Grid-Based Parameter Adaptation in Particle Swarm Optimization

Vasileios A. Tatsis, Konstantinos E. Parsopoulos

Department of Computer Science and Engineering,
University of Ioannina,
GR-45110 Ioannina, Greece
{vtatsis, kostasp}@cse.uoi.gr

Abstract

Metaheuristics have been established as essential tools for solving complex optimization problems. Although they have proved to be useful in cases where other algorithms fail, their performance is heavily dependent on their proper parametrization. This is typically addressed through a laborious preprocessing phase of parameter tuning, which consumes significant amount of computational resources and development time. For this purpose various techniques, also called tuners, have been proposed in the relevant literature. Tuners are distinguished into offline and online ones, depending on whether they tune the parameters prior or during the algorithm's execution. Recently a grid-based parameter tuner was proposed and successfully applied on the Differential Evolution algorithm. The present work extends the tuner's application also to Particle Swarm Optimization for the adaptation of its scalar parameters and neighborhood radius. Proper modifications are introduced and a first empirical evaluation is conducted on a large-scale optimization test suite where standard PSO exhibited questionable performance. The results show that PSO equipped with the proposed tuner becomes more efficient while, concurrently, the user is relieved from the burden of proper parametrization.

1 Introduction

Metaheurstics have been widely recognized as valuable approaches for solving a variety of complex optimization problems in science and engineering. Although optimality of the final solution is not guaranteed, metaheuristics can provide useful (sub-)optimal solutions within reasonable computation time. Also, they typically have minor requirements regarding the form and mathematical properties of the underlying problem models. This property renders them viable as alternatives for tackling otherwise intractable real-world problems.

Numerous experimental evidence suggests that performance of metaheuristics is highly dependent on their parameterization [5]. Also, it is widely perceived that proper parameter values are problem-dependent and can be hardly generalized to different problem types. Thus, the user is typically obligated to conduct a laborious parameter-tuning procedure prior to the application of the algorithm. A number of diverse techniques, also called *tuners*, have been proposed for this purpose. Tuners define formal procedures for determining proper parameter values for the algorithm. These techniques are often based on Computational Statistics and they require a significant computational cost that, in some cases, may exceed even the cost of solving the problem itself. Such techniques are also called *offline tuners* [5, 6]. Design of Experiments [2], F-Race [3], and ParamILS [8] are among the most popular ones.

Alternatively, *online tuners* are techniques for the dynamic adaptation of the algorithm's parameters during its execution. They are typically based on performance feedback from the algorithm on the considered problem instance. The absence of preprocessing phase and the minimal human interaction renders online tuners quite appealing. Yet, their outcome is barely reusable in different problems even of the same type, since it is explicitly fitted to the specific run of the algorithm on the specific problem instance [5, 6].

There is a number of online parameter-tuning techniques but most of them are algorithm-dependent. With respect to our algorithm of interest, namely Particle Swarm Optimization, one can mention the technique proposed in [16], where the inertia weight is automatically controlled based on the swarm's distribution and particles' fitness values. In [4] an adaptable PSO algorithm was proposed based on a stability criterion, while in [13] a fuzzy system was employed for the same purpose.

Recently, a generic online tuner based on parameter search in a discretized parameter space (grid) was proposed in [14]. The tuner was demonstrated on the Differential Evolution (DE) algorithm, which is widely known for its parameter sensitivity. The derived DEGPA approach adapts the scalar parameters of DE according to estimations of the algorithm's performance. The estimations are based on short runs initiated from the current population instance but with neighboring parameter values in the grid. DEGPA was extended in the DEGPOA variant [15] by adapting also the mutation operator of the DE algorithm. Empirical assessment was conducted on a large-scale optimization test suite published in the Soft Computing journal [10] and significant improvements for DE were achieved.

The present work aims at extending the previous studies by incorporating the grid-based parameter adaptation approach in the Particle Swarm Optimization (PSO) algorithm. PSO is more thoroughly studied and theoretically analyzed than DE, with default parameter sets been proposed in the relevant literature. Nevertheless, it is unclear whether these values are appropriate for a wide range of problems. Thus, there are several cases where the user needs to assign arbitrary parameter values. Especially in the considered test suite [10], PSO equipped with its default parametrization exhibits questionable performance. Thus, the use of the grid-based parameter adaptation technique in PSO appears to be an attractive option.

The rest of the paper is organized as follows: Section 2 briefly describes the PSO algorithm, while Section 3 introduces the proposed approach. Section 4 offers experimental evaluation and analysis and, finally, Section 5 concludes the paper.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) occupies a salient position among the state-of-the-art in metaheuristics literature. It was initially proposed by J. Kennedy and R. Eberhart in 1995 [9]. Its robustness and efficiency in a variety of complex optimization problems, as well as its easy implementation and minor requirements on the underlying problem's model, has placed PSO among the most popular algorithms.

The main concept of PSO involves a *swarm* of search points called *particles*. The particles cooperatively probe the search space through adaptable position shifts, while retaining in *memory* their best visited positions. Putting it formally, let,

$$f : X \to \mathbb{R}, \qquad X \subset \mathbb{R}^n,$$

be the objective function under consideration and,

$$S = \{x_1, x_2, \ldots, x_N\},$$

be a swarm of search points, which is randomly and uniformly initialized in X. Each particle,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in X, \qquad i \in I \stackrel{\text{def}}{=} \{1, 2, \ldots, N\},$$

represents a candidate solution of the problem. We will denote the objective value of x_i simply as $f_i = f(x_i)$. Each particle x_i iteratively moves in X according to an adaptable position shift called *velocity*, which is denoted as,

$$v_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\}.$$

Its best visited position, denoted as,

$$p_i = \{p_{i1}, p_{i2}, \ldots, p_{in}\} \in X,$$

is stored in an external memory,

$$P = \{p_1, p_2, \ldots, p_N\}.$$

The operation of PSO is strongly dependent on information exchange among the particles. Specifically, the particles communicate their best positions to other particles through communication channels that define their *neighborhoods*. This socially shared information is then used to guide their move.

