Ann Oper Res (2007) 156: 99–127
DOI 10.1007/s10479-007-0224-y

Memetic particle swarm optimization

Y.G. Petalas · K.E. Parsopoulos · M.N. Vrahatis

Published online: 9 August 2007

Abstract We propose a new Memetic Particle Swarm Optimization scheme that incorporates local search techniques in the standard Particle Swarm Optimization algorithm, resulting in an efficient and effective optimization method, which is analyzed theoretically. The proposed algorithm is applied to different unconstrained, constrained, minimax and integer programming problems and the obtained results are compared to that of the global and local variants of Particle Swarm Optimization, justifying the superiority of the memetic approach.

Keywords Global optimization · Particle swarm optimization · Memetic algorithms · Local search

1 Introduction

Particle Swarm Optimization (PSO) is a stochastic, population-based search algorithm that gained a lot of attention since its development in 1995. Its popularity can be attributed to its easy implementation and its ability to solve efficiently a plethora of problems in science and engineering, including optimal design of power systems (Abido 2002), feature selection for structure-activity correlations in medical applications (Agrafiotis and Cedeno 2002), biological applications (Cockshott and Hartman 2001), size and shape optimization (Fourie and Groenwold 2002; Ray and Liew 2002), environmental applications (Lu et al. 2002), analysis in chemical processes (Ourique et al. 2002), bioinformatics (Parsopoulos et al. 2004), task assignment problems (Saldam et al. 2002), industrial control (Papageorgiou et al. 2004) and numerical optimization (Parsopoulos and Vrahatis 2002c, 2004).

Y.G. Petalas · K.E. Parsopoulos · M.N. Vrahatis (✉)
Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras
Artificial Intelligence Research Center (UPAIRC), University of Patras, 26110 Patras, Greece
e-mail: vrahatis@math.upatras.gr

Y.G. Petalas
e-mail: petalas@math.upatras.gr

K.E. Parsopoulos
e-mail: kostasp@math.upatras.gr

The main inspiration behind PSO springs from the simulation and analysis of social dynamics and the interactions among the members of organized colonies, therefore, it is categorized as a swarm intelligence algorithm. PSO has many common key characteristics with Evolutionary Algorithms (EAs), such as Genetic Algorithms (Holland 1975), Evolution Strategies (Schwefel 1995) and Differential Evolution (Storn and Price 1997), thereby sharing many aspects of their behavior.

EAs have proved to be very useful in many applications. However, there is a well-known problem regarding their local search abilities in optimization problems (Angeline 1998). More specifically, although most EAs are capable of detecting the region of attraction of the global optimizer fast, once there, they cannot perform a refined local search to compute the optimum with high accuracy, unless specific procedures are incorporated in their operators. Some versions of PSO also exhibit this deficiency.

The aforementioned drawback of EAs triggered the development of Memetic Algorithms (MAs), which incorporate local search components. MAs constitute a class of metaheuristics that combines population-based optimization algorithms with local search procedures (Dawkins 1976; Moscato 1989, 1999). More specifically, MAs consist of a global component, which is responsible for a rough search of the search space and the detection of the most promising regions, and a local search component, which is used for probing the detected promising regions, in order to obtain solutions with high accuracy. EAs have been used as the global component in MAs with Simulated Annealing and random local search (Moscato, 1999). MAs have proved to be an unrivaled methodology in several problem domains (Moscato, 1999; Petalas and Vrahatis 2004a, 2004b).

We propose a new algorithm that combines PSO with local search methods, resulting in an efficient Memetic PSO scheme. The performance of the new scheme is investigated on different test problems, including unconstrained, constrained, minimax and integer minimization problems. The results are compared with the corresponding results of both the local and global variants of PSO. The rest of the paper is organized as follows: a description of Memetic PSO is provided in Section 2, along with descriptions of Memetic Algorithms, PSO and the proposed approach. Section 3 is devoted to the experimental results, as well as to a description of the Random Walk with Directional Exploitation local search that is employed to investigate the performance of the proposed scheme. The paper concludes in Section 4.

2 Memetic particle swarm optimization

Memetic PSO (MPSO) is a hybrid algorithm that combines PSO with local search techniques. MPSO consists of two main components, a global one that is responsible for the global search of the search space, and a local one, which performs more refined search around potential solutions of the problem at hand. In the following, the Memetic Algorithms, Particle Swarm Optimization, as well as the proposed MPSO scheme are described, along with a convergence analysis of the proposed MPSO scheme.

2.1 Memetic algorithms

MAs comprise a family of population-based, heuristic search algorithms, designed to perform global optimization. The main inspiration behind their development was Dawkins' "meme" (Dawkins 1976), which represents a unit of cultural evolution that can exhibit refinement, as well as models of adaptation in natural systems that combine evolutionary

adaptation of individuals with individual learning within a lifetime. MAs include a stage of individual optimization or learning (usually in the form of a local search) as part of their search operation.

