Available online at www.sciencedirect.com
ScienceDirect
Mathematical and Computer Modelling 46 (2007) 198–213
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

Parameter selection and adaptation in Unified Particle Swarm Optimization

K.E. Parsopoulos, M.N. Vrahatis*

Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras, GR-26110 Patras, Greece
University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece

Received 18 May 2006; accepted 15 December 2006

Abstract

The performance of the recently proposed Unified Particle Swarm Optimization method is investigated under different schemes for the determination and adaptation of the unification factor, which is the main parameter of the method, controlling its exploration and exploitation properties. Widely used benchmark problems are employed and numerous experiments are conducted along with statistical tests to yield useful conclusions regarding the effect of the parameter on the algorithm's performance as well as the most efficient adaptation schemes.

Keywords: Particle Swarm Optimization; Unified Particle Swarm Optimization; Parameter selection; Parameter adaptation

1. Introduction

Swarm intelligence is a relatively new category of stochastic, population-based optimization algorithms. These algorithms are closely related to evolutionary algorithms that are based on procedures that imitate natural evolution [1,2]. Swarm intelligence algorithms draw inspiration from the collective behavior and emergent intelligence that arise in socially organized populations. They have been designed primarily to address problems that cannot be tackled through traditional optimization algorithms. Such problems are characterized by discontinuities, lack of derivative information, noisy function values and disjoint search spaces [3,4]. Ant Colony Optimization was the first widely used swarm intelligence algorithm and has been successfully applied on numerous discrete optimization tasks [3,5].

Particle Swarm Optimization (PSO) is a swarm intelligence algorithm that is used mainly for numerical optimization tasks [4,6]. PSO gained increasing popularity in recent years due to its ability to solve efficiently and effectively a plethora of problems in science and engineering [7–17]. *Unified Particle Swarm Optimization* (UPSO) was recently introduced [17] as a modification of PSO that aggregates its local and global variant, combining their exploration and exploitation abilities without imposing additional requirements in terms of function evaluations. Convergence in probability was proved for a version of the new method and preliminary experimental results on

* Corresponding author at: Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras, GR-26110 Patras, Greece.
E-mail addresses: kostasp@math.upatras.gr (K.E. Parsopoulos), vrahatis@math.upatras.gr (M.N. Vrahatis).

0895-7177/$ - see front matter
doi:10.1016/j.mcm.2006.12.019

both static and dynamic benchmark problems suggested that UPSO is superior than both the global and local variant of the standard PSO [14,17]. In UPSO's main scheme, a new parameter, called the *unification factor*, was introduced in order to control the influence of the local and global PSO variants in the UPSO scheme.

We investigate experimentally the performance of UPSO using different schemes for the selection and adaptation of the unification factor. For this purpose, the DeJong suite of benchmark problems is used and six different adaptation schemes that cover a wide range of adaptation techniques [18] are employed. Experiments are performed and analyzed to justify UPSO's superiority against standard PSO and provide intuition regarding the most promising values of the unification factor and its adaptation schemes. The rest of the paper is organized as follows. PSO and UPSO are described in Section 2. The schemes for the selection and adaptation of UPSO's parameter are described in Section 3. Experimental results are reported in Section 4 and the paper closes with conclusions in Section 5.

2. Descriptions of the algorithms

For completeness purposes, we describe PSO and discuss its main issues prior to the presentation of UPSO.

2.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced in 1995 by Eberhart and Kennedy as a numerical optimization algorithm [6,19]. The main ideas behind the development of PSO stem from the fields of social psychology and evolutionary computation. PSO's dynamic is governed by fundamental laws encountered in swarms in nature and it follows the five principles of swarm intelligence [20], namely *proximity* (i.e., ability to perform simple space and time computations), *quality* (i.e., ability to respond to quality factors in the environment), *diverse response* (i.e., the method should not commit its activities along excessively narrow channels), *stability* (i.e., the behavior of the method must not change with small changes in the environment) and *adaptability* (i.e., the method must be able to alter its behavior when the computational cost is not prohibitive); hence it was categorized as a swarm intelligence algorithm [4].

Similarly to other population-based algorithms, PSO exploits a population of search points to probe the search space. In the context of PSO, the population is called a *swarm*, while the search points are called *particles*, following the notation in early precursors of the algorithm that were used for the simulation of swarms [4]. Each particle moves in the search space with an adaptable velocity, recording the best position it has ever visited in the search space, i.e., (in minimization problems) the position with the lowest function value. The adaptation of the velocity is based on information coming from the particle itself, as well as from the rest of the particles. More specifically, each particle has a "neighborhood" that consists of some prespecified particles and the best position ever attained by any member of the neighborhood is communicated to the particle and influences its movement.

Assume the problem of minimizing an n-dimensional function,

$$\min_{X \in S} f(X),$$

with $S \subset \mathbb{R}^n$. Then, a swarm to tackle this problem consists of N particles,

$$\mathbb{S} = \{X_1, X_2, \ldots, X_N\},$$

which are n-dimensional vectors, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in S$, $i = 1, \ldots, N$. The *velocity*, V_i, of the ith particle, as well as its *best position*, P_i, are also n-dimensional vectors,

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top,$$
$$P_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in S, \quad i = 1, \ldots, N.$$

The neighborhoods are usually defined based on the particles' indices. The most common neighborhood topology is the "ring" topology, where the neighborhood of a particle consists of particles with neighboring indices. Thus, a neighborhood of radius m of X_i is the set

$$\mathbb{X}_i = \{X_{i-m}, \ldots, X_i, \ldots, X_{i+m}\},$$

where the particle X_1 is assumed to follow immediately after X_N. A thorough investigation of different neighborhood topologies is provided in [21].

Let g_i denote the index of the particle that attained the best previous position among all the particles in the neighborhood of X_i, i.e.,

$$f(P_{g_i}) \leqslant f(P_j), \quad \forall\, j \in \{i-m, \ldots, i+m\},$$

and let t be the iteration counter. Then, the velocity and position of X_i are updated according to the equations [22–24],

$$V_i(t+1) = \chi[V_i(t) + c_1 r_1 (P_i(t) - X_i(t)) + c_2 r_2 (P_{g_i}(t) - X_i(t))], \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{2}$$

where χ is a parameter called the *constriction coefficient*; c_1, c_2 are positive acceleration parameters called *cognitive* and *social* parameter, respectively; and r_1, r_2 are vectors with components uniformly distributed in the range [0, 1]. All vector operations in Eqs. (1) and (2) are performed componentwise. The best positions are updated at each iteration according to

$$P_i(t+1) = \begin{cases} X_i(t+1), & \text{if } f(X_i(t+1)) < f(P_i(t)), \\ P_i(t), & \text{otherwise.} \end{cases}$$

The constriction coefficient was incorporated in PSO as a means to control the magnitude of the velocities, alleviating the "swarm explosion" effect that was detrimental for the convergence of early PSO versions [25]. In those versions, the parameters were determined empirically, based on extensive experimentation on benchmark problems. In recent versions, the results reported in the stability analysis due to Clerc and Kennedy [23,24] imply that parameters are selected such that the relation,

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \tag{3}$$

holds for $\phi > 4$, where $\phi = c_1 + c_2$, to promote convergence. Also, in early PSO versions, a threshold, $V_{\max}$, was imposed on the absolute value of the velocity vector's components. Although its use is not necessary in the constriction coefficient version of PSO, experimental results indicate that it can enhance the algorithm's performance [22]. The initialization of the swarm and velocities is usually performed randomly in the search space, following a uniform distribution. The best positions are initially set equal to the initial swarm.

The search procedure of a population-based algorithm such as PSO consists of two main phases, *exploration* and *exploitation*. The former is responsible for the detection of the most promising regions in the search space, while the latter promotes convergence of the particles towards the best detected solution. These two phases can take place either once or successively during the execution of the algorithm.