There are two major PSO models with respect to the extent of information sharing. In the global (*gbest*) model each particle is aware of the overall best position of the whole swarm. In the local (*lbest*) model, only particles that belong to strict subsets of the swarm share their best positions. The communication channels among the particles are determined by the employed *neighborhood topology*. The most popular neighborhood topology is the *ring*, where the i-th particle takes into account the findings of particles with indices belonging in the set,

$$NB_i = \{i-m, \ldots, i-1, i, i+1, \ldots, i+m\} \subset I.$$

This neighborhood topology can be depicted as a ring-shaped graph (hence the name) where particles lie on the nodes and each one is connected only with its two immediate neighbors. The parameter m defines the size of the neighborhood and it is called the *neighborhood's radius*.

Let the index g_i denote the best particle in NB_i, i.e.,

$$g_i = \arg\min_{k \in NB_i} f(p_k),$$

and let t denote the iteration counter. Then, the update equations of plain PSO are given as follows [12]:

$$v_{ij}^{(t+1)} = w\, v_{ij}^{(t)} + c_1\, \texttt{rand}()\, \left(p_{ij}^{(t)} - x_{ij}^{(t)}\right) + c_2\, \texttt{rand}()\, \left(p_{g_i j}^{(t)} - x_{ij}^{(t)}\right), \tag{1}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{2}$$

where $i \in I$ and $j = 1, 2, \ldots, n$; w is a velocity clamping parameter called *inertia weight*; c_1, c_2, are called the *cognitive parameter* and *social parameter*, respectively; and $\texttt{rand}()$ is the realization of a uniform random number generator that returns a random real number in the range $[0, 1]$ at each call.

Finally, each particle updates its best position at every iteration as follows,

$$p_i^{(t+1)} = \begin{cases} x_i^{(t+1)}, & \text{if } f_i^{(t+1)} \leqslant f\left(p_i^{(t)}\right), \\ p_i^{(t)}, & \text{otherwise.} \end{cases} \tag{3}$$

The algorithm iterates until a user-defined termination condition is satisfied. The reader can find thorough presentation of PSO and its variants in [11, 12].

3 Proposed Approach

The proposed approach is an extension of the previously introduced DEGPA approach [14, 15]. In the present case, the grid-based adaptation scheme is used to dynamically adapt the scalar parameters w, c_1, c_2, and the neighborhood radius m of the lbest PSO during its run. We will henceforth call the proposed approach *Particle Swarm Optimization with Parameter and Neighborhood Adaptation* (PSOPNA).

Initially, the search space of the scalar parameters of PSO shall be defined. In the present work we assumed the following ranges,

$$w \in (0, 1], \qquad c_1, c_2 \in (0, 3],$$

which are commonly used in various applications. These ranges are discretized with predefined step sizes,

$$\lambda_w = \lambda_{c_1} = \lambda_{c_2} = 0.1. \tag{4}$$

Also, a set of possible values for the neighborhood radius m is defined. In our case we considered,

$$m \in \{1, 3, 5, 7, 9\}.$$

For each value of m, a 3-dimensional grid parameter search space is defined as follows,

$$\mathcal{G}_m = \{(w, c_1, c_2);\ w \in \{0.0, 0.1, \ldots, 1.0\}, c_1, c_2 \in \{0.0, 0.1, 0.2, \ldots, 3.0\}\}.$$

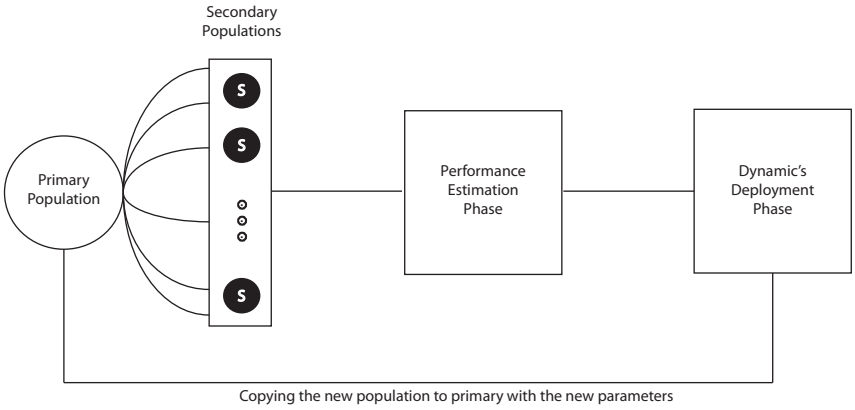

Figure 1: A complete cycle of the proposed approach.

The grid's density can be changed by modifying the step sizes of Eq. (4) if more fine-grained parameter search is needed. Obviously, each triplet (w, c_1, c_2) in the interior of $\mathcal{G}_m$ has 6 immediate neighboring points along the 3 directional axes.