MAs were first proposed in 1989 (Moscato 1989), where Simulated Annealing was used for local search with a competitive and cooperative game between agents, interspersed with the use of a crossover operator, to tackle the traveling salesman problem. The method gained wide acceptance, due to its ability to solve difficult problems.

Although MAs bear a similarity with Genetic Algorithms (GAs) (Goldberg 1989), they mimic rather cultural evolution than biological evolution. GAs are a way of solving problems by mimicking the same processes nature uses. To evolve a solution of a problem, GAs employ the same combination of selection, recombination and mutation that is applied to genes. In nature, genes are usually not modified during an individual's lifetime, whereas memes are. Therefore, most MAs can be interpreted as a cooperative–competitive algorithm of optimizing agents.

In general, an MA can be described through the following abstract description:

Begin
    Population Initialization
    LocalSearch
    Evaluation
    Repeat
        Recombination
        Mutation
        LocalSearch
        Evaluation
        Selection
    Until termination criterion is satisfied
    Return best solution
End

In particular, at the beginning, the population is initialized within the search space. The LocalSearch function takes an individual as input, and performs a local search. The Evaluation function plays the role of the objective function. After the initial population has been created, the recombination process takes place for selected individuals. A new individual is created by recombining the selected individuals according to the Recombination function. The Mutation function performs the mutation operation on some individuals of the population. The Selection function chooses the individuals that will survive in the next population. The termination condition can include various criteria, such as time-expiration and/or generation-expiration.

MAs have been successfully applied in combinatorial optimization, and especially for the approximate solution of NP-hard optimization problems. Their success can be attributed to the synergy of the different search approaches that are combined in the MA (Krasnogor 2002; Land 1998; Merz 1998).

The first implementations of MAs were hybrid algorithms that exploit GAs as an evolutionary algorithm and a local search at each iteration (GA–LS) (Belew et al. 1991; Hart 1994; Hinton and Nowlan 1987; Geesing and Stork 1991; Muhlenbein et al. 1988). The GA–LS hybrid scheme is interesting due to the interaction between the local and global search components of the algorithm. An important aspect of this phenomenon is the Baldwin effect (Belew 1990; Hinton and Nowlan 1987), in which learning in natural systems

speeds up the rate of evolutionary change. Similar effects have been observed by a number of authors that used GA–LS hybrids (Belew et al. 1991; Hinton and Nowlan 1987; Geesing and Stork 1991).

2.2 Particle swarm optimization

PSO is a stochastic optimization algorithm that exploits a population of individuals to synchronously probe the search space. The inspiration behind its development springs from the study of the collective behavior in decentralized systems, where populations of simple agents interact among them, as well as with their environment (Kennedy and Eberhart 2001). In PSO's context, the population is called a *swarm* and the individuals (i.e., the search agents) are called *particles*.

Each particle moves with an adaptable velocity within the search space, and retains a memory of the best position it has ever encountered. There are two main variants of PSO with respect to the information exchange scheme among the particles. In the *global* variant, the best position ever attained by all individuals of the swarm is communicated to all the particles at each iteration. In the *local* variant, each particle is assigned to a neighborhood consisting of some of the particles. In this case, the best position ever attained by the particles that comprise a neighborhood is communicated among them (Kennedy and Eberhart 2001). Neighboring particles are determined based on their indices rather than their actual distance in the search space.

Assume an n-dimensional search space, $S \subset \mathbb{R}^n$, and a swarm consisting of N particles. The ith particle is an n-dimensional vector,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in S.$$

The velocity of this particle is also an n-dimensional vector,

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top.$$

The best previous position encountered by the ith particle in S is denoted by

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in S.$$

If r is the neighborhood's radius, then the neighborhood of x_i is defined as

$$\{x_{i-r}, x_{i-r+1}, \ldots, x_i, \ldots, x_{i+r-1}, x_{i+r}\}.$$

The particles are assumed to lie on a ring topology, i.e., x_1 is the immediate neighbor of x_N. Assume g_i to be the index of the particle that attained the best previous position among all the particles in the neighborhood of x_i, and t to be the iteration counter. Then, the swarm is manipulated by the equations (Clerc and Kennedy 2002):

$$v_i^{(t+1)} = \chi\left[v_i^{(t)} + c_1 r_1\left(p_i^{(t)} - x_i^{(t)}\right) + c_2 r_2\left(p_{g_i}^{(t)} - x_i^{(t)}\right)\right], \tag{1}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}, \tag{2}$$

where $i = 1, \ldots, N$; χ is a parameter called *constriction factor*; c_1 and c_2 are two positive constants called *cognitive* and *social* parameter, respectively; and r_1, r_2, are random vectors with components uniformly distributed within $[0, 1]$. All operations between vectors are performed componentwise.

The constriction factor is a mechanism for controlling the magnitude of the velocities. It is derived analytically through the formula (Clerc and Kennedy 2002),

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

where $\varphi = c_1 + c_2$. The values received for $\varphi > 4$ and $\kappa = 1$ are considered the most common settings of χ due to their good average performance (Clerc and Kennedy 2002). Different settings of χ, as well as a thorough theoretical analysis of the derivation of (3), can be found in (Clerc and Kennedy 2002; Trelea 2003).