The concept of neighborhood gave rise to two main variants of PSO, with respect to the number of particles that comprise the neighborhoods. In the *global* variant, the whole swarm is considered as the neighborhood for every particle. On the other hand, in the *local* variant, the neighborhood size is strictly smaller than the size of the swarm. Although the global variant is actually a generalization of the local variant, we distinguish between them due to their different exploration and exploitation properties. More specifically, the global variant converges faster than the local one, since all particles are attracted by the same best position. Therefore, it is distinguished for its exploitation ability. On the other hand, the local variant has better exploration properties, since the information regarding the best position of each neighborhood is gradually communicated to the rest of the particles through their neighbors in the ring topology. Thus, the attraction towards a specific point is weaker, preventing the swarm from getting trapped in suboptimal solutions. Evidently, proper selection of the neighborhood size affects PSO's trade-off between exploration and exploitation, albeit there is no formal procedure to determine the optimal size.

2.2. Unified Particle Swarm Optimization

Unified Particle Swarm Optimization (UPSO) was recently proposed as a scheme that harnesses the local and global PSO variants, combining their exploration and exploitation properties [17]. Let X_i be the ith particle of the swarm, g be the index of the best particle in the whole swarm and g_i be the index of the best particle in the neighborhood of X_i, as described in the previous section. Also, let $\mathcal{G}_i(t+1)$ be the velocity update of X_i for the global PSO variant,

let $\mathcal{L}_i(t+1)$ be the velocity update of X_i for the local PSO variant, and t denote the iteration counter. Then, from Eq. (1), it holds that [17],

$$\mathcal{G}_i(t+1) = \chi\left[V_i(t) + c_1 r_1(P_i(t) - X_i(t)) + c_2 r_2(P_g(t) - X_i(t))\right], \tag{4}$$

$$\mathcal{L}_i(t+1) = \chi\left[V_i(t) + c_1 r_1'(P_i(t) - X_i(t)) + c_2 r_2'(P_{g_i}(t) - X_i(t))\right]. \tag{5}$$

The aggregation of the search directions defined by Eqs. (4) and (5) results in the main UPSO scheme [17],

$$\mathcal{U}_i(t+1) = u\,\mathcal{G}_i(t+1) + (1-u)\,\mathcal{L}_i(t+1), \quad u \in [0,1], \tag{6}$$

$$X_i(t+1) = X_i(t) + \mathcal{U}_i(t+1). \tag{7}$$

The parameter u is called the *unification factor* and it balances the influence of the global and local search directions. The standard global PSO variant is obtained by setting $u = 1$ in Eq. (6), while $u = 0$ results in the standard local PSO variant. All intermediate values of $u \in (0, 1)$ define composite UPSO variants that combine the exploration and exploitation properties of the global and local PSO variant.

Besides the basic UPSO scheme, a stochastic parameter can also be incorporated in Eq. (6) to enhance UPSO's exploration capabilities [17]. Thus, depending on which variant UPSO is mostly based, Eq. (6) becomes [17]

$$\mathcal{U}_i(t+1) = r_3\,u\,\mathcal{G}_i(t+1) + (1-u)\,\mathcal{L}_i(t+1), \tag{8}$$

which is mostly based on the local variant, or

$$\mathcal{U}_i(t+1) = u\,\mathcal{G}_i(t+1) + r_3\,(1-u)\,\mathcal{L}_i(t+1), \tag{9}$$

which is mostly based on the global variant. The parameter $r_3 \sim \mathcal{N}(M, \Sigma)$ is a normally distributed parameter with mean vector M and covariance matrix Σ. The use of r_3 imitates mutation in evolutionary algorithms. However, the mutation in UPSO is biased towards directions that are consistent with the PSO dynamic, in contrast to the pure random mutation used in evolutionary algorithms. Following the assumptions of Matyas [26], a proof of convergence in probability was derived for the UPSO variants of Eqs. (8) and (9) [17].

3. Parameter selection and adaptation schemes

As already mentioned in the preceding section, the unification factor, u, controls the balance between the exploration and exploitation capabilities of UPSO. Small values of u (close to 0) in Eq. (6) favor the local PSO variant component, thereby resulting in better exploration. On the other hand, large values (close to 1) favor the global variant component, promoting exploitation. Therefore, values of u around the middle point, 0.5, of the range [0, 1] result in the most balanced UPSO versions, with respect to its exploration and exploitation abilities. However, such balanced versions do not take full advantage of any special structure of the problem at hand (e.g., unimodality, convexity etc.). In such cases, unification factors that are biased towards 0 or 1 may exhibit better performance. Moreover, on-line adaptation of the unification factor during the execution of the algorithm can enhance its performance.

The unification factor can be determined either in the *swarm-level*, i.e., the same u is assigned to all particles, or in the *particle-level*, i.e., each particle assumes a different value of u [18]. In the first case, the particles share the same exploration–exploitation abilities resulting in swarms with uniform behavior of the particles, while in the latter case, each particle has its own special exploration/exploitation ability, resulting in swarms with higher diversity of behaviors.

Optimal values of the unification factor depend always on the problem at hand. We considered different schemes for the selection and adaptation of u to gain intuition regarding UPSO's performance and we describe them in the following paragraphs.

3.1. Quantized unification factor

This is a swarm-level scheme where all particles have the same, fixed unification factor value. We considered the set,

$$W = \{0.0, 0.1, 0.2, \ldots, 1.0\},$$

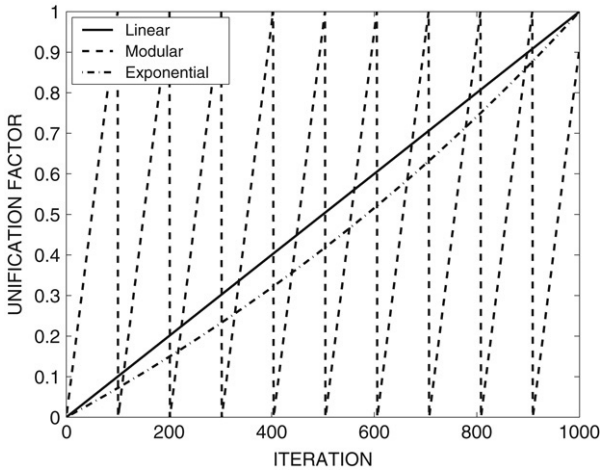

Fig. 1. The three different increasing schemes of the unification factor.

of distinct, equidistant values in [0, 1] and we investigated each one separately, in order to gain intuition regarding the most promising values per problem.

3.2. Increasing unification factor

According to this swarm-level scheme, all particles have the same unification factor which is initialized at 0 and is increased up to 1 during the execution of the algorithm. Thus, exploration is favored in the first stages of the algorithm's execution (since $u = 0$ corresponds to the local PSO variant) and exploitation is promoted at the final stages, where u assumes higher values.

Let t denote the iteration number, $u(t)$ be the unification factor at iteration t, and MIT denotes the maximum number of iterations. Three different increasing schedules were considered in our study:

1. *Linear*. The unification factor is linearly increased from 0 up to 1 according to the scheme,
$$u(t) = \frac{t}{MIT}.$$
This scheme corresponds to a smooth and relatively slow transition from exploration to exploitation.
2. *Modular*. The unification factor increases repeatedly from 0 to 1 every q iterations according to the scheme,
$$u(t) = \frac{t \bmod (q+1)}{q}.$$
This scheme modifies repeatedly the dynamic of the algorithm from exploration to exploitation throughout its execution. In our experiments we used $q = 100$.
3. *Exponential*. The unification factor increases from 0 to 1 exponentially according to the scheme,
$$u(t) = \exp\left(\frac{t \log(2.0)}{MIT}\right) - 1.0.$$
This scheme results in slow transition from exploration to exploitation in the early stages of the algorithm's execution, but it exhibits faster transition in the later stages.