The algorithm starts by randomly initializing a swarm, called the *primary swarm* and denoted as S_p, assuming an initial neighborhood radius m and a set of parameters $(w, c_1, c_2) \in \mathcal{G}_m$. A reasonable initial choice is,

$$m = 1, \quad (w, c_1, c_2) = (0.5, 1.5, 1.5),$$

but the algorithm works also for different choices as it will be shown later. The primary swarm is evolved for a number of iterations defined as,

$$t_p = 10 \times n,$$

following the setting in [14]. Then, three main phases take place, namely *cloning*, *performance estimation*, and *dynamic deployment*, which are explained below.

Cloning

The primary swarm, along with its best positions, is copied in 7 *secondary swarms*, $S_1, \ldots, S_7$, of the same neighborhood radius m, each one assuming different scalar parameters as follows,

$$w' = w + s_w\,\lambda_w, \quad c_1' = c_1 + s_1\,\lambda_{c_1}, \quad c_2' = c_2 + s_2\,\lambda_{c_2}, \qquad s_w, s_1, s_2 \in \{-1, 0, 1\}. \tag{5}$$

Obviously, the case $s_w = s_1 = s_2 = 0$ corresponds to the parameter setting of the primary swarm. Besides the 7 secondary swarms of same radius m, four additional secondary swarms, $S_8, \ldots, S_{11}$, are considered, adopting the scalar parameters (w, c_1, c_2) of the primary swarm but for different neighborhood radius values, i.e., for $m = 3, 5, 7$, and 9. Intuitively, these secondary swarms define bridges from the 3-dimensional grid of $m = 1$ to the other grids, aiming at identifying a more beneficial neighborhood radius for the algorithm.

Performance Estimation

All the 11 secondary swarms are then evolved for a small number of iterations, $t_s \ll t_p$, according to the standard PSO procedure, updating also their best positions. These short runs aim at revealing rough performance trends of the secondary swarms with their assigned parameter settings. For time-efficiency purpose, the short runs can be done in parallel by evoking a separate thread for each individual secondary swarm, thereby taking full advantage of modern multi-core computer systems. Following previous work, typical values for t_s lie between 5 and 10 iterations.

Algorithm 1 Pseudocode of the PSOPNA approach.

initialize$(S_p, P_p, m, w, c_1, c_2)$ `/* primary swarm */`
evolve(S_p, P_p, t_p)
$M \leftarrow 11$ `/* number of secondary swarms */`
while (not termination) **do**
  `/* Cloning */`
  for $(i = 1 \dots M)$ **do**
    if $(i \leqslant 7)$ **then**
      `/* secondary swarms: same radius, different parameters */`
      $(S_i, P_i, m', w', c_1', c_2') \leftarrow (S_p, P_p, m, w', c_1', c_2')$
    else
      `/* secondary swarms: different radius, same parameters */`
      $(S_i, P_i, m', w', c_1', c_2') \leftarrow (S_p, P_p, m', w, c_1, c_2)$
    end if
  end for
  `/* Performance Estimation */`
  for $(i = 1 \dots M)$ **do**
    evolve(S_i, P_i, t_s) `/* short runs (in parallel) */`
  end for
  `/* Dynamic's Deployment */`
  $(S_p, P_p, m, w, c_1, c_2) \leftarrow$ **select**$(S_i, P_i, m', w', c_1', c_2'; \text{AOV}, \text{OVSD}, \text{IQR})$
  evolve$(S_p, P_p, m, w, c_1, c_2, t_p)$
end while

After the short runs, for each secondary swarm S_j the *average objective value* (AOV) of its best positions P_j is computed [14],

$$\bar{f}_j = \frac{1}{N} \sum_{i=1}^{N} f\left(p_i\right), \qquad p_i \in P_j, \quad i \in I, \quad j = 1, \dots, 11. \tag{6}$$

along with the *objective value standard deviation* (OVSD) of its best positions [15],

$$\sigma_j = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(f(p_i) - \bar{f}_j\right)}, \qquad p_i \in P_j, \quad i \in I, \quad j = 1, \dots, 11. \tag{7}$$

The 11 obtained performance pairs,

$$(\bar{f}_j, \sigma_j), \qquad j = 1, \dots, 11,$$

are then compared in terms of Pareto dominance and the non-dominated ones are distinguished. In order to select one among the non-dominated secondary swarms but also reduce the possibility of being misled due to temporarily extremal (high or low) objective values, we consider a diversity-oriented performance measure computed for each non-dominated swarm. Specifically, we select the secondary swarm that has the highest *interquartile range* (IQR).

IQR is a common statistical measure of variability, based on the division of a data set into four equal quartiles. The data is sorted and the IQR is defined as,

$$IQR = Q3 - Q1,$$

where $Q1$ and $Q3$ specify the 1st and 3rd quartile of the data, respectively. In our case, IQR is computed on the best positions values $f\left(p_i\right)$ of the non-dominated secondary swarms. The secondary swarm with the highest IQR value is then selected in order to retain search diversity.

Table 1: Statistical comparisons of PSOPNA against plain PSO.

		$\text{PSOPNA}_{0.5/1.5}$			$\text{PSOPNA}_{0.2/1.0}$			$\text{PSOPNA}_{0.8/2.0}$		
Dimension	Algorithm	$+$	$-$	$=$	$+$	$-$	$=$	$+$	$-$	$=$
50	PSO	7	6	6	15	0	4	12	3	4
100	PSO	8	6	5	14	1	4	16	3	0
200	PSO	8	3	8	15	1	3	17	2	0
500	PSO	8	7	4	8	3	8	16	2	1

"$+$" denotes wins, "$-$" denotes losses, and "=" denotes ties of PSOPNA against PSO

Table 2: Statistical comparisons of $\text{PSOPNA}_{0.5/1.5}$ against the base algorithms.

