levels that allow further progress. Also interesting is the fact that the parameter FTol decreased to a value of -11.85; however, it is not clear whether the local search method actually reaches such small tolerance values given the fact that the setting for the parameter MaxITER is the same as the default.

3.4 Stage III: Vectorial PSO rules

IPSOLS-Stage-III is the same as IPSOLS-Stage-II except for a modification of Eq. 1. In the original PSO algorithm and in most, if not all, variants proposed so far, particles move independently in each dimension of the search space. In contrast, IPSOLS-Stage-III uses a modification of Eq. 1 so that particles accelerate *along* the attraction vectors toward their own personal best position and

Table 3: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-III in 10-dimensional instances†

Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	MaxITER
Default	0.72	1.49	1.49	1	1	FC	-1	10
Tuned	0.0	0.0	2.34	45	10	FC	-1	5

† FC stands for fully connected, R for ring. FTol and MaxITER are parameters that determine the stopping condition of the local search method.

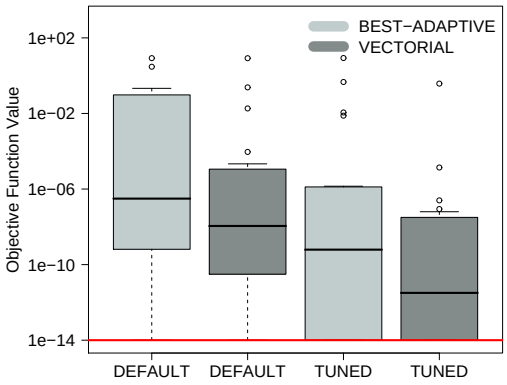


Figure 3: Box-plot of the median objective function values obtained by the two configurations listed in Table 2 of IPSOLS-Stage-II, and the two configurations listed in Table 3 of IPSOLS-Stage-III. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

their neighbor's personal best position. This modification is straightforward and consists of using the same pseudorandom numbers on all dimensions instead of generating different numbers for each dimension. This modification is inspired by the way Powell's conjugate directions method works. Once a good direction of improvement is detected, the algorithm searches along that direction. We conjectured that such a modification would provide an extra improvement on nonseparable problems while keeping a similar performance on separable ones thanks to the interaction with Powell's conjugate directions method, which always starts searching along the original coordinate system.

Table 3 lists the default and the tuned configurations of IPSOLS-Stage-III. The distributions of the median objective function values obtained with IPSOLS-Stage-II and IPSOLS-Stage-III are shown in Figure 3.

Through the use of Wilcoxon's test, it is possible to reject the null hypothesis of equal performance when comparing the default configurations ($p = 0.004$). However, it is not possible to reject the null hypothesis in the case of the tuned configurations ($p = 0.06$). The addition of the vectorial update rules does not return a significant improvement over IPSOLS-Stage-II when both versions are tuned; however, with default settings it does. We therefore keep the vectorial update rules as they may seem to make the IPSOLS-Stage-III less sensitive to different parameter values.

Table 4: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-IV in 10-dimensional instances†

Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	MaxITER
Default	0.72	1.49	1.49	1	1	FC	-1	10
Tuned	0.0	0.0	2.32	64	5	FC	-1.32	4

† FC stands for fully connected, R for ring. FTol and MaxITER are parameters that determine the stopping condition of the local search method.

3.5 Stage IV: Penalizing Bound Constraints Violation

One advantage of BOBYQA over Powell's conjugate directions method is that is has a built-in mechanism for dealing with bound constraints. In tests with IPSOLS-Stage-III, we observed that Powell's conjugate directions method would make some excursions outside the bounds. Hence, we opted for including a mechanism into IPSOLS-Stage-III that would help to enforce bound constraints; one reason to do so is that it is known that some benchmark functions may have better solutions outside the defined boundary constraints, an example being Schwefel's sine root function. In initial experiments, we noted that simply clipping solutions to the bound deteriorated the performance of the algorithm significantly. IPSOLS-Stage-IV tries to include a mechanism to enforce boundary constraints using the penalty function

$$P(\boldsymbol{x}) = \mathit{fes} \cdot \sum_{i=1}^{n} B(x_i)\,, \tag{4}$$

where $B(x_i)$ is defined as

$$B(x_i) = \begin{cases} 0, & \text{if } x_{\min} \leq x_i \leq x_{\max} \\ (x_{\min} - x_i)^2, & \text{if } x_i < x_{\min} \\ (x_{\max} - x_i)^2, & \text{if } x_i > x_{\max}\,, \end{cases} \tag{5}$$

where n is the dimensionality of the problem, x_{min} and x_{max} are the minimum and maximum limits of the search range, respectively, and *fes* is the number of function evaluations that have been used so far. The goal with this penalty function is to discourage long and far excursions outside the bounds. Strictly speaking, Eq. 4 does not change the algorithm but the objective function. Nevertheless, we describe it as if it was part of the algorithm because bound constraint handling mechanisms are important components of any algorithm.

Table 4 shows the default and tuned configurations of IPSOLS-Stage-IV. It can be seen that, as may be expected, not many changes in the tuned configuration can be found with respect to the tuned configuration of IPSOLS-Stage-III.

Figure 4 shows the box-plots of the median objective function values obtained with IPSOLS-Stage-III and IPSOLS-Stage-IV. IPSOLS-Stage-III and IPSOLS-Stage-IV differ slightly and thus we expected that their performance would be slightly different as well. This is confirmed through the application of Wilcoxon's test on the samples of results. The default and tuned versions can be considered equivalent ($p = 0.87$ and $p = 0.81$, respectively). Therefore, mainly motivated by the potential importance of being able to enforce bound constraints, we kept the design of IPSOLS-Stage-IV.S

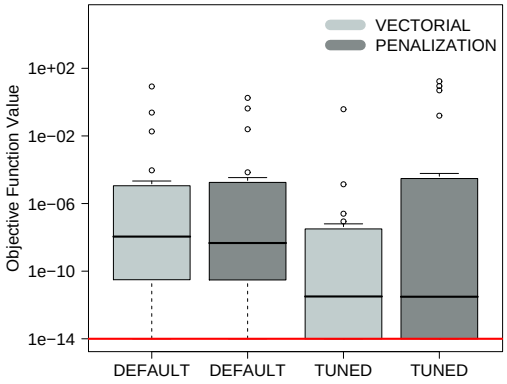


Figure 4: Box-plot of the median objective function values obtained by the two configurations listed in Table 3 of IPSOLS-Stage-III, and the two configurations listed in Table 4 of IPSOLS-Stage-IV. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

3.6 Stage V: Fighting Stagnation by Modifying the Local Search Call Strategy

We noticed in preliminary experiments that IPSOLS-Stage-IV stagnated in functions in which IPSOLS-Stage-I had very good results (e.g., F_5). A common approach to deal with stagnation is to add diversification strategies [15]. The first diversification strategy that we added to IPSOLS-Stage-IV (the second one is described in the next section) is based on the way IPSOLS-Stage-I operates. Specifically, in IPSOLS-Stage-I all particles call the local search method, but in IPSOLS-Stage-IV only the best particle does. In IPSOLS-Stage-V, we let other particles call the local search method but only sporadically. We introduce a parameter, MaxFailures, which determines the maximum number of repeated calls to the local search method from the same initial solution that does not result in a solution improvement. Each particle maintains a failures counter and when that counter reaches the value MaxFailures, the local search procedure cannot be called again from that particle's personal best position. In that situation, local search is applied from a random particle's personal best position as long as this random particle's failures counter is less than MaxFailures. If it is not the case, the algorithm continues sampling particles at random until it finds one that satisfies the requirements, or until all particles are tried. This last situation is not likely to happen because new particles are periodically added to the population. When a particle's personal best position improves thanks to a PSO move, the failures counter is reset so that the local search procedure can be called again from that newly discovered solution.

In Table 5, the default and tuned configurations of IPSOLS-Stage-V are listed. The distributions of the median objective function values of the default and tuned configurations of IPSOLS-Stage-IV and IPSOLS-Stage-V are shown

Table 5: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-V in 10-dimensional instances[†]

Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	MaxITER	MaxFailures
Default	0.72	1.49	1.49	1	1	FC	-1	10	2
Tuned	0.0	0.0	2.6	3	1	FC	-1	7	20

[†] FC stands for fully connected, R for ring. FTol and MaxITER are parameters that determine the stopping condition of the local search method.

in Figure 5.