The three increasing schemes are depicted in Fig. 1 for $MIT = 1000$.

3.3. Sigmoid increasing unification factor

This swarm-level scheme is similar to the increasing scheme described in the previous paragraph. All particles assume the same unification factor, which is increased according to the relation,

$$u(t) = F_{\text{sig}}\left(t - \frac{MIT}{20}, \lambda\right),$$

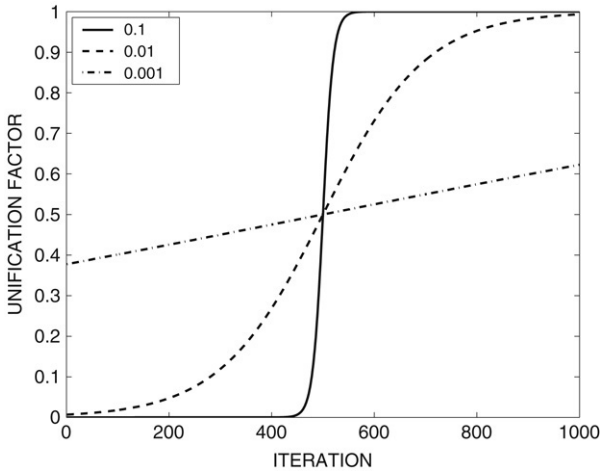

Fig. 2. Sigmoid transitions of the unification factor for different values of λ.

where,

$$F_{\text{sig}}(x, \lambda) = \frac{1}{1 + \exp(-\lambda x)}.$$

This scheme results in a sigmoid transition from exploration to exploitation and we treat it as a special increasing scheme since the form of the sigmoid depends on the value of the parameter λ. In our experiments we investigated the values $\lambda = 10^{-1}, 10^{-2}$ and 10^{-3}. The corresponding sigmoids are depicted in Fig. 2 for $MIT = 1000$. The selection of the sigmoid function was triggered by its wide applicability as an activation function in artificial neural networks.

3.4. Swarm partitioning

This is a particle-level scheme where the swarm is divided in partitions that consist of a prespecified number of particles, which is called the *partition's size*. All particles in a partition share the same value of u and each partition has a different value of u. In our experiments, the swarm was partitioned into 11 partitions, each having a value of $u \in W = \{0.0, 0.1, 0.2, \ldots, 1.0\}$. Depending on the swarm size, it is possible to have partitions with different sizes. For example, a swarm of size 100 can be partitioned in 10 partitions of size 9 and one partition of size 10. The small difference in the size of the last partition is not expected to modify the dynamic of the scheme significantly.

In order to have neighborhoods that consist of particles that belong in different partitions, the assignment of particles to partitions is not performed sequentially, e.g., the first 10 particles to partition 1, the next 10 particles to partition 2 etc. Instead, the ith particle is assigned to the $(1 + ((i - 1) \bmod 11))$th partition. Thus, the first 11 particles, $X_1, \ldots, X_{11}$, are assigned to partitions 1–11, respectively, one to each partition. Then, X_{12} is assigned to partition 1, X_{13} is assigned to partition 2 etc. Using this scheme, particles with different behaviors can interact by sharing information through their neighborhoods, while retaining an almost equal influence of each distinct value of the unification factor to the swarm's behavior, regardless of its performance.

3.5. Dominance of the best

This is also a particle-level scheme. It works similarly to Swarm Partitioning, but after a specific number of iterations, the partition where the best particle of the swarm belongs (i.e., the particle with the best position ever) gains an additional particle from the partition where the particle with the worst among the best positions belongs. The minimum size of a partition is set to 1 to prevent losing the behavioral diversity of the swarm. If the size of the worst partition is 1, then a particle from the immediately worst partition with size higher than 1 is rendered to the best partition.

The Dominance of the Best scheme incorporates an award system for the best partition in order to strengthen the influence of the best unification factor in the swarm, although it preserves the existence of all different values of u by

Table 1
The DeJong benchmark problems suite

Prob.	Name	Formula	Dim.	Range	Goal
1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	10^{-2}
2	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	$[-30, 30]^n$	10^2
3	Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12, 5.12]^n$	10^2
4	Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \frac{x_i}{\sqrt{i}} + 1$	30	$[-600, 600]^n$	10^{-1}
5	Schaffer's F6	$f(x) = 0.5 + \frac{(\sin \sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2}$	2	$[-100, 100]^2$	10^{-5}

For each problem, its formula, dimension, range and desired error goal are reported.

posing a minimum partition size. This mechanism prevents the swarm from becoming completely biased towards the best performing unification factor, which could be detrimental to its overall performance.

3.6. Self-adaptation

In this particle-level scheme, each particle has its own unification factor, which is incorporated in the particle vector as an additional variable, augmenting the dimension of the problem. Thus, the particle X_i is defined as

$$X_i = (x_{i1}, \ldots, x_{in}, u_i)^\top \in S \times [0, 1],$$

where n is the dimension of the original problem and u_i is the unification factor of X_i. According to the Self-Adaptation scheme, the dynamic of UPSO is allowed to determine itself the optimal u for each particle, individually, capturing any special structure of the problem at hand.

4. Results and discussion

The DeJong test suite, which is widely used in evolutionary computation and was also used in established works on PSO [22–24], was considered for the investigation of UPSO's performance under the different schemes for the determination of the unification factor that were described in the previous section. The definitions of the test problems as well as their dimension, admissible range of the parameters and error goal values are reported in Table 1.

For each test problem, 100 independent experiments were conducted. The swarm size was set to 30 in all cases due to its promising results, with respect to the expected number of required function evaluations, reported in [24]. The swarm was allowed to perform a maximum number of 10 000 iterations. The particles were constrained in the ranges reported in Table 1 and a maximum value, $V_{\max}$, equal to half the range of the parameters was imposed on the velocities. The PSO parameters were set to their default values, $\chi = 0.729$, $c_1 = c_2 = 2.05$ [23,24]. For the computation of the local PSO components in UPSO, a neighborhood of radius 1 was used, in order to take full advantage of its exploration properties [17].

The Quantized, Increasing, Sigmoid, Swarm Partitioning, Dominance of the Best and Self-Adaptive schemes for the determination of u are denoted as "QUA", "INC", "SIG", "PAR", "DOM" and "SEL", respectively, with "Lin", "Mod" and "Exp" denoting the Linear, Modular and Exponential case of INC. In the Sigmoid scheme, the parameters $\lambda = 10^{-1}, 10^{-2}$ and 10^{-3} were used for the transition of the unification factor from 0 to 1 in the first 1000 iterations. The decision to complete the transition in 1000 iterations instead of the maximum number of iterations was based on the fact that in most cases PSO needed less than 1000 iterations to converge and our intention was to study its performance using the whole range of values that can be assigned to u in the sigmoid case.

In Swarm Partitioning, since the swarm size was 30 and the number of partitions was 11, we considered some partitions of 3 particles and some partitions of 2 particles. More specifically, partitions of $u = 0.0, 0.1, \ldots, 0.7$, consisted of 3 particles, while the remaining partitions consisted of 2 particles. The same also held for the initialization of the Dominance of the Best scheme. Additionally in this scheme, the update of the partitions was performed every 20 iterations.