	Dimension											
	50			100			200			500		
	$+$	$-$	$=$	$+$	$-$	$=$	$+$	$-$	$=$	$+$	$-$	$=$
DE_{bin}	6	12	1	6	10	3	7	8	4	9	5	5
DE_{exp}	3	13	3	2	17	0	1	15	3	6	12	1
CHC	10	4	5	11	4	4	9	5	5	17	1	1
GCMAES	14	3	2	13	4	2	13	5	1	n/a	n/a	n/a

"$+$" denotes wins, "$-$" denotes losses, and "=" denotes ties of PSOPNA against the corresponding base algorithm

Dynamic's Deployment

This is the final phase of the proposed approach, where the selected secondary swarm along with all its parameters becomes the primary swarm, replacing the existing one. In order to make complete use of newly detected best positions in the short runs, the overall bests of all the unselected secondary swarms are also inserted into the new primary swarm, replacing equal number of worst individuals. The new primary swarm is then evolved for t_p iterations to fully exploit the new parametrization.

This step completes a full cycle of the algorithm, and the whole procedure is repeated anew from the cloning phase. Given a maximum computational budget of $V_{\max}$ function evaluations, we can estimate the maximum number of cycles that will be performed by the algorithm as follows,

$$c_{\max} = \left\lfloor \frac{V_{\max}}{(t_p + 11\, t_s)\, N} \right\rfloor, \tag{8}$$

where $\lfloor . \rfloor$ defines the standard floor function. The main procedure is graphically illustrated in Fig. 1, while the approach is given in the pseudocode of Algorithm 1.

4 Experimental Evaluation

The experimental evaluation of the proposed PSOPNA approach was conducted on the large-scale optimization test suite published in the *Special Issue on Large Scale Continuous Optimization Problems* of the Soft Computing journal [10]. The test suite consists of 19 scalable test functions, henceforth denoted as $F1 - F19$, of dimension $n \in \{50, 100, 200, 500\}$. Note that the test suite includes the test functions of the IEEE CEC 2008 Competition on Large Scale Global Optimization.

The test suite provides three base algorithms for comparisons, namely DE with exponential crossover, CHC, and GCMAES [1, 7, 10]. All base algorithms are considered with their optimal parameters provided in [10]. Besides those algorithms, we also considered DE with binomial crossover, which is more popular than the one with exponential crossover. The maximum computational budget in the test suite is

Table 3: Average and standard deviation of error values for dimension $n = 50$ and 100.