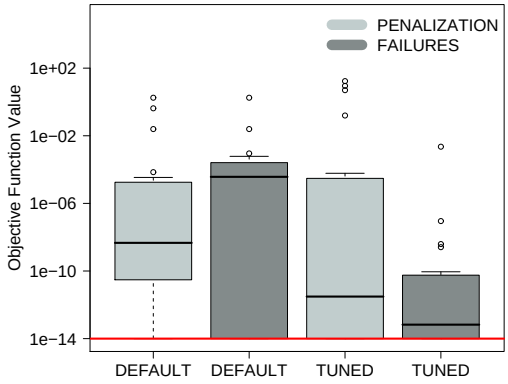


Figure 5: Box-plot of the median objective function values obtained by the two configurations listed in Table 4 of IPSOLS-Stage-IV, and the two configurations listed in Table 5 of IPSOLS-Stage-V. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

So far, we have been able to accept all modifications based on the analysis of the the distributions of median objective function values. In this case, the introduced modification with default parameter settings is worse than the unmodified version with default settings ($p = 0.03$). After tuning, the null hypothesis that assumes that both samples are drawn from the same population cannot be rejected ($p = 0.36$). It would seem clear that the proposed modification could be safely rejected. However, if we look at the distribution of the mean objective function values instead of the median[4], the situation is completely different. In that case, the default and the tuned configurations of IPSOLS-Stage-V are better than the one of IPSOLS-Stage-IV ($p = 0.002$ and $p = 0.01$, respectively). This result indicates that the introduced modification indeed reduces the likelihood of IPSOLS-Stage-IV of stagnating, but this is seen only in the upper quartiles of the solution quality distributions generated by the algorithm, that is, the worst results are improved but not necessarily the best. In fact, the lower quartiles of these distributions (i.e., the best solutions) are either deteriorated, as with the default configuration, or not affected, as it seems to be the case with the tuned configuration.

[4]We remind the reader that the complete set of results can be found in [22].

Table 6: Default and best configuration obtained through iterated F-Race for IPSOLS-Stage-VI in 10-dimensional instances†

Configuration	w	φ_1	φ_2	Init. pop. size	k	Topology	FTol	MaxITER	MaxFailures	MaxStagIter
Default	0.72	1.49	1.49	1	1	FC	-1	10	2	10
Tuned	0.0	0.84	1.85	1	1	FC	-1	87	13	27

† FC stands for fully connected, R for ring. FTol and MaxITER are parameters that determine the stopping condition of the local search method.

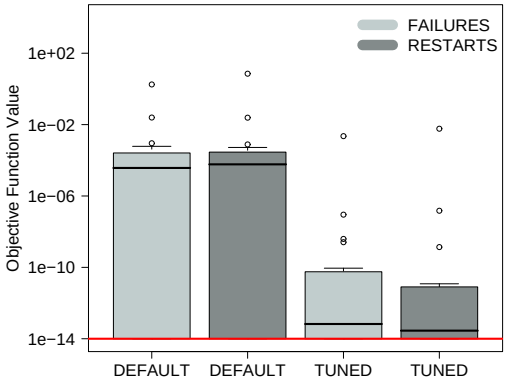


Figure 6: Box-plot of the median objective function values obtained by the two configurations listed in Table 5 of IPSOLS-Stage-V, and the two configurations listed in Table 6 of IPSOLS-Stage-VI. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

3.7 Stage VI: Fighting Stagnation with Restarts

In IPSOLS-Stage-VI, we included an algorithm-level diversification strategy. This strategy consists in restarting the algorithm but keeping the best-so-far solution. In particular, the best-so-far solution becomes the new first particle's current and previous best position. The restart criterion is the number of PSO-level consecutive iterations with a relative solution improvement lower than a certain threshold ϵ. In our experiments, we set ϵ = *0-threshold*. The number of consecutive iterations without significant improvement is a parameter of IPSOLS-Stage-VI, MaxStagIter.

The list of default and tuned parameter settings for IPSOLS-Stage-VI is shown in Table 6. The distributions of the median objective function values obtained by the default and tuned configurations for IPSOLS-Stage-V and IPSOLS-Stage-VI are shown in Figure 6.

Comparing the aggregated data of IPSOLS-Stage-V and IPSOLS-Stage-VI using Wilcoxon tests, the null hypothesis cannot be rejected neither with medians nor with means. While the actual performance of IPSOLS-Stage-VI is statistically equivalent to that of IPSOLS-Stage-V with the 100-dimensional versions of the benchmark functions, we prefer to keep the extra diversification layer that a strong restart offers. It will be seen later during the scalability tests

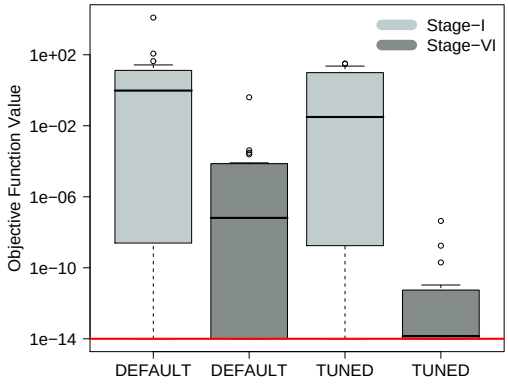


Figure 7: Box-plot of the median objective function values obtained by the two configurations listed in Table 1 of IPSOLS-Stage-I using Powell's conjugate directions method, and the two configurations listed in Table 6 of IPSOLS-Stage-VI. In all cases, we used the 100-dimensional versions of the benchmark functions proposed for this special issue. The line at the bottom represents the 0-threshold.

that this extra diversification layer does not hurt performance and provides flexibility that can be exploited in application scenarios beyond the current special issue, for example, if an even larger number of function evaluations could be used.

With IPSOLS-Stage-VI, we conclude the redesign cycle of IPSOLS. In Figure 7 we can clearly see the improvement with respect to the original algorithm that we were able to obtain through a tuning-in-the-loop redesign process.

It is interesting to see the positive effects that tuning had on most of the tested variants of IPSOLS. It should not be forgotten that the tuning process was performed with instances whose size was only 10% the size of the instances over which we tested the effectiveness of tuning. In the following section, we will see that the effectiveness of tuning extends also to instances that are up to 100 times larger than the ones seen during tuning.

4 Performance Scalability Study

In this section, we study the performance of IPSOLS-Stage-VI, which for simplicity we refer to as IPSOLS again, on the set of 19 benchmark functions proposed for this special issue. The study is perfomed on the 50-, 100-, 200-, 500-, and 1000-dimensional versions of these functions. While perhaps only the 500- and 1000-dimensional versions can really be called "high-dimensional", the study is also designed to determine the scalability of IPSOLS's performance as the dimensionality of the objective function increases.

4.1 Reference Algorithms

The performance of the tuned and non-tuned versions of IPSOLS was compared to that of differential evolution (DE) [32], the CHC algorithm [11], and the CMA-ES algorithm with increasing population size (G-CMA-ES) [1]. In particular, DE and G-CMA-ES are considered to be state-of-the-art algorithms for continuous optimization problems. The parameter settings used with these algorithms as well as summary statistics of their performance on the benchmark functions suite used for this special issue are available at [21]. For more detailed information on these algorithms, we refer the reader to the original sources.

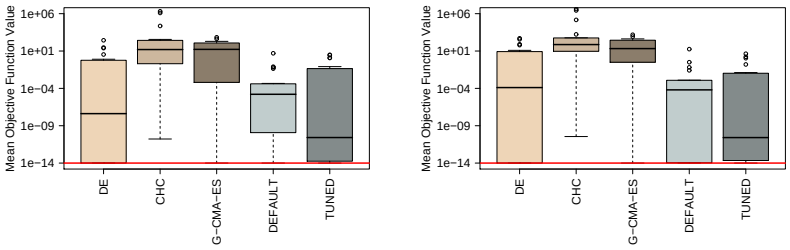
4.2 Solution Quality Scalability

Let us first report the main trends of the computational results. We use box-plots that indicate the distribution of the average objective function value obtained by each algorithm on the 19 benchmark functions suite. These box-plots are shown in Figure 8.

Both CHC and G-CMA-ES obtain relatively high average objective function values in all dimensionalities with respect to DE, and both configurations of IPSOLS. Comparing CHC and G-CMA-ES with both configurations of IPSOLS, it appears that the performance differences increase with increasing dimensionality. This fact can be noted, for example, in the much lower medians of the average objective values for dimensions 500 and 1 000. To compare IPSOLS with DE, which is the most competitive reference algorithm, we need to perform a statistical analysis.

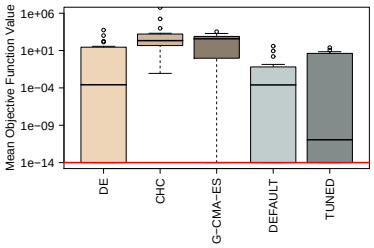
We tested the null hypothesis with Wilcoxon's test at the 5% a confidence level. When using the average objective function values, we cannot reject the null hypothesis in any of the five function dimensionalities. However, if we consider the median objective function values, the situation is different. With the default configuration of IPSOLS the null hypothesis can be rejected only in the 1000-dimensional case ($p = 0.019$). With the tuned configuration of IPSOLS, however, the null hypothesis can be rejected in all five cases ($p = 0.003$, $p = 0.001$, $p = 0.001$, $p = 0.005$, and $p = 0.005$ for the 50-, 100-, 200-, 500-, and 1000-dimensional cases, respectively). To understand why the tuned configuration of IPSOLS can outperform DE in the median case but not in the average case, we need to look at the numerical values given in Tables 7 to 13.