Table 2
Results for Test Problem 1

Type	Param.	Suc.	Mean	St.D.	Min.	Max.	ENFE
QUAN	0.0	1.00	5.698×10^2	5.757×10^1	4.080×10^2	6.930×10^2	1.709×10^4
	0.1	1.00	2.768×10^2	2.674×10^1	2.040×10^2	3.540×10^2	8.304×10^3
	0.2	1.00	2.307×10^2	2.258×10^1	1.850×10^2	2.920×10^2	6.920×10^3
	0.3	1.00	2.062×10^2	1.856×10^1	1.670×10^2	2.560×10^2	6.185×10^3
	0.4	1.00	1.930×10^2	1.513×10^1	1.560×10^2	2.440×10^2	5.791×10^3
	0.5	1.00	1.921×10^2	1.569×10^1	1.460×10^2	2.400×10^2	5.764×10^3
	0.6	1.00	1.948×10^2	1.820×10^1	1.620×10^2	2.620×10^2	5.845×10^3
	0.7	1.00	1.993×10^2	1.633×10^1	1.650×10^2	2.520×10^2	5.979×10^3
	0.8	1.00	2.214×10^2	2.021×10^1	1.590×10^2	2.720×10^2	6.642×10^3
	0.9	1.00	2.592×10^2	2.007×10^1	2.160×10^2	3.160×10^2	7.776×10^3
	1.0	0.91	1.231×10^3	2.772×10^3	3.100×10^2	1.000×10^4	1.199×10^4
INC	Lin	1.00	4.017×10^2	2.735×10^1	3.420×10^2	4.760×10^2	1.205×10^4
	Mod	1.00	2.272×10^2	1.987×10^1	1.760×10^2	2.690×10^2	6.815×10^3
	Exp	1.00	4.294×10^2	2.738×10^1	3.660×10^2	5.110×10^2	1.288×10^4
SIG	10^{-1}	1.00	5.005×10^2	2.675×10^1	4.040×10^2	5.620×10^2	1.501×10^4
	10^{-2}	1.00	3.408×10^2	2.237×10^1	2.830×10^2	4.050×10^2	1.022×10^4
	10^{-3}	1.00	1.962×10^2	1.825×10^1	1.490×10^2	2.580×10^2	5.886×10^3
PAR		1.00	2.521×10^2	2.105×10^1	2.020×10^2	3.320×10^2	7.563×10^3
DOM		1.00	2.278×10^2	2.410×10^1	1.890×10^2	3.500×10^2	6.834×10^3
SELF		1.00	2.007×10^2	1.548×10^1	1.610×10^2	2.390×10^2	6.020×10^3

In the Self-Adaptive scheme, the unification factor of each particle was randomly initialized, using a uniform distribution, and constrained within [0.3, 0.6]. Initially, the full range [0.0, 1.0] was used; however, the algorithm was unable to reach the desired goal within the maximum number of iterations, although it was moving close to the global minimizer of the problem. This inability can be attributed to the increase in the dimension of the problem after the inclusion of u in the particle. Thus, either the maximum number of iterations should be increased or a smaller range for u should be used in order to have a fair comparison with the other adaptation schemes. To this end, we chose to retain the maximum number of iterations and constrain u in [0.3, 0.6], since the same maximum number of iterations was used in related works [22,24]. The selection of [0.3, 0.6] was made due to the promising results obtained through the Quantized scheme for this range.

For each test problem, the success rate, i.e., the fraction of the number of experiments (out of 100 experiments) in which the goal was attained, as well as the mean, standard deviation, minimum and maximum value of the required iterations were recorded. Additionally, the expected number of function evaluations (ENFE), defined as [24]

$$\text{ENFE} = \frac{(\text{Number of Particles}) \times (\text{Average Number of Iterations})}{(\text{Success Rate})},$$

was computed for each case. For the computation of the average number of iterations only the successful experiments were considered [24].

For each pair of algorithms, a t-test was performed to check the statistical significance of the difference between their performances (i.e., the required number of iterations) using a significance level of 99%. In the corresponding tables, the Quantized schemes with $u = 0.0, \ldots, 1.0$, are denoted as $Q_0, \ldots, Q_{10}$, respectively; the Linear, Modular and Exponential Increasing schemes are denoted as I_L, I_M and I_E, respectively; the Sigmoid schemes with $\lambda = 10^{-1}$, 10^{-2} and 10^{-3} are denoted as S_1, S_2 and S_3, respectively; the Swarm Partitioning and the Dominance of the Best schemes are denoted as P and D, respectively; and the Self-Adaptive scheme is denoted as Se. The symbol "+" denotes that the performances of two algorithms have a statistically significant difference, while "−" stands for the lack of a statistical significant difference. All results and t-tests are reported in Tables 2–11.

In Test Problem 1, all schemes were successful, outperforming the local and global variant of the standard PSO, which correspond to the values $u = 0$ and $u = 1$ of the Quantized scheme, respectively. Most of the algorithms had

Table 3
t-tests for Test Problem 1

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	I_L	I_M	I_E	S_1	S_2	S_3	P	D	Se
Q_0		+	+	+	+	+	+	+	+	+	−	+	+	+	+	+	+	+	+	+
Q_1	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Q_2	+	+		+	+	+	+	+	+	+	+	+	−	+	+	+	+	+	−	+
Q_3	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	−
Q_4	+	+	+	+		−	−	+	+	+	+	+	+	+	+	+	−	+	+	+
Q_5	+	+	+	+	−		−	+	+	+	+	+	+	+	+	+	−	+	+	+
Q_6	+	+	+	+	−	−		−	+	+	+	+	+	+	+	+	−	+	+	−
Q_7	+	+	+	+	+	+	−		+	+	+	+	+	+	+	+	−	+	+	−
Q_8	+	+	+	+	+	+	+	+		+	+	+	−	+	+	+	+	+	−	+
Q_9	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	−	+	+
Q_{10}	−	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+
I_L	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+
I_M	+	+	−	+	+	+	+	+	−	+	+	+		+	+	+	+	+	−	+
I_E	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+
S_1	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
S_2	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+
S_3	+	+	+	+	−	−	−	−	+	+	+	+	+	+	+	+		+	+	−
P	+	+	+	+	+	+	+	+	+	−	+	+	+	+	+	+	+		+	+
D	+	+	−	+	+	+	+	+	−	+	+	+	−	+	+	+	+	+		+
Se	+	+	+	−	+	+	−	−	+	+	+	+	+	+	+	+	−	+	+	

Table 4
Results for Test Problem 2

Type	Param.	Suc.	Mean	St.D.	Min.	Max.	ENFE
QUAN	0.0	1.00	4.673×10^2	2.065×10^2	2.370×10^2	1.518×10^3	1.402×10^4
	0.1	1.00	3.014×10^2	3.580×10^2	1.350×10^2	3.358×10^3	9.041×10^3
	0.2	1.00	2.401×10^2	1.727×10^2	1.170×10^2	1.098×10^3	7.203×10^3
	0.3	1.00	2.773×10^2	3.165×10^2	1.160×10^2	2.444×10^3	8.320×10^3
	0.4	1.00	3.310×10^2	7.036×10^2	1.050×10^2	6.450×10^3	9.931×10^3
	0.5	0.97	6.290×10^2	1.786×10^3	1.000×10^2	1.000×10^4	1.049×10^4
	0.6	0.98	4.486×10^2	1.477×10^3	1.030×10^2	1.000×10^4	7.764×10^3
	0.7	0.99	3.307×10^2	1.019×10^3	9.700×10^1	1.000×10^4	7.060×10^3
	0.8	0.97	5.483×10^2	1.690×10^3	8.800×10^1	1.000×10^4	7.917×10^3
	0.9	1.00	2.926×10^2	2.821×10^2	1.170×10^2	1.978×10^3	8.778×10^3
	1.0	0.68	3.583×10^3	4.446×10^3	2.370×10^2	1.000×10^4	2.483×10^4
INC	Lin	1.00	4.144×10^2	3.117×10^2	2.400×10^2	2.309×10^3	1.243×10^4
	Mod	0.99	3.692×10^2	1.018×10^3	1.130×10^2	1.000×10^4	8.241×10^3
	Exp	1.00	3.852×10^2	1.740×10^2	2.490×10^2	1.247×10^3	1.156×10^4
SIG	10^{-1}	1.00	4.282×10^2	1.156×10^2	2.520×10^2	1.152×10^3	1.285×10^4
	10^{-2}	1.00	3.379×10^2	1.669×10^2	2.030×10^2	1.178×10^3	1.014×10^4
	10^{-3}	1.00	1.931×10^2	1.106×10^2	1.020×10^2	8.140×10^2	5.794×10^3
PAR		0.97	6.231×10^2	1.679×10^3	1.290×10^2	1.000×10^4	1.030×10^4
DOM		1.00	3.781×10^2	5.693×10^2	1.260×10^2	4.470×10^3	1.134×10^4
SELF		1.00	2.820×10^2	4.737×10^2	1.100×10^2	3.930×10^3	8.460×10^3