Prob.	$\text{PSOPNA}_{0.5/1.5}$		DE_{bin}		DE_{exp}		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
50-dimensional										
$F1$	$2.32e-13$	$2.80e-13$	$3.00e-17$	$7.69e-18$	$2.78e-17$	$6.29e-33$	$2.90e+02$	$5.69e+02$	$2.78e-17$	$6.29e-33$
$F2$	$1.59e+01$	$9.16e+00$	$3.87e+01$	$8.90e+00$	$3.31e-01$	$5.90e-02$	$7.72e+01$	$1.23e+01$	$7.69e-11$	$4.83e-11$
$F3$	$4.09e+01$	$3.79e+01$	$6.99e+01$	$3.58e+01$	$3.10e+01$	$8.65e+00$	$5.64e+07$	$1.42e+08$	$6.38e-01$	$1.49e+00$
$F4$	$1.74e+02$	$4.29e+01$	$3.21e+01$	$1.38e+01$	$4.79e-02$	$2.01e-01$	$1.12e+02$	$2.74e+01$	$3.72e+02$	$8.68e+01$
$F5$	$6.39e-03$	$1.18e-02$	$9.86e-04$	$2.76e-03$	$0.00e+00$	$0.00e+00$	$9.02e-01$	$1.82e+00$	$2.16e-01$	$5.64e-01$
$F6$	$2.30e+00$	$1.54e+00$	$7.16e-14$	$1.86e-14$	$1.39e-13$	$9.43e-15$	$3.23e+00$	$2.44e+00$	$1.90e+01$	$1.02e+00$
$F7$	$2.80e-16$	$5.35e-16$	$2.22e-15$	$1.17e-15$	$8.88e-17$	$1.96e-16$	$1.23e-09$	$1.45e-09$	$2.10e+01$	$1.38e+01$
$F8$	$1.31e-01$	$2.64e-01$	$9.02e+10$	$0.00e+00$	$9.02e+10$	$0.00e+00$	$9.02e+10$	$9.02e+06$	$9.03e+10$	$9.39e+07$
$F9$	$1.44e+02$	$4.12e+01$	$2.85e+02$	$5.30e+00$	$2.73e+02$	$7.40e-01$	$3.11e+02$	$4.98e+00$	$3.16e+02$	$7.03e+00$
$F10$	$6.63e+00$	$5.57e+00$	$1.53e+00$	$1.29e+00$	$6.50e-29$	$3.60e-29$	$7.72e+00$	$2.93e+00$	$9.25e+00$	$2.82e+00$
$F11$	$1.39e+02$	$3.81e+01$	$9.65e-01$	$2.02e+00$	$6.26e-05$	$1.30e-05$	$1.01e-02$	$1.26e-02$	$1.95e+02$	$3.65e+01$
$F12$	$3.19e+01$	$3.65e+01$	$5.82e+00$	$1.03e+01$	$5.26e-13$	$1.64e-13$	$8.23e+01$	$1.53e+02$	$1.14e+02$	$1.01e+01$
$F13$	$1.13e+02$	$4.70e+01$	$5.97e+01$	$2.22e+01$	$2.48e+01$	$1.31e+00$	$1.43e+07$	$3.29e+07$	$1.16e+02$	$1.43e+01$
$F14$	$1.25e+02$	$2.49e+01$	$3.35e+01$	$1.86e+01$	$3.55e-08$	$2.26e-08$	$6.76e+01$	$1.30e+01$	$2.71e+02$	$7.30e+01$
$F15$	$4.20e-02$	$2.10e-01$	$2.29e-01$	$6.07e-01$	$1.99e-24$	$3.22e-24$	$3.07e+00$	$5.32e+00$	$3.94e+01$	$1.25e+02$
$F16$	$9.35e+01$	$4.98e+01$	$5.64e+00$	$8.47e+00$	$1.56e-09$	$2.81e-10$	$5.60e+01$	$5.16e+01$	$2.23e+02$	$1.50e+01$
$F17$	$2.98e+02$	$3.84e+01$	$1.51e+01$	$1.43e+01$	$8.52e-01$	$4.92e-01$	$7.61e+06$	$2.44e+07$	$3.47e+02$	$2.18e+01$
$F18$	$7.28e+01$	$7.22e+00$	$5.73e+00$	$5.26e+00$	$1.28e-04$	$4.63e-05$	$6.76e+01$	$3.46e+01$	$3.59e+02$	$8.45e+01$
$F19$	$2.76e+00$	$3.04e+00$	$1.23e+00$	$9.26e-01$	$2.00e-24$	$1.50e-24$	$1.95e+02$	$5.01e+02$	$1.71e+03$	$5.84e+03$
100-dimensional										
$F1$	$2.76e-12$	$5.37e-12$	$1.12e-16$	$4.28e-17$	$7.77e-17$	$1.13e-17$	$4.67e+02$	$7.02e+02$	$5.55e-17$	$1.26e-32$
$F2$	$3.64e+01$	$1.08e+01$	$7.74e+01$	$7.77e+00$	$4.60e+00$	$4.24e-01$	$9.96e+01$	$1.16e+01$	$2.61e-03$	$1.30e-02$
$F3$	$1.32e+02$	$8.18e+01$	$4.43e+02$	$3.63e+02$	$8.01e+01$	$1.03e+01$	$1.52e+08$	$2.69e+08$	$1.23e+01$	$1.80e+01$
$F4$	$4.86e+02$	$8.67e+01$	$1.01e+02$	$2.25e+01$	$9.53e-03$	$4.76e-02$	$2.92e+02$	$5.16e+01$	$8.38e+02$	$1.39e+02$
$F5$	$3.15e-03$	$7.81e-03$	$2.93e-02$	$5.32e-02$	$2.55e-17$	$5.19e-18$	$5.95e+00$	$1.29e+01$	$2.68e+00$	$1.05e+01$
$F6$	$5.14e+00$	$2.28e+00$	$1.55e+00$	$3.88e-01$	$3.10e-13$	$1.62e-14$	$4.79e+00$	$1.87e+00$	$1.86e+01$	$2.45e+00$
$F7$	$7.69e-13$	$2.72e-12$	$1.39e-14$	$7.12e-15$	$3.80e-17$	$5.29e-17$	$8.67e-02$	$3.70e-01$	$6.35e+01$	$2.36e+01$
$F8$	$2.71e+02$	$3.44e+02$	$1.79e+11$	$0.00e+00$	$1.79e+11$	$0.00e+00$	$1.79e+11$	$1.92e+07$	$1.80e+11$	$3.54e+08$
$F9$	$4.46e+02$	$5.74e+01$	$5.43e+02$	$1.36e+01$	$5.06e+02$	$9.16e-01$	$5.87e+02$	$1.01e+01$	$6.08e+02$	$1.07e+01$
$F10$	$2.50e+01$	$6.38e+00$	$1.54e+01$	$3.31e+00$	$1.35e-28$	$3.86e-29$	$2.89e+01$	$1.01e+01$	$1.93e+01$	$5.10e+00$
$F11$	$4.78e+02$	$6.32e+01$	$4.31e+01$	$2.09e+01$	$1.25e-04$	$1.43e-05$	$2.80e+01$	$3.02e+01$	$4.82e+02$	$4.27e+01$
$F12$	$1.43e+02$	$5.57e+01$	$7.21e+01$	$3.21e+01$	$6.44e-11$	$1.52e-11$	$8.72e+02$	$2.55e+03$	$2.41e+02$	$1.23e+01$
$F13$	$2.61e+02$	$5.82e+01$	$2.76e+02$	$6.18e+01$	$6.13e+01$	$1.00e+00$	$9.37e+07$	$4.02e+08$	$2.59e+02$	$2.16e+01$
$F14$	$3.31e+02$	$4.86e+01$	$9.37e+01$	$1.56e+01$	$4.48e-02$	$2.24e-01$	$2.25e+02$	$4.59e+01$	$6.19e+02$	$9.25e+01$
$F15$	$1.83e+00$	$2.30e+00$	$3.67e+00$	$1.76e+00$	$7.10e-23$	$7.00e-23$	$5.99e+00$	$1.19e+01$	$5.57e+01$	$5.22e+01$
$F16$	$3.39e+02$	$4.56e+01$	$1.10e+02$	$3.80e+01$	$1.94e-02$	$9.70e-02$	$2.08e+02$	$1.49e+02$	$4.84e+02$	$2.08e+01$
$F17$	$6.26e+02$	$6.07e+01$	$1.78e+02$	$5.49e+01$	$1.19e+01$	$2.62e+00$	$4.36e+07$	$7.09e+07$	$7.04e+02$	$3.92e+01$
$F18$	$1.81e+02$	$2.17e+01$	$1.04e+02$	$4.39e+01$	$2.92e-04$	$6.77e-05$	$2.37e+02$	$7.02e+01$	$1.09e+03$	$4.15e+02$
$F19$	$1.38e+01$	$6.03e+00$	$1.17e+01$	$2.61e+00$	$4.79e-23$	$2.65e-23$	$4.70e+02$	$1.84e+03$	$5.83e+03$	$9.85e+03$