There is a pattern in the results obtained with IPSOLS that is present in many functions across dimensionalities. This pattern is the following: either IPSOLS or its tuned configuration obtain the lowest minimum and median values but its maximum value is higher than other algorithms' maximum value. The result is a high average solution value. Take as an extreme example the results obtained by the tuned configuration of IPSOLS with function F_{11} in its 100-dimensional version. The minimum and median values obtained by the algorithm are both 0e+00. The maximum value is 3.70e−02, which averaged over 25 runs gives 1.48e−03. This number is exactly the one reported in the corresponding entry in Table 10. This means that in one single run out of the 25 executed, or in other words, with a probability of approximately 0.04, IPSOLS stagnated when solving F_{11}. However, there are data that make us believe that this problem can be solved by further tuning the parameters of the algorithm on specific problems. For example, in functions F_9 and F_{13} the best results are

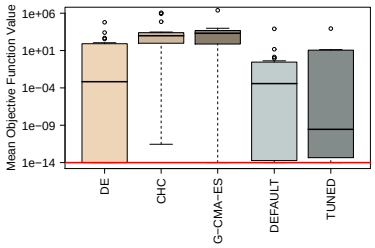


(a) 50 dimensions

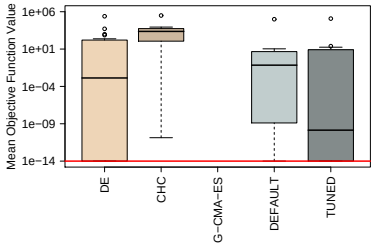
(b) 100 dimensions



(c) 200 dimensions



(d) 500 dimensions



(e) 1000 dimensions

Figure 8: The box-plots show the distribution of the average objective function values obtained with DE, CHC, G-CMA-ES, Default IPSOLS (labeled "Default"), and Tuned IPSOLS (labeled "Tuned") on the 19 benchmark functions. In the plot that corresponds to 1000 dimensions, the results obtained with G-CMA-ES are missing due to this algorithm's excessive computation time for this dimensionality. The line at the bottom of each plot represents the 0-threshold.

either obtained by the default or the tuned configuration of IPSOLS. We tested IPSOLS on functions F_{11} and $F14$ with a maximum budget equal to $1000n$, that is, twice as many function evaluations as the protocol defined for this special issue. In that case, IPSOLS is able to find the optimal solution in all 25 runs in the case of function F_{11} and to reduce the value of the worst solution found for F_{14}. It is in this kind of situations that the second-level restart strategy introduced in IPSOLS provides advantages.

Table 7: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_1			F_2			F_3					
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.60e−01	3.29e−01	8.49e−01	2.56e−01	2.89e+01	2.90e+01	3.10e+01	2.55e+01
	CHC	1.67e−11	1.63e−11	2.33e−11	1.23e−11	6.19e+01	5.89e+01	8.43e+01	5.13e+01	1.25e+06	1.33e+02	2.01e+07	9.74e−01
	G-CMA-ES	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.75e−11	2.64e−11	3.82e−11	2.08e−11	7.97e−01	**0.00e+00**	3.99e+00	**0.00e+00**
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.47e−04	8.14e−05	7.48e−04	2.19e−06	2.35e−08	**0.00e+00**	5.88e−07	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**2.56e−14**	**1.42e−14**	**8.53e−14**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
100	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	4.45e+00	4.34e+00	5.59e+00	3.82e+00	8.01e+01	7.81e+01	1.25e+02	7.55e+01
	CHC	3.56e−11	3.54e−11	4.80e−11	2.64e−11	8.58e+01	8.58e+01	9.74e+01	7.30e+01	4.19e+06	5.89e+03	7.26e+07	9.12e+01
	G-CMA-ES	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.51e−10	1.62e−10	2.72e−10	7.71e−11	3.88e+00	2.27e+00	1.84e+01	**0.00e+00**
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.59e−04	1.18e−04	8.03e−04	7.22e−06	**0.00e+00**	**0.00e+00**	1.59e−14	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**3.42e−14**	**2.84e−14**	**1.14e−13**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
200	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.92e+01	1.93e+01	2.10e+01	1.74e+01	1.78e+02	1.77e+02	2.27e+02	1.74e+02
	CHC	8.34e−01	1.81e−11	2.09e+01	1.36e−11	1.03e+02	1.02e+02	1.15e+02	9.34e+01	2.01e+07	1.12e+05	4.31e+08	2.07e+02
	G-CMA-ES	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.16e−09	9.91e−10	5.79e−09	4.90e−10	8.91e+01	8.95e+01	1.19e+02	**0.00e+00**
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.49e−04	1.80e−04	7.28e−04	1.69e−05	**0.00e+00**	**0.00e+00**	2.09e−14	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**3.23e−14**	**2.84e−14**	**1.42e−13**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
500	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	5.35e+01	5.33e+01	5.59e+01	5.13e+01	4.76e+02	4.74e+02	5.22e+02	4.70e+02
	CHC	2.84e−12	2.67e−12	4.38e−12	1.93e−12	1.29e+02	1.28e+02	1.41e+02	1.16e+02	1.14e+06	6.57e+02	2.85e+07	4.94e+02
	G-CMA-ES	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.48e−04	3.31e−04	5.70e−04	1.52e−04	3.58e+02	3.55e+02	8.31e+02	2.50e+02
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.77e−04	2.80e−04	1.66e−03	5.16e−06	3.81e−12	**0.00e+00**	8.85e−11	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**8.07e−14**	**4.97e−14**	**3.55e−13**	**1.07e−14**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
1000	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	8.46e+01	8.44e+01	8.65e+01	8.22e+01	9.69e+02	9.69e+02	9.71e+02	9.66e+02
	CHC	1.36e−11	1.30e−11	2.33e−11	7.56e−12	1.44e+02	1.44e+02	1.57e+02	1.38e+02	8.75e+03	1.49e+03	1.80e+05	1.22e+03
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	4.51e−04	3.70e−04	1.46e−03	1.14e−05	7.73e+00	**0.00e+00**	1.57e+02	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**6.68e−14**	**5.68e−14**	**2.13e−13**	**1.42e−14**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**

Table 8: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_4			F_5			F_6					
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	3.98e−02	1.51e−13	9.95e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.43e−13	1.42e−13	1.71e−13	1.14e−13
	CHC	7.43e+01	7.36e+01	1.00e+02	5.47e+01	1.67e−03	1.32e−11	2.21e−02	9.92e−12	6.15e−07	6.27e−07	7.33e−07	4.72e−07
	G-CMA-ES	1.05e+02	1.08e+02	1.33e+02	7.16e+01	2.96e−04	**0.00e+00**	7.40e−03	**0.00e+00**	2.09e+01	2.11e+01	2.12e+01	2.00e+01
	IPSOLS	7.96e−02	**0.00e+00**	1.99e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.08e−12	**0.00e+00**	5.20e−011	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	6.72e−03	**0.00e+00**	1.41e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
100	DE	7.96e−02	4.23e−13	9.95e−01	1.37e−13	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.10e−13	3.13e−13	3.41e−13	2.84e−13
	CHC	2.19e+02	2.17e+02	2.93e+02	1.64e+02	3.83e−03	9.89e−12	4.16e−02	7.16e−12	4.10e−07	4.12e−07	5.46e−07	3.02e−07
	G-CMA-ES	2.50e+02	2.50e+02	3.50e+02	1.86e+02	1.58e−03	**0.00e+00**	1.48e−02	**0.00e+00**	2.12e+01	2.13e+01	2.14e+01	2.00e+01
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.53e−12	**0.00e+00**	4.01e−11	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.31e−02	**0.00e+00**	3.27e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
200	DE	1.27e−01	3.58e−12	9.95e−01	7.44e−13	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	6.54e−13	6.54e−13	7.11e−13	5.97e−13
	CHC	5.40e+02	5.22e+02	7.72e+02	4.02e+02	8.76e−03	7.40e−03	4.67e−02	4.41e−12	1.23e+00	1.30e+00	4.41e+00	1.30e−07
	G-CMA-ES	6.48e+02	6.68e+02	8.31e+02	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.14e+01	2.14e+01	2.15e+01	2.14e+01
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
500	DE	3.20e−01	9.22e−03	5.22e+02	4.64e−12	**0.00e+00**	**0.00e+00**	2.25e+00	**0.00e+00**	1.65e−12	1.65e−12	**0.00e+00**	1.59e−12
	CHC	1.91e+03	1.91e+03	2.22e+03	1.46e+03	6.98e−03	1.99e−13	4.42e−02	8.53e−14	5.16e+00	5.18e+00	8.05e+00	2.83e+00
	G-CMA-ES	2.10e+03	2.07e+03	2.31e+03	1.88e+03	2.96e−04	**0.00e+00**	7.40e−03	**0.00e+00**	2.15e+01	2.15e+01	2.16e+01	2.15e+01
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Tuned IPSOLS	2.82e−03	**0.00e+00**	7.04e−02	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
1000	DE	1.44e+00	1.32e+00	4.69e+00	2.76e−11	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.29e−12	3.30e−12	3.41e−12	3.18e−12
	CHC	4.76e+03	4.73e+03	5.36e+03	4.13e+03	7.02e−03	1.71e−13	3.83e−02	1.14e−13	1.38e+01	1.40e+01	1.63e+01	1.03e+01
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	3.98e−02	**0.00e+00**	9.95e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**