statistically significant differences in their performance. Only the case of $u = 1$ (pure global PSO version) in the Quantized scheme, exhibited a success rate smaller than 1.0. At a first glance, this contradicts the claim that $u = 1$ promotes exploitation, since the problem is unimodal and convex, thus, it should be solved efficiently by UPSO with an exploitation-oriented unification factor. The explanation for the decreased performance lies in the dimension of the problem. The value $u = 1$ results in an exploitation-oriented UPSO version (pure global PSO), which converges

Table 5
t-tests for Test Problem 2

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	I_L	I_M	I_E	S_1	S_2	S_3	P	D	Se
Q_0		+	+	+	−	−	−	−	−	+	+	−	−	+	−	+	+	−	−	+
Q_1	+		−	−	−	−	−	−	−	−	+	−	−	−	+	−	+	−	−	−
Q_2	+	−		−	−	−	−	−	−	−	+	+	−	+	+	+	−	−	−	−
Q_3	+	−	−		−	−	−	−	−	−	+	+	−	+	+	−	−	−	−	−
Q_4	−	−	−	−		−	−	−	−	−	+	−	−	−	−	−	−	−	−	−
Q_5	−	−	−	−	−		−	−	−	−	+	−	−	−	−	−	−	−	−	−
Q_6	−	−	−	−	−	−		−	−	−	+	−	−	−	−	−	−	−	−	−
Q_7	−	−	−	−	−	−	−		−	−	+	−	−	−	−	−	−	−	−	−
Q_8	−	−	−	−	−	−	−	−		−	+	−	−	−	−	−	−	−	−	−
Q_9	+	−	−	−	−	−	−	−	−		+	+	−	+	+	−	+	−	−	−
Q_{10}	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+
I_L	−	−	+	+	−	−	−	−	−	+	+		−	−	−	−	+	−	−	−
I_M	−	−	−	−	−	−	−	−	−	−	+	−		−	−	−	−	−	−	−
I_E	+	−	+	+	−	−	−	−	−	+	+	−	−		−	−	+	−	−	−
S_1	−	+	+	+	−	−	−	−	−	+	+	−	−	−		+	+	−	−	+
S_2	+	−	+	−	−	−	−	−	−	−	+	−	−	−	+		+	−	−	−
S_3	+	+	−	−	−	−	−	−	−	+	+	+	−	+	+	+		−	+	−
P	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−	−		−	−
D	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−	+	−		−
Se	+	−	−	−	−	−	−	−	−	−	+	−	−	−	+	−	−	−	−	

Table 6
Results for Test Problem 3

Type	Param.	Suc.	Mean	St.D.	Min.	Max.	ENFE
QUAN	0.0	0.95	9.628×10^2	2.124×10^3	1.140×10^2	1.000×10^4	1.538×10^4
	0.1	0.99	4.623×10^2	9.991×10^2	1.170×10^2	1.000×10^4	1.109×10^4
	0.2	1.00	2.813×10^2	2.155×10^2	1.100×10^2	1.893×10^3	8.440×10^3
	0.3	1.00	1.891×10^2	7.819×10^1	1.020×10^2	6.430×10^2	5.673×10^3
	0.4	1.00	1.674×10^2	5.795×10^1	9.300×10^1	5.080×10^2	5.023×10^3
	0.5	1.00	1.313×10^2	3.541×10^1	6.100×10^1	2.510×10^2	3.939×10^3
	0.6	1.00	1.912×10^2	6.436×10^2	8.000×10^1	6.553×10^3	5.737×10^3
	0.7	0.94	7.520×10^2	2.353×10^3	6.000×10^1	1.000×10^4	5.160×10^3
	0.8	0.84	1.732×10^3	3.632×10^3	7.300×10^1	1.000×10^4	5.604×10^3
	0.9	0.73	2.826×10^3	4.386×10^3	8.800×10^1	1.000×10^4	7.075×10^3
	1.0	0.52	4.895×10^3	4.930×10^3	1.150×10^2	1.000×10^4	1.049×10^4
INC	Lin	1.00	1.149×10^3	1.960×10^3	6.300×10^1	8.657×10^3	3.448×10^4
	Mod	1.00	1.408×10^2	3.386×10^1	7.800×10^1	2.800×10^2	4.226×10^3
	Exp	0.98	1.714×10^3	2.796×10^3	8.700×10^1	1.000×10^4	4.729×10^4
SIG	10^{-1}	0.96	9.170×10^2	1.954×10^3	9.400×10^1	1.000×10^4	1.683×10^4
	10^{-2}	1.00	5.716×10^2	6.808×10^2	1.060×10^2	6.710×10^3	1.715×10^4
	10^{-3}	1.00	1.675×10^2	5.828×10^1	8.200×10^1	5.510×10^2	5.026×10^3
PAR		0.98	3.657×10^2	1.386×10^3	8.400×10^1	1.000×10^4	5.177×10^3
DOM		0.94	7.353×10^2	2.353×10^3	8.300×10^1	1.000×10^4	4.595×10^3
SELF		1.00	1.273×10^2	2.896×10^1	7.900×10^1	2.270×10^2	3.820×10^3

rapidly toward the global minimizer but at a different rate per dimension. Thus, the algorithm can be trapped in suboptimal solutions, although very close to the actual global minimizer. The Quantized scheme with $u \in [0.4, 0.7]$, along with the Sigmoid scheme with $\lambda = 10^{-3}$ and the Self-Adaptive scheme exhibited the best performance with small differences among them, followed by the Dominance of the Best and the Swarm Partitioning schemes.

Table 7
t-tests for Test Problem 3

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	I_L	I_M	I_E	S_1	S_2	S_3	P	D	Se
Q_0		−	+	+	+	+	+	−	−	+	+	−	+	−	−	−	+	−	−	+
Q_1	−		−	+	+	+	−	−	+	+	+	+	+	+	−	−	+	−	−	+
Q_2	+	−		+	+	+	−	−	+	+	+	+	+	+	+	+	+	−	−	+
Q_3	+	+	+		−	+	−	−	+	+	+	+	+	+	+	+	−	−	−	+
Q_4	+	+	+	−		+	−	−	+	+	+	+	+	+	+	+	−	−	−	+
Q_5	+	+	+	+	+		−	+	+	+	+	+	−	+	+	+	+	−	−	−
Q_6	+	−	−	−	−	−		−	+	+	+	+	−	+	+	+	−	−	−	−
Q_7	−	−	−	−	−	+	−		−	+	+	−	−	+	−	−	−	−	−	+
Q_8	−	+	+	+	+	+	+	−		−	+	−	+	−	−	+	+	+	−	+
Q_9	+	+	+	+	+	+	+	+	−		+	+	+	−	+	+	+	+	+	+
Q_{10}	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+
I_L	−	+	+	+	+	+	+	−	−	+	+		+	−	−	+	+	+	−	+
I_M	+	+	+	+	+	−	−	−	+	+	+	+		+	+	+	+	−	−	+
I_E	−	+	+	+	+	+	+	+	−	−	+	−	+		−	+	+	+	+	+
S_1	−	−	+	+	+	+	+	−	−	+	+	−	+	−		−	+	−	−	+
S_2	−	−	+	+	+	+	+	−	+	+	+	+	+	+	−		+	−	−	+
S_3	+	+	+	−	−	+	−	−	+	+	+	+	+	+	+	+		−	−	+
P	−	−	−	−	−	−	−	−	+	+	+	+	−	+	−	−	−		−	−
D	−	−	−	−	−	−	−	−	−	+	+	−	−	+	−	−	−	−		−
Se	+	+	+	+	+	−	−	+	+	+	+	+	+	+	+	+	+	−	−	