defined as [10],

$$V_{\max} = 5000 \times n,$$

function evaluations. The algorithm's solution quality is measured according to its error from the provided optimal solution, i.e.,

$$err_{\text{alg}} = f\left(x_{\text{alg}}\right) - f\left(x^*\right),$$

where x^* is the optimal solution and x_{alg} is the best solution achieved by the algorithm.

The C programming language was used for the implementation of the proposed approach. A standard master-slave model was implemented under the OpenMPI library (http://www.open-mpi.org/) for the parallelization of the secondary swarms. All experiments were conducted on Intel® i7 machines providing 8 CPUs each, with 8GB RAM, running under Ubuntu Linux. The proposed approach and the base algorithms were run for 25 independent experiments, and the average value and standard deviation of the error values err_{alg} were recorded.

In previous works [14, 15], the central point of the grid was taken as the initial parameter setting. This is a reasonable choice in absence of any hints regarding the proper parametrization of the algorithm. In

Table 4: Average and standard deviation of error values for dimension $n = 200$ and 500.

Prob.	$\text{PSOPNA}_{0.5/1.5}$		DE_{bin}		DE_{exp}		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
200-dimensional										
$F1$	$6.10e-09$	$2.93e-08$	$6.39e-16$	$5.32e-16$	$1.78e-16$	$1.60e-17$	$9.61e+02$	$1.65e+03$	$1.17e-16$	$1.13e-17$
$F2$	$6.10e+01$	$8.44e+00$	$1.01e+02$	$5.90e+00$	$1.89e+01$	$1.05e+00$	$1.17e+02$	$7.60e+00$	$7.47e-02$	$2.44e-01$
$F3$	$3.07e+02$	$1.14e+02$	$6.38e+02$	$3.80e+02$	$1.79e+02$	$8.89e+00$	$2.54e+08$	$3.97e+08$	$1.24e+02$	$8.85e+01$
$F4$	$1.22e+03$	$1.34e+02$	$4.21e+02$	$5.63e+01$	$8.52e-02$	$3.98e-01$	$6.32e+02$	$8.43e+01$	$1.57e+03$	$1.54e+02$
$F5$	$4.20e-02$	$1.06e-01$	$3.00e-01$	$7.73e-01$	$7.49e-17$	$6.94e-18$	$1.02e+01$	$1.59e+01$	$1.13e+00$	$2.84e+00$
$F6$	$1.11e+01$	$4.22e+00$	$5.28e+00$	$9.65e-01$	$6.46e-13$	$2.53e-14$	$8.14e+00$	$2.26e+00$	$1.93e+01$	$7.39e-01$
$F7$	$2.77e-08$	$1.18e-07$	$1.78e-11$	$4.65e-11$	$2.25e-16$	$1.92e-16$	$3.95e-01$	$1.21e+00$	$1.25e+02$	$1.92e+01$
$F8$	$1.01e+04$	$1.30e+04$	$8.33e+11$	$0.00e+00$	$8.33e+11$	$0.00e+00$	$8.33e+11$	$3.09e+08$	$8.56e+11$	$3.36e+09$
$F9$	$1.18e+03$	$7.39e+01$	$1.13e+03$	$1.87e+01$	$1.01e+03$	$1.30e+00$	$1.18e+03$	$8.29e+00$	$1.22e+03$	$1.79e+01$
$F10$	$5.64e+01$	$9.29e+00$	$5.52e+01$	$1.02e+01$	$2.77e-28$	$5.31e-29$	$7.34e+01$	$6.25e+01$	$3.76e+01$	$2.78e+01$
$F11$	$1.17e+03$	$6.83e+01$	$3.95e+02$	$6.11e+01$	$2.55e-04$	$3.20e-05$	$4.03e+02$	$8.45e+01$	$1.08e+03$	$8.25e+01$
$F12$	$3.56e+02$	$5.83e+01$	$2.84e+02$	$4.97e+01$	$9.97e-10$	$2.01e-10$	$8.11e+02$	$1.58e+03$	$5.87e+02$	$4.52e+02$
$F13$	$6.25e+02$	$8.59e+01$	$7.52e+02$	$2.71e+02$	$1.40e+02$	$1.26e+01$	$2.06e+08$	$3.51e+08$	$5.92e+02$	$1.08e+02$
$F14$	$8.96e+02$	$1.17e+02$	$3.11e+02$	$3.66e+01$	$8.08e-03$	$4.04e-02$	$4.90e+02$	$5.23e+01$	$1.26e+03$	$1.81e+02$
$F15$	$6.30e+00$	$3.07e+00$	$1.17e+01$	$2.85e+00$	$3.71e-24$	$2.32e-24$	$1.40e+01$	$9.80e+00$	$1.95e+02$	$1.66e+02$
$F16$	$7.60e+02$	$6.07e+01$	$5.58e+02$	$7.96e+01$	$7.85e-09$	$1.11e-09$	$6.77e+02$	$6.04e+02$	$9.56e+02$	$3.33e+01$