Table 9: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_7				F_8				F_9			
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.44e+00	3.54e+00	4.62e+00	1.89e+00	2.73e+02	2.73e+02	2.74e+02	2.72e+02
	CHC	2.66e−09	1.40e−09	9.92e−09	4.58e−10	2.24e+02	1.75e+02	6.27e+02	3.19e+01	3.10e+02	3.08e+02	3.20e+02	2.99e+02
	G-CMA-ES	11.01e−10	7.67e−11	2.32e−10	6.16e−11	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.66e+01	1.61e+01	3.36e+01	4.38e+00
	IPSOLS	**0.00e+00**	**0.00e+00**	1.08e−14	**0.00e+00**	7.62e−08	6.45e−08	2.68e−07	1.55e−08	6.50e−02	4.14e−04	1.61e+00	8.53e−05
	Tuned IPSOLS	4.98e−12	**0.00e+00**	1.25e−10	**0.00e+00**	4.78e−09	1.75e−09	2.36e−08	2.59e−10	**4.95e−06**	**0.00e+00**	**1.24e−04**	**0.00e+00**
100	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.69e+02	3.47e+02	4.84e+02	2.86e+02	5.06e+02	5.06e+02	5.07e+02	5.04e+02
	CHC	1.40e−02	7.56e−10	3.50e−01	2.05e−10	1.69e+03	1.66e+03	3.26e+03	9.32e+02	5.86e+02	5.85e+02	5.99e+02	5.69e+02
	G-CMA-ES	4.22e−04	6.98e−07	9.41e−03	2.78e−09	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.02e+02	1.06e+02	1.56e+02	4.31e+01
	IPSOLS	1.16e−14	**0.00e+00**	1.82e−13	**0.00e+00**	2.59e−02	2.44e−02	5.80e−02	1.18e−02	**6.52e−02**	7.74e−04	**1.61e+00**	2.37e−05
	Tuned IPSOLS	3.00e−12	1.53e−14	1.95e−11	**0.00e+00**	7.18e−03	5.89e−03	1.64e−02	2.13e−03	1.09e+00	**0.00e+00**	2.74e+01	**0.00e+00**
200	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	5.53e+03	5.33e+03	6.63e+03	4.82e+03	1.01e+03	1.01e+03	1.01e+03	1.01e+03
	CHC	2.59e−01	1.90e−08	2.65e+00	1.20e−09	9.38e+03	9.33e+03	1.49e+04	5.91e+03	1.19e+03	1.19e+03	1.22e+03	1.16e+03
	G-CMA-ES	1.17e−01	2.61e−02	7.85e−01	4.62e−05	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.75e+02	3.81e+02	4.92e+02	2.95e+02
	IPSOLS	1.18e−14	**0.00e+00**	4.77e−14	**0.00e+00**	3.88e+01	3.44e+01	6.16e+01	2.71e+01	**2.09e−03**	1.97e−03	**7.27e−03**	3.63e−04
	Tuned IPSOLS	1.14e−11	3.49e−14	9.29e−11	**0.00e+00**	2.47e+01	2.33e+01	3.75e+01	1.74e+01	4.09e+00	**0.00e+00**	6.24e+01	**0.00e+00**
500	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	6.09e+04	6.11e+04	6.51e+04	5.51e+04	2.52e+03	2.52e+03	2.53e+03	2.52e+03
	CHC	1.27e−01	6.04e−08	1.66e+00	7.76e−09	7.22e+04	7.24e+04	8.86e+04	6.09e+04	3.00e+03	3.00e+03	3.03e+03	2.97e+03
	G-CMA-ES	7.21e+153	2.14e+143	2.45e+156	1.23e+124	**2.36e−06**	**2.31e−06**	**3.91e−06**	**7.68e−07**	1.74e+03	1.76e+03	1.85e+03	1.58e+03
	IPSOLS	2.72e−14	1.07e−14	1.49e−13	**0.00e+00**	8.05e+03	8.08e+03	1.03e+04	6.63e+03	**1.49e−01**	6.52e−03	**1.17e+00**	6.88e−04
	Tuned IPSOLS	1.06e−11	3.94e−14	1.73e−10	**0.00e+00**	8.64e+03	8.72e+03	1.14e+04	6.26e+03	1.13e+01	**0.00e+00**	1.30e+02	**0.00e+00**
1000	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.46e+05	2.46e+05	2.58e+05	2.31e+05	5.13e+03	5.13e+03	5.14e+03	5.12e+03
	CHC	3.52e−01	1.88e−03	2.90e+00	2.28e−07	3.11e+05	3.13e+05	3.43e+05	2.61e+05	6.11e+03	6.11e+03	6.16e+03	6.06e+03
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	6.99e−14	**0.00e+00**	3.97e−13	**0.00e+00**	**9.88e+04**	**9.97e+04**	**1.16e+05**	**7.58e+04**	**8.31e+00**	2.12e−02	9.59e+01	9.28e−04
	Tuned IPSOLS	7.13e−11	2.04e−13	5.78e−10	**0.00e+00**	1.20e+05	1.19e+05	1.59e+05	9.82e+04	9.76e+00	**0.00e+00**	**8.41e+01**	**0.00e+00**

Table 10: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_{10}				F_{11}				F_{12}			
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**6.23e−05**	5.60e−05	**1.04e−04**	3.35e−05	**5.35e−13**	**5.27e−13**	**7.19e−13**	**2.72e−13**
	CHC	7.30e+00	7.03e+00	1.62e+01	4.62e−11	2.16e+00	1.12e−02	1.34e+01	3.68e−04	9.57e−01	8.30e−11	2.39e+01	6.04e−11
	G-CMA-ES	6.81e+00	6.71e+00	1.26e+01	2.10e+00	3.01e+01	2.83e+01	6.94e+01	7.83e+00	1.88e+02	1.87e+02	2.51e+02	1.15e+02
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.55e−04	2.46e−04	8.63e−04	8.36e−05	8.58e−05	5.21e−05	2.35e−04	5.23e−10
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	8.19e−02	**0.00e+00**	2.05e+00	**0.00e+00**	1.17e−11	1.02e−12	1.95e−10	5.71e−13
100	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**1.28e−04**	1.29e−04	**1.70e−04**	7.89e−05	5.99e−11	6.18e−11	**8.27e−11**	3.42e−11
	CHC	3.30e+01	2.52e+01	9.61e+01	1.42e+01	7.32e+01	6.84e+01	1.55e+02	1.64e+00	1.03e+01	7.59e−09	5.19e+01	6.43e−11
	G-CMA-ES	1.66e+01	1.08e+01	2.46e+01	9.28e+00	1.64e+02	1.51e+02	2.60e+02	8.07e+01	4.17e+02	4.20e+02	4.77e+02	3.46e+02
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.17e−03	5.25e−04	3.89e−03	2.22e−04	2.34e−04	1.84e−04	6.28e−04	5.07e−10
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.48e−03	**0.00e+00**	3.70e−02	**0.00e+00**	**2.58e−11**	**2.93e−12**	5.65e−10	**1.96e−12**
200	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**2.62e−04**	2.59e−04	**3.10e−04**	2.26e−04	9.76e−10	9.36e−10	1.49e−09	6.65e−10
	CHC	7.13e+01	6.60e+01	1.42e+02	3.57e+01	3.85e+02	3.68e+02	5.96e+02	1.19e+02	7.44e+01	7.47e+01	1.59e+02	1.60e+01
	G-CMA-ES	4.43e+01	4.41e+01	5.85e+01	3.04e+01	8.03e+02	7.93e+02	1.04e+03	6.37e+02	9.06e+02	9.08e+02	1.01e+03	8.30e+02
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	4.86e−02	1.67e−03	1.16e+00	1.98e−04	5.51e−04	4.01e−04	1.81e−03	1.84e−04
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	6.98e+00	**0.00e+00**	7.51e+01	**0.00e+00**	**6.65e−12**	**6.11e−12**	**1.03e−11**	**4.65e−12**
500	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**6.76e−04**	6.71e−04	**7.83e−04**	6.13e−04	7.07e−09	6.98e−09	9.29e−09	5.95e−09
	CHC	1.86e+02	1.64e+02	5.18e+02	1.08e+02	1.81e+03	1.77e+03	2.47e+03	1.50e+03	4.48e+02	4.49e+02	5.52e+02	3.63e+02
	G-CMA-ES	1.27e+02	1.27e+02	1.55e+02	1.03e+02	4.16e+03	4.18e+03	4.54e+03	3.50e+03	2.58e+03	2.59e+03	2.76e+03	2.41e+03
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	4.21e−01	3.95e−03	1.04e+01	4.45e−04	1.18e−03	9.13e−04	3.58e−03	1.09e−04
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.14e+01	**0.00e+00**	1.92e+02	**0.00e+00**	**1.93e−11**	**1.88e−11**	**2.72e−11**	**1.49e−11**
1000	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**1.35e−03**	1.35e−03	**1.48e−03**	1.25e−03	**1.68e−08**	1.70e−08	**1.94e−08**	1.40e−08
	CHC	3.83e+02	3.17e+02	7.40e+02	2.15e+02	4.82e+03	4.83e+03	5.42e+03	4.42e+03	1.05e+03	1.04e+03	1.21e+03	8.80e+02
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	4.72e+00	1.65e−02	3.56e+01	3.06e−03	6.15e−01	2.54e−03	1.53e+01	4.20e−04
	Tuned IPSOLS	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	9.75e+00	**1.62e−06**	7.80e+01	**0.00e+00**	1.37e+00	**4.09e−11**	1.53e+01	**3.50e−11**