Table 8
Results for Test Problem 4

Type	Param.	Suc.	Mean	St.D.	Min.	Max.	ENFE
QUAN	0.0	1.00	5.317×10^2	7.679×10^1	3.410×10^2	7.000×10^2	1.595×10^4
	0.1	1.00	2.619×10^2	3.044×10^1	1.910×10^2	3.520×10^2	7.858×10^3
	0.2	1.00	2.592×10^2	3.753×10^2	1.570×10^2	3.958×10^3	7.778×10^3
	0.3	1.00	1.957×10^2	2.266×10^1	1.490×10^2	2.680×10^2	5.872×10^3
	0.4	1.00	1.961×10^2	1.305×10^2	1.400×10^2	1.472×10^3	5.882×10^3
	0.5	1.00	1.794×10^2	1.995×10^1	1.430×10^2	2.510×10^2	5.381×10^3
	0.6	0.99	2.790×10^2	9.821×10^2	1.300×10^2	1.000×10^4	5.478×10^3
	0.7	0.99	2.844×10^2	9.816×10^2	1.390×10^2	1.000×10^4	5.643×10^3
	0.8	0.99	3.045×10^2	9.796×10^2	1.550×10^2	1.000×10^4	6.260×10^3
	0.9	0.99	3.401×10^2	9.761×10^2	1.560×10^2	1.000×10^4	7.350×10^3
	1.0	0.90	1.299×10^3	2.915×10^3	2.480×10^2	1.000×10^4	1.109×10^4
INC	Lin	1.00	3.939×10^2	4.259×10^1	3.130×10^2	6.060×10^2	1.182×10^4
	Mod	0.99	3.213×10^2	9.891×10^2	1.650×10^2	1.000×10^4	6.773×10^3
	Exp	1.00	4.142×10^2	3.795×10^1	3.270×10^2	5.260×10^2	1.243×10^4
SIG	10^{-1}	1.00	5.030×10^2	4.282×10^1	3.840×10^2	6.060×10^2	1.509×10^4
	10^{-2}	1.00	3.310×10^2	3.763×10^1	2.620×10^2	4.920×10^2	9.931×10^3
	10^{-3}	0.99	2.826×10^2	9.818×10^2	1.490×10^2	1.000×10^4	5.590×10^3
PAR		0.98	4.448×10^2	1.375×10^3	1.900×10^2	1.000×10^4	7.647×10^3
DOM		1.00	2.266×10^2	2.070×10^2	1.710×10^2	2.265×10^3	6.799×10^3
SELF		1.00	2.000×10^2	1.282×10^2	1.460×10^2	1.456×10^3	6.000×10^3

In Test Problem 2, $u = 0.2$ proved to be the best unification factor in the Quantized scheme. As expected, exploration-oriented (u near 0) variants of UPSO performed better than exploitation-oriented variants (u near 1). The Sigmoid scheme with $\lambda = 10^{-3}$ and the Self-Adaptive scheme also performed well without having statistically significant difference between their performance. The Dominance of the Best scheme had better performance compared to the Swarm Partitioning scheme.

Table 9
t-tests for Test Problem 4

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	I_L	I_M	I_E	S_1	S_2	S_3	P	D	Se
Q_0		+	+	+	+	+	−	−	−	−	+	+	−	+	+	+	−	−	+	+
Q_1	+		−	+	+	+	−	−	−	−	+	+	−	+	+	+	−	−	−	+
Q_2	+	−		−	−	−	−	−	−	−	+	+	−	+	+	−	−	−	−	−
Q_3	+	+	−		−	+	−	−	−	−	+	+	−	+	+	+	−	−	−	−
Q_4	+	+	−	−		−	−	−	−	−	+	+	−	+	+	+	−	−	−	−
Q_5	+	+	−	+	−		−	−	−	−	+	+	−	+	+	+	−	−	−	−
Q_6	−	−	−	−	−	−		−	−	−	+	−	−	−	−	−	−	−	−	−
Q_7	−	−	−	−	−	−	−		−	−	+	−	−	−	−	−	−	−	−	−
Q_8	−	−	−	−	−	−	−	−		−	+	−	−	−	−	−	−	−	−	−
Q_9	−	−	−	−	−	−	−	−	−		+	−	−	−	−	−	−	−	−	−
Q_{10}	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+
I_L	+	+	+	+	+	+	−	−	−	−	+		−	+	+	+	−	−	+	+
I_M	−	−	−	−	−	−	−	−	−	−	+	−		−	−	−	−	−	−	−
I_E	+	+	+	+	+	+	−	−	−	−	+	+	−		+	+	−	−	+	+
S_1	+	+	+	+	+	+	−	−	−	−	+	+	−	+		+	−	−	+	+
S_2	+	+	−	+	+	+	−	−	−	−	+	+	−	+	+		−	−	+	+
S_3	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−		−	−	−
P	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−	−		−	−
D	+	−	−	−	−	−	−	−	−	−	+	+	−	+	+	+	−	−		−
Se	+	+	−	−	−	−	−	−	−	−	+	+	−	+	+	+	−	−	−	

Table 10
Results for Test Problem 5

Type	Param.	Suc.	Mean	St.D.	Min.	Max.	ENFE
QUAN	0.0	0.99	8.956×10^2	1.321×10^3	4.000×10^0	1.000×10^4	2.435×10^4
	0.1	1.00	6.915×10^2	8.829×10^2	4.000×10^0	4.677×10^3	2.074×10^4
	0.2	1.00	5.472×10^2	6.195×10^2	5.000×10^0	3.516×10^3	1.642×10^4
	0.3	1.00	4.074×10^2	4.742×10^2	6.000×10^0	3.330×10^3	1.222×10^4
	0.4	0.98	5.130×10^2	1.393×10^3	7.000×10^1	1.000×10^4	9.778×10^3
	0.5	0.92	1.088×10^3	2.671×10^3	5.000×10^0	1.000×10^4	1.019×10^4
	0.6	0.94	1.021×10^3	2.496×10^3	3.000×10^0	1.000×10^4	1.428×10^4
	0.7	0.84	2.128×10^3	3.616×10^3	5.400×10^1	1.000×10^4	2.243×10^4
	0.8	0.80	2.525×10^3	3.894×10^3	6.000×10^1	1.000×10^4	2.463×10^4
	0.9	0.75	2.750×10^3	4.225×10^3	7.100×10^1	1.000×10^4	1.332×10^4
	1.0	0.76	2.674×10^3	4.162×10^3	4.400×10^1	1.000×10^4	1.421×10^4
INC	Lin	1.00	6.275×10^2	5.258×10^2	4.000×10^0	2.525×10^3	1.882×10^4
	Mod	0.98	6.492×10^2	1.586×10^3	8.000×10^0	1.000×10^4	1.403×10^4
	Exp	1.00	7.685×10^2	7.715×10^2	3.000×10^0	4.084×10^3	2.305×10^4
SIG	10^{-1}	1.00	4.525×10^2	2.581×10^2	3.000×10^0	1.171×10^3	1.357×10^4
	10^{-2}	1.00	4.696×10^2	2.628×10^2	3.000×10^0	1.549×10^3	1.409×10^4
	10^{-3}	0.99	5.581×10^2	1.444×10^3	7.900×10^1	1.000×10^4	1.402×10^4
PAR		1.00	4.102×10^2	6.133×10^2	3.000×10^0	4.939×10^3	1.231×10^4
DOM		0.98	5.322×10^2	1.441×10^3	7.000×10^1	1.000×10^4	1.038×10^4
SELF		1.00	4.560×10^2	9.025×10^2	6.800×10^1	5.987×10^3	1.368×10^4