The performance of a population-based algorithm depends on its ability to perform global search of the search space (exploration) as well as more refined local search (exploitation). Proper balance between these two characteristics results in enhanced performance. In the global variant of PSO, all particles are attracted by the same overall best position, converging faster towards specific points. Thus, the global variant of PSO emphasizes exploitation over exploration. On the other hand, in the local variant, the information of the best position of each neighborhood is communicated slowly to the other particles of the swarm through their neighbors. Therefore, the attraction to specific best positions is weaker, hindering the swarm from getting trapped in locally optimal solutions. Thus, the local variant of PSO emphasizes exploration over exploitation. Proper selection of the neighborhood's size affects the trade-off between exploration and exploitation. The selection of the appropriate neighborhood size is an open problem. In practice, it is up to the practitioner and it is based solely on his experience.

The initialization of the swarm and the velocities, is usually performed randomly and uniformly in the search space, although more sophisticated initialization techniques can enhance the overall performance of the algorithm (Parsopoulos and Vrahatis 2002a).

2.3 The proposed algorithm

MPSO constitutes a combination of the PSO algorithm with a local search method. Thus, various different schemata can be obtained such as:

Scheme 1: Local search is applied on the overall best position, p_g, of the swarm, where g is the index of the best particle.

Scheme 2: For each best position, p_i, $i = 1, \ldots, N$, a random number, r, is generated, and, if $r < \varepsilon$, where $\varepsilon > 0$ is a prescribed threshold, then local search is applied on p_i.

Scheme 3a: Local search is applied both on the best position, p_g, of the swarm, as well as on some randomly selected best positions, p_i, $i \in \{1, \ldots, N\}$.

Scheme 3b: Local search is applied both on the best position, p_g, of the swarm, as well as on some randomly selected best positions, p_i, $i \in \{1, \ldots, N\}$, for which, $\|p_g - p_i\| > c\Delta(S)$, where $c \in (0, 1)$ and $\Delta(S)$ is the diameter of the search space S.

The above three schemata can be applied either in every iteration of the algorithm or at some iterations. Of course, many other related schemata can be considered such as those that apply local search to all particles. However, as we have noticed in experiments, the latter schemata are costly in terms of function evaluations, and, in practice, only a small percentage of the particles (say 5%) has to be considered for applying local search. The same conclusions were derived also by Hart (1994), based on investigation on GA–LS hybrid schemes.

A pseudocode for the MPSO algorithm is given below.

Input: N , χ, c_1, c_2, $x_{\min}$, $x_{\max}$ (lower & upper bounds), F (objective function).
Set $t = 0$.
Initialize $x_i^{(t)}$, $v_i^{(t)} \in [x_{\min}, x_{\max}]$, $p_i^{(t)} \leftarrow x_i^{(t)}$, $i = 1, \ldots, N$.
Evaluate $F(x_i^{(t)})$.
Determine the indices g_i, $i = 1, \ldots, N$.
While (stopping criterion is not satisfied) **Do**
  Update the velocities $v_i^{(t+1)}$, $i = 1, \ldots, N$, according to (1).
  Set $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$, $i = 1, \ldots, N$.
  Constrain each particle x_i in $[x_{\min}, x_{\max}]$.
  Evaluate $F(x_i^{(t+1)})$, $i = 1, \ldots, N$.
  If $F(x_i^{(t+1)}) < F(p_i^{(t)})$ **Then** $p_i^{(t+1)} \leftarrow x_i^{(t+1)}$.
  Else $p_i^{(t+1)} \leftarrow p_i^{(t)}$.
  Update the indices g_i.
  When (local search is applied) **Do**
    Choose (according to one of the Schemata 1–3) $p_q^{(t+1)}$, $q \in \{1, \ldots, N\}$.
    Apply local search on $p_q^{(t+1)}$ and obtain a new solution, y.
    If $F(y) < F(p_q^{(t+1)})$ **Then** $p_q^{(t+1)} \leftarrow y$.
  End When
  Set $t = t + 1$.
End While