Table 11: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_{13}				F_{14}				F_{15}			
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	2.45e+01	2.44e+01	2.64e+01	2.28e+01	**4.16e−08**	2.58e−08	**1.80e−07**	1.32e−08	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	2.08e+06	6.90e+01	5.08e+07	1.14e+01	6.17e+01	5.58e+01	1.48e+02	3.99e+01	3.98e−01	6.05e−05	2.12e+00	1.11e−08
	G-CMA-ES	1.97e+02	1.97e+02	2.32e+02	1.36e+02	1.09e+02	1.05e+02	1.50e+02	7.42e+01	9.79e−04	8.12e−04	3.85e−03	1.73e−04
	IPSOLS	4.62e−04	1.32e−05	9.01e−03	1.34e−06	1.61e−05	1.45e−05	5.49e−05	9.32e−07	5.12e−10	4.43e−10	1.40e−09	1.77e−11
	Tuned IPSOLS	**2.65e−10**	**2.00e−10**	**1.42e−09**	**1.52e−11**	1.18e+00	**1.77e−12**	9.95e+00	**5.01e−13**	2.62e−11	1.07e−11	1.96e−10	2.13e−14
100	DE	6.17e+01	6.17e+01	6.45e+01	5.95e+01	4.79e−02	1.30e−07	9.95e−01	5.65e−08	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	2.70e+06	2.43e+02	5.61e+07	2.41e+01	1.66e+02	1.65e+02	2.24e+02	1.19e+02	8.13e+00	4.55e+00	6.42e+01	2.42e−08
	G-CMA-ES	4.21e+02	4.12e+02	5.52e+02	3.48e+02	2.55e+02	2.52e+02	3.04e+02	2.16e+02	6.30e−01	4.13e−01	2.51e+00	2.39e−04
	IPSOLS	**1.25e−03**	7.35e−05	**1.16e−02**	2.76e−06	**6.31e−05**	5.93e−05	**1.55e−04**	7.77e−06	1.45e−09	1.06e−09	4.16e−09	2.71e−11
	Tuned IPSOLS	1.59e−01	**1.37e−09**	3.99e+00	**5.94e−11**	4.34e+00	**3.66e−12**	3.08e+01	**1.78e−12**	2.89e−11	9.03e−12	1.96e−10	**0.00e+00**
200	DE	1.36e+02	1.36e+02	1.38e+02	1.34e+02	1.38e−01	2.71e−07	**9.95e−01**	1.24e−07	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	5.75e+06	3.35e+02	1.01e+08	1.62e+02	4.29e+02	4.25e+02	5.14e+02	3.52e+02	2.14e+01	1.24e+01	1.23e+02	7.42e−08
	G-CMA-ES	9.43e+02	9.34e+02	1.08e+03	8.02e+02	6.09e+02	6.24e+02	7.05e+02	5.08e+02	1.75e+00	2.10e+00	4.92e+00	4.81e−03
	IPSOLS	**1.39e−01**	9.95e−05	**2.23e+00**	7.78e−06	**7.97e−02**	9.29e−05	1.99e+00	1.72e−05	3.21e−09	2.06e−09	1.09e−08	1.36e−10
	Tuned IPSOLS	2.55e+00	**6.97e−09**	6.22e+01	**1.77e−09**	4.27e+00	**7.70e−12**	5.57e+01	**4.67e−12**	1.08e−10	3.29e−11	9.27e−10	**0.00e+00**
500	DE	3.59e+02	3.58e+02	3.78e+02	3.57e+02	1.35e−01	9.01e−07	1.12e+00	5.55e−07	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	3.22e+07	1.54e+03	2.21e+08	3.35e+02	1.46e+03	1.45e+03	1.77e+03	1.15e+03	6.01e+01	4.01e+01	2.66e+02	6.53e+00
	G-CMA-ES	2.87e+03	2.87e+03	3.55e+03	2.59e+03	1.95e+03	1.95e+03	2.15e+03	1.80e+03	2.82e+262	5.57e+258	3.93e+263	1.25e+217
	IPSOLS	**9.00e−01**	2.66e−04	**2.24e+01**	4.26e−05	**4.00e−02**	1.56e−04	**9.95e−01**	2.93e−05	1.00e−08	1.04e−08	2.48e−08	3.00e−10
	Tuned IPSOLS	4.14e+00	**4.25e−08**	6.92e+01	**6.50e−09**	1.30e+01	**1.52e−10**	9.34e+01	**1.54e−11**	2.83e−10	7.72e−11	1.86e−09	2.48e−12
1000	DE	7.30e+02	7.29e+02	7.31e+02	7.28e+02	**6.90e−01**	9.95e−01	**2.77e+00**	1.30e−06	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	6.66e+07	1.80e+03	1.67e+09	1.32e+03	3.62e+03	3.65e+03	3.98e+03	3.21e+03	8.37e+01	7.76e+01	1.31e+02	2.83e+01
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	1.07e+01	3.92e−03	1.08e+02	9.62e−05	2.79e+00	4.92e−04	2.31e+01	9.98e−05	1.57e−08	1.23e−08	4.74e−08	1.71e−09
	Tuned IPSOLS	**1.28e+00**	**1.50e−07**	**2.92e+01**	**2.01e−08**	1.78e+01	**2.57e−09**	1.56e+02	**3.68e−11**	1.39e−10	6.32e−11	5.72e−10	5.02e−12