$F17$	$1.29e+03$	$8.24e+01$	$1.03e+03$	$1.11e+02$	$3.71e+01$	$8.30e-01$	$1.17e+07$	$1.70e+07$	$1.49e+03$	$8.00e+01$
$F18$	$4.26e+02$	$4.56e+01$	$7.53e+02$	$6.55e+01$	$5.10e-04$	$9.97e-05$	$7.67e+02$	$2.14e+02$	$3.94e+03$	$3.91e+03$
$F19$	$4.16e+01$	$6.67e+00$	$4.04e+01$	$7.71e+00$	$1.67e-22$	$7.58e-23$	$7.51e+02$	$1.76e+03$	$2.53e+04$	$2.45e+04$
500-dimensional										
$F1$	$1.15e+02$	$3.81e+02$	$3.88e-05$	$7.93e-05$	$5.17e-16$	$1.36e-17$	$9.25e+02$	$1.27e+03$	n/a	n/a
$F2$	$9.65e+01$	$1.43e+01$	$1.25e+02$	$5.37e+00$	$5.38e+01$	$1.21e+00$	$1.35e+02$	$5.52e+00$	n/a	n/a
$F3$	$1.24e+07$	$5.97e+07$	$3.44e+04$	$1.62e+05$	$4.74e+02$	$1.48e+00$	$6.93e+08$	$1.72e+09$	n/a	n/a
$F4$	$3.81e+03$	$2.19e+02$	$2.35e+03$	$1.60e+02$	$7.12e-01$	$9.64e-01$	$2.11e+03$	$1.66e+02$	n/a	n/a
$F5$	$8.57e-01$	$1.82e+00$	$3.11e-01$	$5.07e-01$	$2.38e-16$	$1.18e-17$	$1.45e+01$	$2.52e+01$	n/a	n/a
$F6$	$1.91e+01$	$3.69e-01$	$1.49e+01$	$8.38e-01$	$1.64e-12$	$4.85e-14$	$1.27e+01$	$1.26e+00$	n/a	n/a
$F7$	$5.89e-02$	$1.96e-01$	$2.74e-03$	$6.49e-03$	$7.29e-16$	$3.58e-16$	$3.33e-05$	$1.12e-04$	n/a	n/a
$F8$	$1.66e+05$	$1.23e+05$	$4.94e+12$	$0.00e+00$	$4.94e+12$	$0.00e+00$	$4.94e+12$	$1.42e+08$	n/a	n/a
$F9$	$3.29e+03$	$1.18e+02$	$2.97e+03$	$3.17e+01$	$2.52e+03$	$2.10e+00$	$3.00e+03$	$1.64e+01$	n/a	n/a
$F10$	$1.11e+02$	$2.90e+01$	$1.36e+02$	$2.08e+01$	$9.79e-28$	$1.43e-28$	$1.64e+02$	$5.62e+01$	n/a	n/a
$F11$	$3.35e+03$	$1.14e+02$	$2.34e+03$	$9.22e+01$	$6.78e-04$	$3.60e-05$	$1.67e+03$	$1.44e+02$	n/a	n/a
$F12$	$1.09e+03$	$2.33e+02$	$1.02e+03$	$6.68e+01$	$6.80e-09$	$8.58e-10$	$1.62e+03$	$1.83e+03$	n/a	n/a
$F13$	$3.39e+05$	$1.68e+06$	$2.49e+03$	$3.13e+02$	$3.60e+02$	$9.23e+00$	$3.41e+08$	$4.29e+08$	n/a	n/a
$F14$	$2.91e+03$	$1.84e+02$	$1.67e+03$	$1.51e+02$	$3.93e-01$	$1.05e+00$	$1.59e+03$	$1.57e+02$	n/a	n/a
$F15$	$2.47e+01$	$7.28e+00$	$4.44e+01$	$5.59e+00$	$2.93e-18$	$7.16e-18$	$3.50e+01$	$1.20e+01$	n/a	n/a
$F16$	$2.03e+03$	$2.84e+02$	$2.02e+03$	$8.60e+01$	$2.05e-08$	$1.64e-09$	$1.92e+03$	$1.44e+03$	n/a	n/a
$F17$	$3.53e+03$	$2.64e+02$	$3.83e+03$	$1.41e+02$	$1.12e+02$	$1.02e+00$	$6.64e+08$	$1.64e+09$	n/a	n/a
$F18$	$1.31e+03$	$1.38e+02$	$3.37e+03$	$4.34e+02$	$1.25e-03$	$1.87e-04$	$2.74e+03$	$3.59e+02$	n/a	n/a
$F19$	$9.13e+01$	$2.01e+01$	$1.29e+02$	$2.34e+01$	$3.35e-21$	$2.15e-21$	$2.05e+03$	$4.03e+03$	n/a	n/a

the considered 3-dimensional grids of the PSO parameters, the central point corresponds to $(w, c_1, c_2) = (0.5, 1.5, 1.5)$, which happens to be an admittedly efficient parameter setting. Thus, in order to obtain unbiased evidence of the benefits of the proposed approach, we also considered two extremal initial points closer to the boundaries of the grid, i.e.,

$$(w, c_1, c_2) = (0.2, 1.0, 1.0), \qquad (w, c_1, c_2) = (0.8, 2.0, 2.0).$$

The corresponding PSOPNA instances for the three initial parameter settings are henceforth denoted as $\text{PSOPNA}_{0.5/1.5}$, $\text{PSOPNA}_{0.2/1.0}$, and $\text{PSOPNA}_{0.8/2.0}$. Following the setting requirements of the test suite, the swarm size was fixed to the value,

$$N = 60,$$

for all test problems and dimensions.