2.4 A convergence analysis of memetic PSO

A proof of convergence in probability can be given for the MPSO scheme, assuming that a stochastic local search method is applied on the best particle of the swarm at each iteration of the algorithm. The proof follows the analysis of Matyas (1965) for stochastic optimization algorithms. Assume that $F : S \to \mathbb{R}$ is a unimodal objective function, x_{opt} is its unique minimizer in S, and $F_{\text{opt}} = F(x_{\text{opt}})$. Let also g be the index of the best particle of the swarm in the kth iteration, i.e., $p_g^{(k)}$ is the best solution seen by the algorithm since its start up to iteration k. The level set of F at a constant value, K, is defined as $G[K] = \{x : F(x) < K\}$. We assume that $G[K] \neq \emptyset$, for all $K > F_{\text{opt}}$. Let $f(z)$ be the probability distribution of the points generated by the stochastic local search. The proof holds for any probability distribution with $f(z) \neq 0$, for all z. We define as a *successful step* of MPSO at iteration k, the fact that

$$F\left(p_g^{(k+1)}\right) < F\left(p_g^{(k)}\right) - \varepsilon,$$

for a prescribed $\varepsilon > 0$. The probability of a successful step from $p_g^{(k)}$ is given by

$$P_F(x) = \int_{G[F(p_g)-\varepsilon]} f(z - p_g)dz.$$

Then, based on the analysis of Matyas (1965), the following theorem is straightforwardly proved:

Theorem 1 *Let $F(x)$ have a unique minimum in S, $G[K] \neq \emptyset$, for all $K > F_{\text{opt}}$, and $f(z) \neq 0$ for all z. Then, the sequence of best positions, $\{p_g^{(k)}\}$, of the swarm in MPSO tends in probability to x_{opt}.*

Proof Let $\delta(x) = \{z\colon \varrho(z, x) < \delta\}$, $\delta > 0$, be the δ-neighborhood of a point x, where $\varrho(z, x)$ denotes the distance between the points x, z. We will prove that for any $\delta > 0$ it holds that

$$\lim_{k\to\infty} P\left\{\varrho\left(p_g^{(k)}, x_{\text{opt}}\right) > \delta\right\} = \lim_{k\to\infty} P\left\{p_g^{(k)} \notin \delta(x_{\text{opt}})\right\} = 0,$$

i.e., the probability that the distance $\varrho(p_g^{(k)}, x_{\text{opt}}) > \delta$, or equivalently that $p_g^{(k)} \notin \delta(x_{\text{opt}})$, tends to zero. If we denote by F_δ the minimum value of F on the boundary of $\delta(x_{\text{opt}})$, we shall have $F_\delta > F_{\text{opt}}$. We can now define $\varepsilon = \varepsilon(\delta)$ such that $0 < \varepsilon(\delta) < F_\delta - F_{\text{opt}}$. For all previous best positions, $p_g \notin \delta(x_{\text{opt}})$, of the particle under consideration, the inequality $F(p_g) - \varepsilon > F_{\text{opt}}$, is valid. Furthermore, from the assumptions of the theorem, $G[F(p_g) - \varepsilon]$ is a non-empty region. Since $f(z) > 0$ for all z, there will exist an $\alpha > 0$, such that $P_F(p_g) \geq \alpha$, i.e., the probability of a successful step from p_g is positive (although in some cases it may become very small).

Let $F(p_g^{(1)})$ be the function value of the best position, $p_g^{(1)}$, in the first iteration of the algorithm. We denote,

$$\tau = \frac{(F(p_g^{(1)}) - F_\delta)}{\varepsilon},$$

and $m = \lfloor \tau \rfloor$, i.e., m is the largest integer less than τ. From the design of the PSO and MPSO algorithm, if even $m + 1$ steps turn out to be successful, then all the subsequent points of the sequence $\{p_g^{(k)}\}$ lie in $\delta(x_{\text{opt}})$. Consequently, the probability $P\{p_g^{(k)} \notin \delta(x_{\text{opt}})\}$ is less than or equal to the probability that the number of successful steps does not exceed m, i.e.,

$$P\left\{p_g^{(k)} \notin \delta(x_{\text{opt}})\right\} \leq P\left\{\sum_{i=1}^{k} y^{(i)} \leq m\right\},$$

where, $y^{(i)} = 1$, if there was a successful step in iteration i, and $y^{(i)} = 0$, otherwise. The latter probability increases with a decrease in the probability of successful steps, and since $P_F(p_g) \geq \alpha$, it obeys the well-known Newton's theorem (on the binomial probability distribution),

$$P\left\{\sum_{i=1}^{k} y^{(i)} \leq m\right\} \leq \sum_{i=0}^{m} \binom{k}{i} \alpha^i (1-\alpha)^{k-i},$$

where k is the number of steps (iterations) taken. Further, when $k > 2m$ and $\alpha < 0.5$,

$$\begin{aligned}\sum_{i=0}^{m} \binom{k}{i} \alpha^i (1-\alpha)^{k-i} &< (m+1)\binom{k}{m}(1-\alpha)^k \\ &= \frac{m+1}{m!} k(k-1)(k-2)\cdots(k-m+1)(1-\alpha)^k \\ &< \frac{m+1}{m!} k^m (1-\alpha)^k.\end{aligned}$$

Consequently, $P\{\varrho(p_g^{(k)}, x_{\text{opt}}) > \delta\} < \frac{m+1}{m!} k^m (1-\alpha)^k$. Thus, for $\alpha > 0$, it is clear that

$$\lim_{k\to\infty} k^m (1-\alpha)^k = 0,$$

and the theorem is proved. □

We must note that alternative local search schemes may require modifications in the proof in order to remain valid.