Table 12: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function											
		F_{16}				F_{17}				F_{18}			
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
50	DE	**1.56e−09**	1.51e−09	**2.68e−09**	9.33e−10	**7.98e−01**	6.83e−01	**2.24e+00**	1.18e−02	**1.22e−04**	1.20e−04	**2.36e−04**	6.13e−05
	CHC	2.95e−09	2.37e−09	9.17e−09	**0.00e+00**	2.26e+04	7.89e+01	5.59e+05	9.55e−01	1.58e+01	1.39e+01	2.94e+01	3.98e+00
	G-CMA-ES	4.27e+02	4.22e+02	5.27e+02	3.19e+02	6.89e+02	6.71e+02	8.03e+02	5.96e+02	1.31e+02	1.27e+02	1.59e+02	1.13e+02
	IPSOLS	2.91e−04	3.02e−04	6.14e−04	2.04e−05	4.97e+00	4.02e−01	5.17e+01	9.83e−04	4.38e−02	6.53e−05	9.95e−01	6.20e−09
	Tuned IPSOLS	2.80e+00	**3.08e−12**	4.03e+01	2.43e−12	3.10e+00	**4.35e−08**	4.49e+01	**1.25e−09**	1.24e+00	**8.06e−12**	8.95e+00	**3.97e−12**
100	DE	4.27e+02	4.22e+02	5.27e+02	3.19e+02	6.89e+02	6.71e+02	8.03e+02	5.96e+02	1.31e+02	1.27e+02	1.59e+02	1.13e+02
	CHC	2.23e+01	1.65e+01	6.71e+01	1.05e+00	1.47e+05	2.43e+02	1.78e+06	4.14e+01	7.00e+01	6.85e+01	9.51e+01	3.97e+01
	G-CMA-ES	8.59e+02	8.48e+02	9.75e+02	7.47e+02	1.51e+03	1.52e+03	1.74e+03	1.27e+03	3.07e+02	3.13e+02	3.41e+02	2.62e+02
	IPSOLS	4.10e−04	3.90e−04	1.04e−03	1.01e−05	1.79e+01	7.13e+00	7.61e+01	6.28e−03	**3.19e−01**	1.79e−04	**4.98e+00**	3.30e−05
	Tuned IPSOLS	**9.09e−12**	**7.06e−12**	**4.46e−11**	**5.83e−12**	**8.16e−03**	**1.49e−07**	**2.04e−01**	**2.29e−08**	1.68e+00	**1.20e−11**	1.39e+01	**9.98e−12**
200	DE	**7.46e−09**	7.26e−09	**9.60e−09**	5.54e−09	3.70e+01	3.70e+01	**3.95e+01**	3.49e+01	**4.73e−04**	4.70e−04	**6.07e−04**	2.94e−04
	CHC	1.60e+02	1.55e+02	2.94e+02	6.19e+00	1.75e+05	4.15e+02	4.37e+06	2.64e+02	2.12e+02	2.15e+02	2.61e+02	1.63e+02
	G-CMA-ES	1.92e+03	1.90e+03	2.12e+03	1.66e+03	3.36e+03	3.33e+03	3.84e+03	3.07e+03	6.89e+02	6.88e+02	7.35e+02	6.42e+02
	IPSOLS	1.05e−03	9.71e−04	2.54e−03	2.58e−04	**8.74e+00**	1.22e+00	5.91e+01	5.18e−03	1.60e+00	3.56e−04	1.37e+01	6.73e−05
	Tuned IPSOLS	1.65e+00	**1.66e−11**	4.01e+01	**1.30e−11**	1.15e+01	**2.55e−07**	2.88e+02	**4.99e−08**	4.34e+00	**1.92e−09**	2.56e+01	**2.31e−11**
500	DE	**2.04e−08**	2.05e−08	**2.30e−08**	1.71e−08	1.11e+02	1.11e+02	1.13e+02	1.10e+02	**1.22e−03**	1.22e−03	**1.74e−03**	9.19e−04
	CHC	9.55e+02	9.36e+02	1.11e+03	2.32e+01	8.40e+05	1.62e+03	1.21e+07	2.96e+02	7.32e+02	7.29e+02	8.13e+02	6.32e+02
	G-CMA-ES	5.45e+03	5.43e+03	5.85e+03	5.23e+03	9.59e+03	9.50e+03	1.06e+04	8.60e+03	2.05e+03	2.06e+03	2.17e+03	1.92e+03
	IPSOLS	6.74e−02	2.51e−03	1.61e+00	4.56e−04	1.51e+01	1.14e+00	9.60e+01	8.00e−03	1.39e+00	**6.31e−04**	2.39e+01	2.31e−04
	Tuned IPSOLS	1.33e+00	**4.40e−11**	3.16e+01	**4.03e−11**	**1.33e+01**	**2.07e−06**	**8.71e+01**	**2.82e−07**	1.33e+01	8.95e+00	5.47e+01	**6.09e−11**
1000	DE	**4.18e−08**	4.19e−08	**4.71e−08**	3.61e−08	2.36e+02	2.35e+02	2.51e+02	2.34e+02	**2.37e−03**	2.37e−03	**3.23e−03**	2.03e−03
	CHC	2.32e+03	2.30e+03	2.75e+03	5.46e+01	2.04e+07	3.81e+03	3.15e+08	1.91e+03	1.72e+03	1.71e+03	2.02e+03	1.59e+03
	G-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
	IPSOLS	7.00e−02	4.92e−03	1.61e+00	1.91e−03	**4.42e+00**	1.85e−01	**5.25e+01**	1.25e−02	2.06e+00	**1.61e−03**	2.21e+01	5.28e−04
	Tuned IPSOLS	2.41e+00	**9.15e−11**	2.94e+01	**8.28e−11**	6.49e+00	**3.91e−06**	6.05e+01	**2.11e−07**	2.46e+01	1.61e+00	1.08e+02	**1.35e−10**

Table 13: Comparison of the average, median, maximum and minimum objective function values obtained with DE, CHC, G-CMA-ES, and the default and tuned configurations of IPSOLS. The lowest values for each combination of statistic, dimensionality and function are highlighted in **boldface**.

Dimension	Algorithm	Benchmark function F_{19}			
		Avg.	Median	Maximum	Minimum
50	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	3.59e+02	4.20e+00	5.26e+03	**0.00e+00**
	G-CMA-ES	4.76e+00	4.03e+00	9.28e+00	4.13e−01
	IPSOLS	2.36e−10	1.55e−10	6.86e−10	2.98e−11
	Tuned IPSOLS	1.19e−11	1.83e−12	1.15e−10	2.87e−14
100	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	5.45e+02	1.26e+01	5.67e+03	4.20e+00
	G-CMA-ES	2.02e+01	1.47e+01	1.55e+02	6.71e+00
	IPSOLS	3.51e−10	2.74e−10	9.29e−10	1.06e−11
	Tuned IPSOLS	7.33e−12	3.43e−12	5.49e−11	2.62e−14
200	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	2.06e+03	1.65e+02	2.11e+04	1.47e+01
	G-CMA-ES	7.52e+02	5.74e+02	3.08e+03	3.36e+01
	IPSOLS	5.55e−10	2.95e−10	2.62e−09	3.01e−12
	Tuned IPSOLS	1.11e−11	3.52e−12	7.41e−11	4.57e−14
500	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	1.76e+03	5.66e+02	1.17e+04	4.82e+01
	G-CMA-ES	2.44e+06	2.50e+06	6.00e+06	3.48e+05
	IPSOLS	1.58e−09	1.25e−09	5.38e−09	3.67e−11
	Tuned IPSOLS	2.73e−11	9.55e−12	1.86e−10	2.19e−13
1000	DE	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	CHC	4.20e+03	3.45e+03	1.73e+04	1.19e+02
	G-CMA-ES	-	-	-	-
	IPSOLS	2.57e−09	1.42e−09	9.28e−09	1.68e−11
	Tuned IPSOLS	1.01e−10	1.61e−11	7.50e−10	1.34e−13

DE finds the optimum of functions F_1, F_5, F_7, F_{10}, F_{15}, and F_{19} in all the five dimensionalities tested. However, F_{15} and F_{19} are hybrid functions whose basic building blocks come from functions F_7 and F_{10}, which may explain why DE has such a good performance on these functions. Thus, the core of functions that make DE rank so well consists of four functions. G-CMA-ES consistently obtains very good results with functions F_1 and F_8 in all but the 1000-dimensional case. It is interesting to note that F_1 and F_8 are quadratic functions. F_1 is separable but F_8 is a hyperellipsoid that is rotated in all directions, and thus is not separable. These results are explained by recalling that CMA-ES is invariant with respect to the rotation of the coordinate system [13] and that, according to its author, it is a second order method that approximates a positive definite matrix closely related to the Hessian matrix of the objective function [12]. This property allows CMA-ES to converge rapidly to high-quality solutions on quadratic objective functions. G-CMA-ES obtains also good results with function F_5, but its good performance is inconsistent. IPSOLS, in either its default or tuned configuration, finds the optimum of functions F_1, F_3, F_4, F_5, F_6, and F_{10}. These functions are very different from each other. Some of these functions are easily solved if optimization is carried out dimension-per-dimension, just as Powell's conjugate direction set method does. However, it is not the case for all the aforementioned functions. IPSOLS effectively combines the strengths of a powerful local search method with those of an effective heuristic optimization algorithm such as PSO.

4.3 Execution Time Scalability

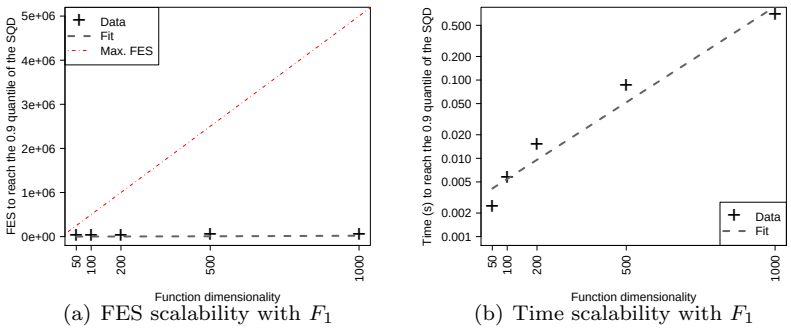
The results presented so far have focused on the solution quality obtained after some computational budget, in our case the maximum number of function evaluations, has expired. While those results are important, it is still unclear

whether the proposed algorithms are really suitable for tackling large-scale continuous optimization problems. The reason is that, in some cases, the execution time can grow so fast with the problem size that an otherwise good algorithm may become impractical. That is the case of G-CMA-ES. Its execution time was so high that its results on 1000-dimensional problems were not even reported.