The experimental assessment consisted of two phases. In the first phase, the three PSOPNA instances were compared against their plain PSO counterparts with the corresponding (fixed) initial parameter sets

and neighborhood radius $m = 1$. Pairwise Wilcoxon ranksum tests were conducted at confidence level 95% for all test problems. A favorable comparison was counted as a win for the PSOPNA approach and denoted with "$+$". Respectively, negative comparisons are denoted with "$-$", and ties (no statistical difference between the algorithms) is denoted as "$=$".

Table 1 summarizes the results. It is clearly seen that for the two extremal initial points PSOPNA dramatically improved the performance over the corresponding plain PSO approaches. Note that the improvement was achieved without any additional preprocessing or preliminary experimentation. Even for the case of the near-optimal initial parameter setting, the proposed approach achieved better or equivalent performance in more than 60% of the problems. Also, we observe that increasing the dimension from 50 to 500 does not radically change the observed performance, which is a trait of nice scaling properties.

Table 2 reports the results of the second experimentation phase where the $\text{PSOPNA}_{0.5/1.5}$ version is compared against the base algorithms of the test suite. The specific PSOPNA approach was considered as it is the one that would be most probably selected by a practitioner. As we can see, PSOPNA was competitive against two of the base algorithms, namely CHC and GCMAES, while it was outperformed by the DE variants. However, it shall be noted that plain PSO was completely out of competition against all base algorithms, while the dominant DE algorithm was shown to be the best one for the specific test suite.

Moreover, it shall be emphasized that all base algorithms assumed their optimal parameter settings specifically fine-tuned for the considered test suite, while PSOPNA did not spent any function evaluations on fine-tuning. Thus, in a completely fair comparison, PSOPNA should be receiving PSOPNA also the additional computational budget that is spent by the rest of the algorithms for their laborious fine-tuning.

For completeness purpose, the achieved averages and standard deviations of the obtained solution values of PSOPNA and the base algorithms are reported in Tables 3-4 for all test problems. Missing values due to excessive running time are denoted with "n/a" for the GCMAES approach in the 500-dimensional case.

5 Conclusion

The performance of metaheurstics has been shown to be dependent on their proper parameterization. Parameter tuning is a laborious and complex task that needs significant amount of time and computational resources. The present work extends a recently proposed grid-based parameter adaptation technique, which was successfully applied on the DE algorithm. The PSO algorithm is equipped with this technique in the present work, in order to verify its wide applicability regardless of the considered algorithm and its initial parameterization.

The proposed technique is used to adaptively control the scalar parameters and the neighborhood radius of the PSO algorithm. Moreover, a diversity-oriented evaluation measure for the secondary swarms is proposed in order to promote diversity and deter premature convergence. The experiments were conducted on an established large-scale optimization test suite, following the setting in previous studies. The results were promising, showing significant performance boost of PSO, while relieving the user from the burden of parameter tuning.

Future research will include more efficient performance-estimation techniques for the secondary swarms, more sophisticated search procedures in the parameter space (currently under development), as well as more challenging problems.

References

[1] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Proc. of the 2005 IEEE Congress on Evolutionary Computation*, pages 769–1776, 2005.

[2] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation*. Springer-Verlag, Berlin, 2006.

[3] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-race and iterated f-race: An overview. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[4] T. Cai, F. Pan, and J. Chen. Adaptive particle swarm optimization algorithm. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 3, pages 2245–2247 Vol.3, June 2004.

[5] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[6] A. E. Eiben and S. K. Smit. Evolutionary algorithm parameters and methods to tune them. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, chapter 2, pages 15–36. Springer, Berlin Heidelberg, 2011.

[7] L. J. Eshelman and Schaffer J. D. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2:187–202, 1993.

[8] H. H. Hoos. Automated algorithm configuration and parameter tuning. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, chapter 3, pages 37–72. Springer, Berlin Heidelberg, 2011.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[10] M. Lozano, F. Herrera, and D. Molina. Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15:2085–2087, 2011.

[11] K. E. Parsopoulos. Particle swarm methods. In M.G.c. Resende, R. Marti, and P. Pardalos, editors, *Handbook of Heuristics*. Springer, 2016.

[12] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global), 2010.

[13] Y. Shi and R. C. Eberhart. Fuzzy adaptive particle swarm optimization. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 101–106 vol. 1, 2001.

[14] V. A. Tatsis and K. E. Parsopoulos. Differential evolution with grid-based parameter adaptation. *Soft Computing*, pages 1–23, 2015.

[15] V. A. Tatsis and K. E. Parsopoulos. Grid search for operator and parameter control in differential evolution. In *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, SETN '16, pages 7:1–7:9, New York, NY, USA, 2016. ACM.

[16] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381, Dec 2009.