Test Problem 3 is highly multimodal; thus we anticipated that more balanced UPSO versions would perform better. Indeed, the value $u = 0.5$ proved to be the best, with unification factors higher than 0.7 exhibiting poor performance. The Self-Adaptive scheme had the best overall performance. We must note the performance of the Modular Increasing scheme, which had one of the best performances. This is an indication that in highly multimodal functions, the iterative modification of the unification factor from 0 to 1 can provide better results than fixed values.

Table 11
t-tests for Test Problem 5

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	I_L	I_M	I_E	S_1	S_2	S_3	P	D	Se
Q_0		−	−	+	−	−	−	+	+	+	+	−	−	−	+	+	−	+	−	+
Q_1	−		−	+	−	−	−	+	+	+	+	−	−	−	−	−	−	+	−	−
Q_2	−	−		−	−	−	−	+	+	+	+	−	−	−	−	−	−	−	−	−
Q_3	+	+	−		−	−	−	+	+	+	+	+	−	+	−	−	−	−	−	−
Q_4	−	−	−	−		−	−	+	+	+	+	−	−	−	−	−	−	−	−	−
Q_5	−	−	−	−	−		−	−	+	+	+	−	−	−	−	−	−	−	−	−
Q_6	−	−	−	−	−	−		−	+	+	+	−	−	−	−	−	−	−	−	−
Q_7	+	+	+	+	+	−	−		−	−	−	+	+	+	+	+	+	+	+	+
Q_8	+	+	+	+	+	+	+	−		−	−	+	+	+	+	+	+	+	+	+
Q_9	+	+	+	+	+	+	+	−	−		−	+	+	+	+	+	+	+	+	+
Q_{10}	+	+	+	+	+	+	+	−	−	−		+	+	+	+	+	+	+	+	+
I_L	−	−	−	+	−	−	−	+	+	+	+		−	−	+	+	−	+	−	−
I_M	−	−	−	−	−	−	−	+	+	+	+	−		−	−	−	−	−	−	−
I_E	−	−	−	+	−	−	−	+	+	+	+	−	−		+	+	−	+	−	+
S_1	+	−	−	−	−	−	−	+	+	+	+	+	−	+		−	−	−	−	−
S_2	+	−	−	−	−	−	−	+	+	+	+	+	−	+	−		−	−	−	−
S_3	−	−	−	−	−	−	−	+	+	+	+	−	−	−	−	−		−	−	−
P	+	+	−	−	−	−	−	+	+	+	+	+	−	+	−	−	−		−	−
D	−	−	−	−	−	−	−	+	+	+	+	−	−	−	−	−	−	−		−
Se	+	−	−	−	−	−	−	+	+	+	+	−	−	+	−	−	−	−	−	

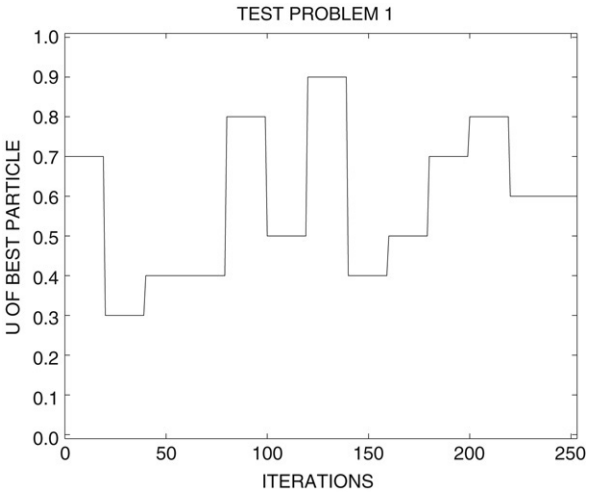

Fig. 3. Evolution of the unification factor of the best particle for the Dominance of the Best scheme, for Test Problem 1.

In Test Problem 4, $u = 0.5$ proved again to be the best, with exploration-oriented UPSO versions outperforming the exploitation-oriented ones. Self-Adaptive and Dominance of the Best also proved to be very efficient, with all algorithms having just a few statistically significant differences among their performances. Finally, in Test Problem 5, the Quantized scheme with unification factor $u = 0.3$ was the best among the Quantized schemes, with Swarm Partitioning and Self-Adaptive having the best overall performance.

The values $u = 0.2$ and $u = 0.3$ were the only cases of the Quantized scheme that had success rates equal to 1.0 for all test problems, although, in some cases, they were outperformed by more balanced UPSO versions, such as $u = 0.5$. Also, the Linearly Increasing and Self-Adaptive schemes were successful in all test problems, with the latter always outperforming both the local and global PSO variants with respect to the mean number of required iterations. The evolution of the unification factor of the best particle (i.e., the unification factor that corresponds to the best partition) for the Dominance of the Best scheme is illustrated in Figs. 3–7, for a single experiment per problem.

We also observed that the final unification factor in the adaptive schemes, i.e., the unification factor that corresponded to the solution, came rarely in line with the best unification factor observed in the Quantized scheme.

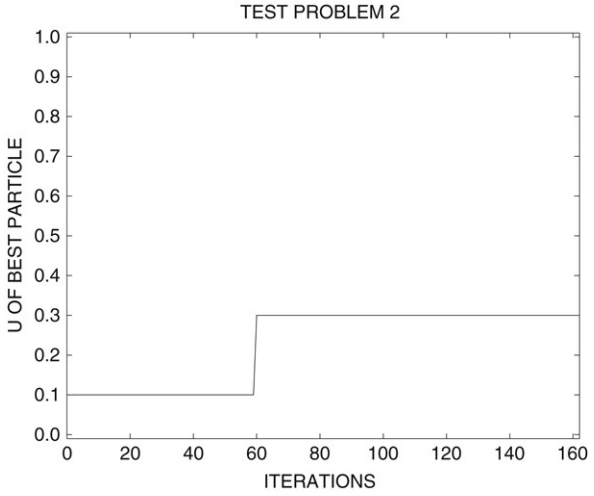

Fig. 4. Evolution of the unification factor of the best particle for the Dominance of the Best scheme, for Test Problem 2.

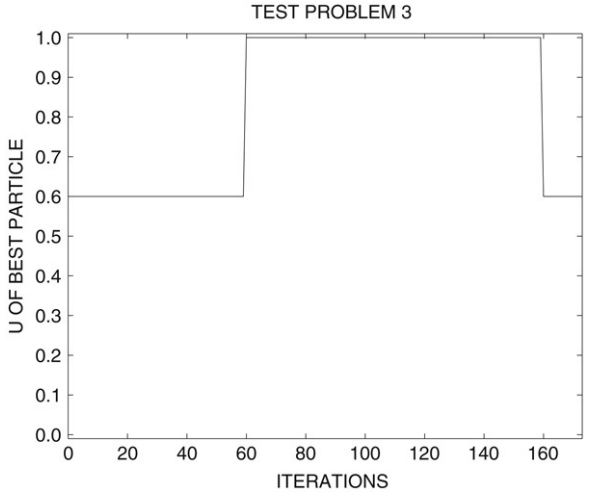

Fig. 5. Evolution of the unification factor of the best particle for the Dominance of the Best scheme, for Test Problem 3.