3 Experimental analysis

This is our first attempt to experimentally show that there always exists an MPSO scheme that outperforms the standard PSO method. To achieve this, we have considered a large number of benchmark problems from various categories and we have experimentally configured the parameters used by the proposed approach. In particular, MPSO was tested on 29 well-known and widely used unconstrained, constrained, minimax and integer optimization benchmark problems. For all test problems, the PSO parameters were set to their default values, $\chi = 0.729$, $c_1 = c_2 = 2.05$ (Clerc and Kennedy 2002). The remaining parameters, such as the number of iterations and the step length of the local search method used, were problem dependent and, thus, individually specified for each test problem. For the local search component of MPSO, the Random Walk with Direction Exploitation, which is described in the following, was employed. The derived scheme is denoted as RWMPSO.

3.1 Random walk with direction exploitation

Random Walk with Direction Exploitation (RWDE) is an iterative, stochastic optimization method that generates a sequence of approximations of the optimizer by assuming a random vector as a search direction. RWDE can be applied in discontinuous and non-differentiable functions, and it has been proved effective in cases where other methods fail due to difficulties posed by the form of the objective function, e.g., sharply varying functions and shallow regions (Rao 1992).

Let $x^{(t)}$ be the approximation of the minimizer at the tth iteration. Then, the new value (approximation), $x^{(t+1)}$, at the $(t+1)$th iteration, is computed through the equation,

$$x^{(t+1)} = x^{(t)} + \lambda z^{(t)},$$

where λ is a prescribed scalar step-length, and $z^{(t)}$ is a unit-length random vector. The workings of RWDE are summarized in the following steps:

Step 1. Initialize the iteration number, $t = 0$. Start with an initial point, $x^{(1)}$, and a scalar step length, $\lambda = \lambda_{\text{init}}$. Compute the function value, $F^{(1)} = F(x^{(1)})$, where F is the objective function.

Step 2. Set $t = t + 1$ and check whether t is greater than a threshold, $t_{\max}$, and if so terminate; otherwise generate a unit-length random vector, $z^{(t)}$, and continue.

Step 3. Compute the value of the objective function,

$$F' = F\left(x^{(t)} + \lambda z^{(t)}\right).$$

Step 4. Compare the values F' and $F^{(t)}$. If $F' < F^{(t)}$, then set $x^{(t+1)} = x^{(t)} + \lambda z^{(t)}$; set $t = t + 1$, $\lambda = \lambda_{\text{init}}$, $F^{(t)} = F'$, and check whether t is greater than $t_{\max}$ and if so terminate; otherwise go to Step 3. If $F' > F^{(t)}$, set $x^{(t+1)} = x^{(t)}$, reduce the scalar step length $\lambda = \lambda/2$, and repeat Steps 2–4. If $F' = F^{(t)}$, set $x^{(t+1)} = x^{(t)}$ and repeat Steps 2–4.

Instead of RWDE, we could use different, more sophisticated stochastic local searches (Hoos and Stützle 2004). The reason for choosing RWDE was to show that our approach can be efficient and effective without a thorough investigation of the efficiency of the stochastic local search used. Additionally, the selection of RWDE as the local search component of MPSO was based on its simplicity and its relative efficiency. RWDE does not make any continuity or differentiability assumptions on the objective function, thus, it is consistent with the PSO framework that requires function values solely. Therefore, it was preferred over gradient-based local search algorithms. Moreover, it is easily implemented and it can be modified with minor effort to suit different problems.

3.2 Unconstrained optimization problems

The benchmark problems that were used are:

Test Problem 1 (Trelea 2003) (Sphere). This problem is defined by

$$F_1(x) = \sum_{i=1}^{n} x_i^2, \tag{4}$$

where n is the dimension of the problem. The global minimizer is $x^* = (0, \ldots, 0)^\top$ with $F_1(x^*) = 0$.

Test Problem 2 (Trelea 2003) (Generalized Rosenbrock). This problem is defined by

$$F_2(x) = \sum_{i=1}^{n-1} \bigl(100\bigl(x_{i+1} - x_i^2\bigr)^2 + (x_i - 1)^2\bigr), \tag{5}$$

where n is the dimension of the problem. The global minimizer is $x^* = (1, \ldots, 1)^\top$ with $F_2(x^*) = 0$.

Test Problem 3 (Trelea 2003) (Rastrigin). This problem is defined by

$$F_3(x) = 10n + \sum_{i=1}^{n} \bigl(x_i^2 - 10\cos(2\pi x_i)\bigr), \tag{6}$$

where n is the dimension of the problem. The global minimizer is $x^* = (0, \ldots, 0)^\top$ with $F_3(x^*) = 0$.

Test Problem 4 (Trelea 2003) (Griewank). This problem is defined by

$$F_4(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{7}$$

where n is the dimension of the problem. The global minimizer is $x^* = (0, \ldots, 0)^\top$ with $F_4(x^*) = 0$.