In this section, we study the execution time scalability of IPSOLS. During each run of the algorithm, every time the best-so-far solution improved, we recorded the number of function evaluations and the CPU time used up to that moment. With this information we can estimate the solution quality distribution (SQD) of the algorithm at, for example, the maximum number of function evalutions. In the previous section, we analyzed some statistics (i.e., mean, median, maximum and minimum) of the SQDs of IPSOLS. To conduct the scalability study, we use the 0.9-quantile of the SQDs of IPSOLS on each benchmark function. We chose the 0.9-quantile as a conservative measure of the achievable solution quality. We focus on two measures: the median number of function evaluations (FES) and the median time (in seconds) needed by IPSOLS (in its default configuration) to find a solution that is at least as good as the 0.9-quantile of the SQD after up to $5000n$ function evaluations, where n is the dimensionality of the function tackled. Since the scalability behavior of of any algorithm is problem-dependent, we show only three examples in Figure 9. The rest of the results can be found in [22].

Let us first focus on the behavior of the algorithm with respect to the number of function evaluations consumed. For function F_1, the linear model $fes = 18.08n - 424.96$ fits the data with an adjusted r^2 score of 0.9957. The variable *fes* is the number of function evaluations needed by IPSOLS to find a solution at least as good as the 0.9-quantile of the SQD, and n is the dimensionality of the function. It is clear that the time execution salability of IPSOLS with function F_1 is quite good as a consequence of the fact that Powell's conjugate directions method exploits the separability of the function. In the case of function F_8, the linear model $fes = 4699.85n + 16261.51$ fits the data with an adjusted r^2 score of 0.9999. The slope of this model is almost the same as the slope of the computational budget limit. This means that IPSOLS would most likely continue making progress toward better solutions if more function evaluations were allowed. Again, this is the product of the Powell's conjugate directions method, which in the case of F_8 would align the search directions, one by one, with the axes of the hyperellipsoid that F_8 generates. Finally, we show the execution time of IPSOLS with function F_{17}, which scales quadratically with the size of the problem. In this case the model is $fes = 5.14n^2 - 1643.03n + 182300$ with an adjusted r^2 score of 0.998. The extra complexity that this function represents for IPSOLS is evident.

In terms of CPU time, the scalability of IPSOLS seems to follow the form $time = Ae^{Bn}$. For F_1, the parameters of the fitted model are $A = 0.0031$ and $B = 0.0056$ with an adjusted r^2 score equal to 0.949. For F_8, the parameters are $A = 3.1598$ and $B = 0.0042$ with an adjusted r^2 score equal to 0.902. Finally, for F_{17}, the parameters are $A = 2.0851$ and $B = 0.0075$ with an adjusted r^2 score equal to 0.936. Even though the fitted model is exponential, the scalability of IPSOLS in seconds is very good because the coefficient B is rather small (in the order of 1e−03) for all the 19 functions studied. We ran our experiments on Intel Xeon E5410 quad core 2.33GHz computers with 2×6MB L2 cache and 8GB of RAM. All computers used Linux (kernel 2.6.9-78.0.22) as operating system



(a) FES scalability with F_1 (b) Time scalability with F_1

(c) FES scalability with F_8 (d) Time scalability with F_8

(e) FES scalability with F_{17} (f) Time scalability with F_{17}

Figure 9: Median number of function evaluations (FES) and median number of seconds to find a solution at least as good as the 0.9-quantile of solution quality distribution (SQD) after up to $5000n$ FES (i.e., the maximum computational budget represented by the thin dotted line) on functions F_1, F_8, and F_{17}. The result of a regression analysis over the observed data is shown with a dotted line.

and IPSOLS was compiled with GCC 4.1.0.

5 Related Work and Conclusions

In this article, we have made extensive usage of automatic algorithm configuration methods to redesign and specialize a generic IPSOLS algorithm, a PSO algorithm that uses an incremental population size and a local search algorithm to improve some particles' solutions. Our article is not the first to use auto-

matic parameter configuration methods in the context of heuristic algorithms for continuous optimization. Earlier approaches designed to tune numerical parameters of algorithms for continuous optimization include SPO [4], SPO$^+$ [16], and REVAC [26]. A comparison of such methods has been presented in [31]. In all these approaches, however, the algorithm was tuned on the very same benchmark function on which it was later tested; thus all these examples are prone to over-tuning. Only rather recently, an experimental comparison of tuning algorithms for calibrating numerical algorithm parameters in numerical optimization that does not incur the problems of over-tuning has been presented [35]. In this present study, we also avoid over-tuning by applying Iterated F-race to problems of much smaller dimension than the ones on which the algorithms are tested later. An additional novelty, when compared to earlier researches on automatically tuning parameters of algorithms for continuous optimization, is that here we make extensive usage of automatic tuning in the design process of a high-performing algorithm; other work so far has focused on the fine-tuning of an already developed, final algorithmic scheme.

Efforts to benchmark continuous optimization algorithms on large-scale functions are rather recent. They include special sessions and benchmark competitions at the IEEE CEC 2008 and 2010 conferences and the ISDA 2009 workshop, where, in part, similar functions have been tested. Clearly, it would be interesting to apply our algorithm to these earlier benchmarks under the conditions imposed in those algorithm comparisons.

The IPSOLS algorithm we have designed and fine-tuned in this article has shown to reach very high performance on the high-dimensional benchmark functions that have been proposed by F. Herrera *et al.* [14]. In fact, the final IPSOLS algorithm was found to clearly outperform CHC [11] and G-CMA-ES [1] in its default and its tuned version. Its performance was competitive with the differential evolution (DE) algorithm, which was given as a baseline comparison for these functions and which turned out to be of extremely high performance. When using the algorithms' median results as measured across 25 trials on each of the 19 benchmark functions, our tuned IPSOLS algorithm was statistically significantly better than the DE algorithm; when basing the comparison on means across the 25 trials on each benchmark function, no statistically significant differences could be detected. This latter result could be attributed to the fact that in (very) few runs, our IPSOLS algorithm returns relatively poor quality solutions. However, we are convinced that by a more refined restart criterion, these convergence problems may be resolved in future work.

Apart from the high-performing IPSOLS algorithm, a major contribution in this article is rather of methodological nature. In fact, we have shown that the tuning–in–the–loop approach to redesign an effective IPSOLS algorithm for high-dimensional functions can be very effective. This redesign process has taken six stages and insights obtained in earlier stages have been exploited to direct the further development process in later stages. Interestingly, even the default version of the IPSOLS algorithm obtained at stage six of the redesign process outperformed the tuned version of IPSOLS from the first redesign stage. This clearly illustrates that integrating automatic algorithm configuration techniques into a systematic engineering process of stochastic local search (SLS) algorithms [33,34] is a very powerful means of obtaining new high-performance algorithms.

There are many avenues for further research. A first one is to extend automatic algorithm configuration techniques to better deal with the scaling behav-

ior of algorithm parameters to very large-scale problems. In fact, our algorithm tuning was done on 10-dimensional versions of the high-dimensional benchmark functions and, maybe luckily, the found parameter settings turned out to result in very high performance even on benchmark functions that had 100 times larger dimension. Further research is needed to explore better ways of how to find well scaling algorithm parameters. A second one is to further elaborate on automatic algorithm configuration for continuous optimization. Some approaches exist but often these suffer from over-tuning, since tuning is done on single benchmark functions. A third, promising direction is to apply our tuning–in–the–loop algorithm engineering also to other algorithms for continuous function optimization. In fact, the DE algorithm proposed as a baseline for the competition would be an interesting candidate for such an undertaking. Finally, we intend to apply extensively automatic algorithm configuration techniques, integrated into a systematic SLS algorithm engineering process, to develop new state-of-the-art SLS algorithms for continuous optimization. Our initial results on applying such a process in the redesign of an IPSOLS algorithm for large-scale continuous function optimization encourage us to continue research in this direction.

Acknowledgements

The authors thank Manuel López-Ibáñez for adapting the code of iterated F-Race to deal with the tuning task studied in this paper.

A Appendix. Supplementary Material

A.1 Benchmark Functions

A set of 19 scalable benchmark functions was used in this paper. Their mathematical definition is shown in Table A.1. These functions were proposed by F. Herrera *et al.* [14]. The source code that implements them was also provided by them (available from [21]).

For functions F_1 to F_{11} (and some of the hybrid functions, F_{12} to F_{19}), candidate solutions, $\boldsymbol{x}$, are transformed as $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$ before evaluation. This transformation shifts the optimal solution from the origin of the coordinate system to $\boldsymbol{o}$, with $\boldsymbol{o} \in [X_{\min}, X_{\max}]^n$. For function F_3, the transformation is $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} + \mathbf{1}$. Hybrid functions combine two basic functions. The combination procedure is shown in [14]. The parameter m_{ns} is used to control the number of components that are taken from a nonseparable function (functions F_3, F_5, F_9, and F_{10}). The higher m_{ns}, the larger the number of components evaluated that come from a nonseparable function.

A.2 Ranges and Setting of Free and Fixed Algorithm Parameters

During tuning, the number of free parameters and their corresponding range or domain has to be given to iterated F-Race. The list of free parameters and their corresponding range or domain as used with iterated F-Race is given in Table A.2. A description of their meaning and effect is given in the main text.