This is an indication that, given a problem, the adaptive schemes capture its dynamic and change their behavior during the execution of the algorithm according to its performance at each iteration. Moreover, Test Problems 3 and 5 were characterized by wide ranges of the solution's unification factor in the adaptive schemes, while, for the rest of the problems, narrow ranges near 0 were observed. As expected, the aforementioned effect was quite mild for the case of Swarm Partitioning, since the influence of each unification factor remains fixed during the iterations.

For all test problems, there was an UPSO version that outperformed the standard PSO variants. The values $u = 0.2$ and $u = 0.3$ proved to be the most effective values of the unification factor, while $u = 0.5$ constitutes a good choice for more balanced search and faster convergence, although it comes at the risk of slightly reduced success rates. The Linearly Decreasing, Sigmoid and Self-Adaptive schemes proved to be robust and reliable, with the latter exhibiting considerably better performance.

5. Conclusions

Different schemes for the determination of UPSO's unification factor were proposed and investigated on the DeJong benchmark problems suite. Extensive experiments provided information regarding the most promising values and adaptation schemes that always outperform both the global and local variant of PSO, justifying the usefulness of

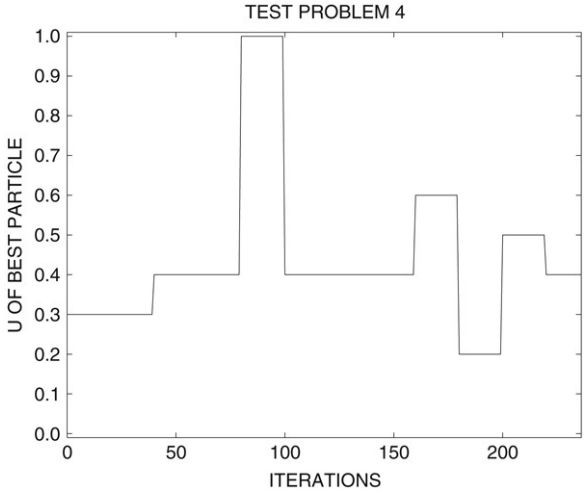

Fig. 6. Evolution of the unification factor of the best particle for the Dominance of the Best scheme, for Test Problem 4.

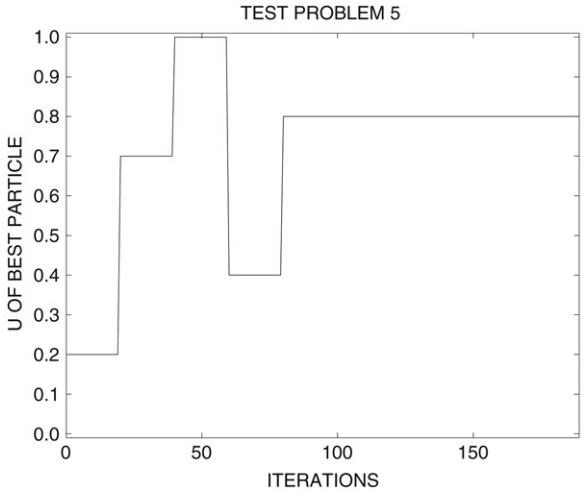

Fig. 7. Evolution of the unification factor of the best particle for the Dominance of the Best scheme, for Test Problem 5.

UPSO and providing intuition regarding the influence of the unification factor on its performance. The self-adaptive scheme exhibited very good performance, indicating the ability of UPSO to capture the dynamic of the problem at hand and determine the proper unification factor.

Summarizing the results, the following conclusions were derived:

(1) There is always an UPSO version that outperforms the standard PSO variants.
(2) If a fixed unification factor is desirable, then $u = 0.2$ and $u = 0.3$ were shown to be the most effective values, while $u = 0.5$ constitutes a good choice for more balanced search and faster convergence at the risk of a slightly reduced success rates.
(3) Adaptive schemes such as Linearly Decreasing, Sigmoid and Self-Adaptive are robust and reliable, with the latter exhibiting considerably better performance, which is an indication of UPSO's ability to capture the structure of the problem at hand and adapt the unification factor properly.

Future work will include further experiments on benchmark and real-life problems using also the UPSO schemes with mutation to fully reveal its effectiveness and efficiency.

References

[1] T. Bäck, D. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, IOP Publishing and Oxford University Press, New York, 1997.
[2] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, New York, 1995.
[3] E. Bonabeau, M. Dorigo, G. Théraulaz, From Natural to Artificial Swarm Intelligence, Oxford University Press, New York, 1999.
[4] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann, 2001.
[5] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 11–32.
[6] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings Sixth Symposium on Micro Machine and Human Science, IEEE Service Center, Piscataway, NJ, 1995, pp. 39–43.
[7] M.A. Abido, Optimal design of power system stabilizers using particle swarm optimization, IEEE Transactions on Energy Conversion 17 (3) (2002) 406–413.
[8] D.K. Agrafiotis, W. Cedeno, Feature selection for structure–activity correlation using binary particle swarms, Journal of Medicinal Chemistry 45 (5) (2002) 1098–1107.
[9] P.C. Fourie, A.A. Groenwold, The particle swarm optimization algorithm in size and shape optimization, Structural Multidisciplinary Optimization 23 (2002) 259–267.
[10] C.O. Ourique, E.C. Biscaia, J. Carlos Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers and Chemical Engineering 26 (2002) 1783–1793.
[11] K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1 (2–3) (2002) 235–306.
[12] K.E. Parsopoulos, M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 211–224.
[13] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, M.N. Vrahatis, Evolutionary computation techniques for optimizing fuzzy cognitive maps in radiation therapy systems, Lecture Notes in Computer Science 3102 (2004) 402–413.
[14] K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization in dynamic environments, Lecture Notes in Computer Science 3449 (2005) 590–599.
[15] T. Ray, K.M. Liew, A swarm metaphor for multiobjective design optimization, Engineering Optimization 34 (2) (2002) 141–153.
[16] A. Saldam, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocessors and Microsystems 26 (2002) 363–371.
[17] K.E. Parsopoulos, M.N. Vrahatis, UPSO: A unified particle swarm optimization scheme, in: Proceedings of the International Conference of Computational Methods in Sciences and Engineering, ICCMSE 2004, in: Lecture Series on Computer and Computational Sciences, vol. 1, VSP International Science Publishers, Zeist, The Netherlands, 2004, pp. 868–873.
[18] P.J. Angeline, Adaptive and self-adaptive evolutionary computations, in: M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, T. Fukuda (Eds.), Computational Intelligence: A Dynamic Systems Perspective, IEEE Press, 1995, pp. 152–163.
[19] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. IEEE Int. Conf. Neural Networks, vol. IV, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[20] M.M. Millonas, Swarms, phase transitions, and collective intelligence, in: M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, T. Fukuda (Eds.), Computational Intelligence: A Dynamic System Perspective, IEEE Press, Piscataway, NJ, 1994, pp. 137–151.
[21] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: Simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 204–210.
[22] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proc. 2000 IEEE CEC, IEEE Service Center, Piscataway, NJ, 2000, pp. 84–88.
[23] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[24] I.C. Trelea, The particle swarm optimization algorithm: Convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.
[25] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, in: V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), in: Evolutionary Programming, vol. VII, Springer, 1998, pp. 601–610.
[26] J. Matyas, Random optimization, Automatization and Remote Control 26 (1965) 244–251.