Test Problem 5 (Trelea 2003) (Schaffer's f6). This problem is defined by

$$F_5(x) = 0.5 - \frac{\left(\sin\left(\sqrt{x_1^2 + x_2^2}\right)\right)^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}. \tag{8}$$

The global minimizer is $x^* = (0, 0)^\top$ with $F_5(x^*) = 0$.

Test Problem 6 (Storn and Price 1997) (Ackley). This problem is defined by

$$F_6(x) = -20\exp\left(-0.02\sqrt{\frac{\sum_{j=1}^n x_j^2}{n}}\right) - \exp\left(\frac{\sum_{j=1}^n \cos(2\pi x_j)}{n}\right) + 20 + \exp(1), \tag{9}$$

where n is the dimension of the problem. The global minimizer is $x^* = (0, \ldots, 0)^\top$ with $F_6(x^*) = 0$.

Test Problem 7 (Storn and Price 1997) (Corana). This problem is defined by

$$F_7(x) = \sum_{j=1}^4 \begin{cases} 0.15\left(z_j - 0.05\operatorname{sign}(z_j)\right)^2 d_j, & \text{if } |x_j - z_j| < 0.05, \\ d_j x_j^2, & \text{otherwise,} \end{cases} \tag{10}$$

where $x_j \in [-1000, 1000]$, $(d_1, d_2, d_3, d_4) = (1, 1000, 10, 100)$, and

$$z_j = \left\lfloor \left| \frac{x_j}{0.2} \right| + 0.49999 \right\rfloor \operatorname{sign}(x_j) 0.2.$$

All points with $|x_j^*| < 0.05$, $j = 1, 2, 3, 4$, are global minimizers with $F_7(x^*) = 0$.

Test Problem 8 (Lee and Yao 2004). This problem is defined by

$$\begin{aligned} F_8(x) = 0.1\Bigg\{ &\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] \\ &+ (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n)\right] \Bigg\} + \sum_{i=1}^n u(x_i, 5, 100, 4), \end{aligned} \tag{11}$$

where n is the dimension of the problem and

$$u(z, a, k, m) = \begin{cases} k(z - a)^m, & z > a, \\ 0, & -a \le z \le a, \\ k(-z - a)^m, & z < -a. \end{cases} \tag{12}$$

The global minimizer is $x^* = (1, \ldots, 1)^\top$ with $F_8(x^*) = 0$.

Test Problem 9 (Lee and Yao 2004). This problem is defined by

$$\begin{aligned} F_9(x) = \frac{\pi}{n}\Bigg\{ &10\sin^2(\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + 10\sin^2(\pi x_{i+1})\right] + (x_n - 1)^2 \Bigg\} \\ &+ \sum_{i=1}^n u(x_i, 10, 100, 4), \end{aligned} \tag{13}$$

Table 1 Parameters for the unconstrained optimization problems

Problem	Dimension	Range	Error Goal
TP1	30	$(-100, 100)^{30}$	10^{-2}
TP2	30	$(-30, 30)^{30}$	10^{2}
TP3	30	$(-5.12, 5.12)^{30}$	10^{2}
TP4	30	$(-600, 600)^{30}$	10^{-1}
TP5	2	$(-100, 100)^{2}$	10^{-5}
TP6	30	$(-32, 32)^{30}$	10^{-3}
TP7	4	$(-1000, 1000)^{4}$	10^{-6}
TP8	30	$(-50, 50)^{30}$	10^{-6}
TP9	30	$(-50, 50)^{30}$	10^{-2}

Table 2 Parameter setting of RWMPSOg for the unconstrained problems

Problem	*SS*	Iter	Step	Best	Prob	Freq
TP1	15	5	1.0	yes	–	1
	30	5	1.0	yes	–	1
	60	5	1.0	yes	–	1
TP2	15	10	1.0	yes	–	1
	30	5	1.0	yes	–	1
	60	5	1.0	yes	–	1
TP3	15	5	1.0	–	0.2	1
	30	5	1.0	–	0.2	1
	60	5	1.0	–	0.1	1
TP4	15	5	4.0	yes	–	1
	30	5	4.0	yes	–	1
	60	5	4.0	yes	–	1
TP5	15	8	1.0	–	0.3	1
	30	8	1.0	–	0.2	1
	60	8	1.0	–	0.1	1
TP6	15	5	1.0	–	0.5	1
	30	5	1.0	–	0.5	1
	60	5	1.0	–	0.4	1
TP7	15	5	1.0	yes	–	20
	30	5	1.0	yes	–	20
	60	5	1.0	yes	–	20
TP8	15	5	1.0	–	0.8	1
	30	5	1.0	–	0.5	1
	60	5	1.0	–	0.3	1
TP9	15	5	1.0	–	0.6	1
	30	5	1.0	–	0.5	1
	60	5	1.0	–	0.3	1

where n is the dimension of the problem and u is defined by (12). The global minimizer is $x^* = (1, \ldots, 1)^\top$ with $F_9(x^*) = 0$.