Table A.1: Benchmark functions

Name	Definition	Search Range $[X_{\min}, X_{\max}]^n$
F_1	$\sum_{i=1}^{n} z_i^2$	[-100,100]^n
F_2	$\max_i\{\lvert z_i\rvert,\ 1 \le i \le n\}$	[-100,100]^n
F_3	$\sum_{i=1}^{n-1} [100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2]$	[-100,100]^n
F_4	$10n + \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i))$	[-5,5]^n
F_5	$\frac{1}{4000}\sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$	[-600,600]^n
F_6	$-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)} + 20 + e$	[-32,32]^n
F_7	$\sum_{i=1}^{n} \lvert z_i\rvert + \prod_{i=1}^{n} \lvert z_i\rvert$	[-10,10]^n
F_8	$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j\right)^2$	[-65.536,65.536]^n
F_9	$\sum_{i=1}^{n-1} f_{10}(z_i, z_{i+1}) + f_{10}(z_n, z_1)$, where $f_{10}(x,y) = (x^2+y^2)^{0.25}(\sin^2(50(x^2+y^2)^{0.1}) + 1)$	[-100,100]^n
F_{10}	$\sum_{i=1}^{n-1} (z_i^2 + 2z_{i+1}^2 - 0.3\cos(3\pi z_i) - 0.4\cos(4\pi z_{i+1}) + 0.7)$	[-15,15]^n
F_{11}	$\sum_{i=1}^{n-1} (z_i^2 + z_{x+1}^2)^{0.25}(\sin^2(50(z_1^2 + z_{i+1}^2)^{0.1}) + 1)$	[-100,100]^n
F_{12}	$F_9 \oplus F_1$, $m_{ns} = 0.25$	[-100,100]^n
F_{13}	$F_9 \oplus F_3$, $m_{ns} = 0.25$	[-100,100]^n
F_{14}	$F_9 \oplus F_4$, $m_{ns} = 0.25$	[-5,5]^n
F_{15}	$F_{10} \oplus F_7$, $m_{ns} = 0.25$	[-10,10]^n
F_{16}	$F_9 \oplus F_1$, $m_{ns} = 0.5$	[-100,100]^n
F_{17}	$F_9 \oplus F_3$, $m_{ns} = 0.75$	[-100,100]^n
F_{18}	$F_9 \oplus F_4$, $m_{ns} = 0.75$	[-5,5]^n
F_{19}	$F_{10} \oplus F_7$, $m_{ns} = 0.75$	[-10,10]^n

Other parameter settings for IPSOLS, for both tuned and non-tuned versions, remained fixed. A list of them with their settings is shown in Table A.3.

A.3 Iterated F-Race Parameter Setting

Iterated F-Race [3, 7] has a number of parameters that need to be set before it can be used. The parameters setting used in our work is shown in Table A.4.

In iterated F-Race, the number of iterations L is equal to $2+\text{round}(\log_2(d))$, where d is the number of parameters to tune. In iterated F-Race, each iteration has a different maximum number of evaluations. This number, denoted by B_l, is equal to $(B - B_{\text{used}})/(L - l + 1)$, where l is the iteration counter, B is the overall maximum number of evaluations, and B_{used} is the number of evaluations used until iteration $l-1$. The number of candidate configurations tested during iteration l is equal to $\lfloor B_l/\mu_l \rfloor$. For more information on the parameters of iterated F-Race and their effect, please see [3, 7].

Table A.2: Free parameters in IPSOLS (all versions)

Parameter	Range/Domain
w	$[0, 1]$
φ_1	$[0, 4]$
φ_2	$[0, 4]$
Init. pop. size	$[1, 100]$
k	$[1, 10]$
Topology[1]	$\{FC, R\}$
FTol	$[-13, -1]$
ρ_{start}[2]	$[1, 50]$
s[3]	$[\epsilon, 100]$
MaxFES[2]	$[22, 210]$
MaxITER[3]	$[1, 10]$
MaxFailures	$[1, 20]$
MaxStagIter	$[2, 30]$

[1] FC stands for fully connected, R for ring.
[2] Used with BOBYQA.
[3] Used with Powell's conjugate directions method.

Table A.3: Fixed parameters in IPSOLS

Parameter	Value
Max. pop. size	1000
$V_{\max,j}$, $1 \leq j \leq n$	$X_{\max,j}$
Bound constraint handling (PSO part)	Fix on the bound, set velocity to zero
ϵ	*0-threshold*

Table A.4: Iterated F-Race parameter settings

Parameter	Value
Max. Evaluations (B)	5e+04
μ_l	$76+l$

References

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 1769–1776. IEEE Press, Piscataway, NJ, 2005.

[2] A. Auger, N. Hansen, J. M. P. Zerpa, R. Ros, and M. Schoenauer. Experimental comparisons of derivative free optimization algorithms. In J. Vahrenhold, editor, *LNCS 5526. Proceedings of the Symposium on Experimental Algorithmics (SEA 2009)*, pages 3–15. Springer, Heidelberg, Germany, 2009.

[3] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein et al., editors, *LNCS 4771. Proceedings of the International Workshop on Hybrid Metaheuristics (HM 2007)*, pages 108–122. Springer, Heidelberg, Germany, 2007.

[4] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation–The New Experimentalism*. Springer, Berlin, Germany, 2006.

[5] M. Birattari. *Tuning Metaheuristics: A machine learning perspective*. Springer, Berlin, Germany, 2009.

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

[7] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-Race and iterated F-Race: An overview. In T. Bartz-Beielstein et al., editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer, Berlin, Germany, 2010.

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

[9] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[10] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. MPS-SIAM, Philadelphia, PA, 2000.

[11] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In D. L. Whitley, editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.

[12] N. Hansen. The CMA evolution strategy. Online, July 2010. `http://www.lri.fr/~hansen/cmaesintro.html`.

[13] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, San Francisco, CA, 1995.

[14] F. Herrera, M. Lozano, and D. Molina. Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Last accessed: July 2010. http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf.

[15] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, CA, 2005.

[16] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. An experimental investigation of model-based parameter optimisation: SPO and beyond. In *Proc. of GECCO 2009*, pages 271–278. ACM Press, New York, NY, 2009.

[17] S. G. Johnson. The NLopt nonlinear-optimization package, Last accessed: April 2010. http://ab-initio.mit.edu/nlopt.

[18] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.

[19] A. R. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown. SATenstein: Automatically building local search SAT solvers from components. In C. Boutilier et al., editors, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 517–524. 2009.

[20] M. López-Ibáñez and T. Stützle. Automatic configuration of multi-objective ACO algorithms. In M. Dorigo et al., editors, *LNCS 6234. Proceedings of the International Conference on Swarm Intelligence (ANTS 2010).*, LNCS, pages 96–107. Springer, Heidelberg, Germany, 2010.

[21] M. Lozano and F. Herrera. Call for papers: Special issue of soft computing: A fusion of foundations, methodologies and applications on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, Last accessed: July 2010. http://sci2s.ugr.es/eamhco/CFP.php.

[22] M. A. Montes de Oca, D. Aydin, and T. Stützle. An incremental particle swarm for large-scale optimization problems: Complete data. http://iridia.ulb.ac.be/supp/IridiaSupp2010-011.

[23] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo. Incremental social learning in particle swarms. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 2010. Forthcoming.

[24] M. A. Montes de Oca, K. Van den Enden, and T. Stützle. Incremental particle swarm-guided local search for continuous optimization. In M. J. Blesa et al., editors, *LNCS 5296. Proceedings of the International Workshop on Hybrid Metaheuristics (HM 2008)*, pages 72–86. Springer, Heidelberg, Germany, 2008.

[25] J. More and S. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[26] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 975–980. 2007.

[27] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[28] M. J. D. Powell. *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, chapter The NEWUOA software for unconstrained optimization, pages 255–297. Springer-Verlag, Berlin, Germany, 2006.

[29] M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical Report NA2009/06, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2009.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, New York , NY, second edition, 1992.

[31] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 399–406. IEEE Press, Piscataway, NJ, 2009.

[32] R. Storn and K. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[33] T. Stützle, M. Birattari, and H. H. Hoos, editors. *Engineering Stochastic Local Search Algorithms. Designing Implementing and Analizing Effective Heuristics. International Workshop, SLS 2007.* LNCS 4638. Springer, Heidelberg, Germany, 2007.

[34] T. Stützle, M. Birattari, and H. H. Hoos, editors. *Engineering Stochastic Local Search Algorithms. Designing Implementing and Analizing Effective Heuristics. Second International Workshop, SLS 2009.* LNCS 5752. Springer, Heidelberg, Germany, 2009.

[35] Z. Yuan, M. A. Montes de Oca, M. Birattari, and T. Stützle. Modern continuous optimization algorithms for tuning real and integer algorithm parameters. In M. Dorigo et al., editors, *LNCS 6234. Proceedings of the International Conference on Swarm Intelligence (ANTS 2010).*, pages 204–215. Springer, Heidelberg, Germany, 2010.