The dimension of each test problem, the range in which the particles were constrained, as well as the error goal are reported in Table 1 (Lee and Yao 2004; Storn and Price 1997; Trelea 2003). The maximum number of iterations for every problem was equal to 10^4. For all problems, three different swarm sizes (denoted as *SS*) were considered, namely 15, 30, and 60, following the setup of Trelea (2003). The global and the local PSO variants (denoted as PSOg and PSOl, respectively), were equipped with RWDE, resulting in the global and local RWMPSO variants (denoted as RWMPSOg and RWMPSOl, respectively), and applied on all test problems. For each test problem, 50 independent experiments were conducted. An experiment was considered successful if the desired error goal was achieved within the maximum number of iterations.

The configuration of RWDE was problem dependent. The parameter settings of RWMPSOg and RWMPSOl for the unconstrained problems are reported in Tables 2 and 3, respectively. The first column of the tables denotes the problem, while second column stands for the swarm size. The third and fourth column report the number of iterations and initial step size used by RWDE, respectively. The fifth column has the value "yes" in the cases where RWDE was applied only on the best particle of the swarm. On the other hand, if RWDE was applied on the best position of each particle with a probability, then this probability is reported in column six. Finally, the last column shows the frequency of application of the local search. Thus, the value "1" corresponds to application of the local search at every iteration, while "20" corresponds to application every 20 iterations.

The results of RWMPSOg and RWMPSOl for the unconstrained problems are reported in Tables 4 and 5, respectively, while the corresponding results of the standard PSO are reported in Tables 6 and 7, respectively. More specifically, the number of successes (out of 50 experiments), the minimum, mean, maximum, and standard deviation of the required function evaluations (evaluated only on the successful experiments) are reported. In order to take full advantage of its exploration properties, small neighborhood sizes were selected for the local PSO. Thus, for Test Problems 1–6 and Test Problem 8, the neighborhood radius was equal to 1, while, in Test Problems 7 and 9, a neighborhood radius equal to 2 was used due to its superior performance.

Comparing the global variants, it is clear that RWMPSO improves significantly the performance of PSO. In all problems, the number of successes of RWMPSO is equal or higher than PSO. Even in cases where PSO had no successes (TP6 with swarm size 15) RWMPSO succeeded in 42 out of 50 experiments. Naturally, in some cases, this comes at the cost of some extra function evaluations, although in most cases the required number of function evaluations of RWMPSO is smaller than PSO. The influence of larger swarm sizes in RWMPSO seems to be similar to that in the standard PSO, with larger swarms requiring more function evaluations but having better success rates.

Similar conclusions can be derived also for the local variants of RWMPSO and PSO. The PSOl variant is significantly better with respect to its success rate than its global variant, and this holds also for RWMPSO. RWMPSOl performed in almost all cases better than PSOl, achieving high success rates and requiring (in most cases) significantly smaller number of function evaluations. RWMPSOl had better success rates than RWMPSOg in all test problems, although it was slower. This is also an indication that the neighborhood radius has the same effect on RWMPSO as in the standard PSO.

Interestingly, RWMPSOg outperformed in many cases even PSOl, which is a promising indication that the use of local search in PSOg can enhance significantly its exploration

Table 3 Parameter setting of RWMPSOl for the unconstrained problems

Problem	*SS*	Iter	Step	Best	Prob	Freq
TP1	15	10	1.0	yes	–	1
	30	10	1.0	yes	–	1
	60	5	1.0	yes	–	1
TP2	15	8	0.5	yes	–	50
	30	5	1.0	yes	–	30
	60	5	1.0	yes	–	1
TP3	15	5	1.0	yes	–	20
	30	10	1.0	yes	–	20
	60	5	1.0	yes	–	1
TP4	15	10	8.0	yes	–	1
	30	10	8.0	yes	–	1
	60	10	8.0	yes	–	1
TP5	15	8	1.0	–	0.3	2
	30	8	1.0	–	0.1	1
	60	8	1.0	–	0.1	2
TP6	15	5	1.0	yes	–	20
	30	5	1.0	yes	–	20
	60	5	1.0	yes	–	20
TP7	15	5	1.0	yes	–	20
	30	5	1.0	yes	–	20
	60	5	1.0	yes	–	20
TP8	15	5	1.0	yes	0.3	1
	30	5	1.0	yes	–	2
	60	5	1.0	yes	–	2
TP9	15	3	1.0	–	0.5	1
	30	5	1.0	–	0.1	1
	60	10	1.0	yes	–	2

capabilities. For some problems, where the superiority of an algorithm over another was not clear, t-tests were performed to verify the statistical significance of the results. The hypothesis testing was conducted using the null hypothesis that the mean number of required function evaluations between the two algorithms is equal, at a statistical significance level of 99%. Thus, in Test Problem 1, RWMPSOg was compared with PSOl (they both achieved 50 successes), rejecting the null hypothesis, i.e., RWMPSOg was better than PSOl, for swarm size equal to 15. The same holds also for the RWMPSOl against PSOl, and for the RWMPSOg against RWMPSOl (this seems a natural conclusion if we consider the simplicity and unimodality of Test Problem 1). RWMPSOg was proved to be statistically better than PSOl also for Test Problem 7. Overall, RWMPSO outperformed PSO, improving significantly its performance in most test problems.

Table 4 Results of RWMPSOg for the unconstrained problems

Problem	*SS*	Min	Mean	Max	StD	Suc
TP1	15	5324	6009.7	6713	342.9	50
	30	7507	8615.2	9616	492.0	50
	60	11757	13604.8	15153	660.0	50
TP2	15	3233	9275.5	62928	11272.3	50
	30	4944	10534.2	95361	13234.9	50
	60	6915	15515.3	54052	20835.7	50
TP3	15	4042	14121.2	43767	9229.6	33
	30	6669	18234.5	50466	10670.7	46
	60	6812	19494.8	80063	15016.4	50
TP4	15	5115	5956.5	6602	344.5	50
	30	7062	7954.6	9188	446.5	50
	60	10488	12142.2	13876	786.7	50
TP5	15	4166	17962.7	70755	17727.2	50
	30	265	12425.4	63487	9827.5	50
	60	4826	11019.3	37069	6057.7	50
TP6	15	33726	42746.1	69092	7086.1	42
	30	50051	58797.6	69792	4651.9	48
	60	69607	79473.6	95133	5080.0	50
TP7	15	1560	2094.3	3428	458.0	50
	30	2110	3412.2	6635	825.8	50
	60	4569	5781.5	8238	629.4	50
TP8	15	57145	74845.6	110259	10848.2	47
	30	63590	78764.5	93710	6701.0	49
	60	77465	89876.7	106207	7017.4	49
TP9	15	12882	20300.4	41456	5209.8	49
	30	22377	34710.7	50848	5818.2	50
	60	34611	54114.9	144452	19734.5	50

3.3 Constrained optimization problems

The benchmark problems that were used are:

Test Problem 10 (Himmelblau 1972). This problem is defined by

$$F_{10}(x) = (x_1 - 2)^2 + (x_2 - 1)^2, \tag{14}$$

subject to

$$x_1 = 2x_2 - 1, \qquad \frac{x_1^2}{4} + x_2^2 - 1 \leq 0,$$

with $x_i \in [-100, 100]$, $i = 1, 2$.

Table 5 Results of RWMPSOl for the unconstrained problems

Problem	*SS*	Min	Mean	Max	StD	Suc
TP1	15	6526	7318.3	8746	458.6	50
	30	10021	11129.9	12834	658.5	50
	60	16116	18208.7	20323	947.2	50
TP2	15	3814	7679.0	24288	3846.9	50
	30	7879	12220.5	20581	2554.3	50
	60	10458	20687.0	37905	5662.9	50
TP3	15	2213	8999.1	139085	19172.8	49
	30	4050	13815.3	155611	21787.9	50
	60	4622	16489.6	75139	12050.7	50
TP4	15	5370	6588.2	7698	465.4	50
	30	8379	9881.4	11880	726.2	50
	60	14520	16858.4	21845	1417.6	50
TP5	15	2628	21386.1	86842	21344.6	50
	30	174	18300.2	113044	17947.4	50
	60	271	18080.2	56900	12403.9	50
TP6	15	9844	12978.2	27297	2487.5	50
	30	19879	24600.1	30381	2378.5	50
	60	36395	48110.9	59773	5710.3	50
TP7	15	1922	2685.8	3290	314.6	50
	30	3738	4710.6	5759	562.9	50
	60	7230	8765.1	10001	739.1	50
TP8	15	27995	38248.6	60191	6308.8	49
	30	32514	38916.5	44470	2958.3	50
	60	62821	74476.9	83195	5460.9	50
TP9	15	12116	17579.2	28755	3315.0	50
	30	11422	19030.9	56181	8005.8	50
	60	8951	20441.7	74149	13287.7	50

Test Problem 11 (Floudas and Pardalos 1987). This problem is defined by

$$F_{11}(x) = (x_1 - 10)^3 + (x_2 - 20)^3, \tag{15}$$

subject to

$$\begin{aligned} &100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0, \\ &(x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0, \\ &13 \leq x_1 \leq 100, \quad 0 \leq x_2 \leq 100. \end{aligned}$$

Table 6 Results of PSOg for the unconstrained problems

Problem	*SS*	Min	Mean	Max	StD	Suc
TP1	15	6585	10824.8	24060	3408.4	43
	30	9060	11242.3	17850	1508.1	47
	60	14280	16360.0	19140	933.3	48
TP2	15	4830	14898.3	84735	15417.8	36
	30	6450	12469.7	35100	5877.1	29
	60	8880	26420.0	156900	27337.6	39
	15	2370	3282.3	6105	1109.8	11
TP3	30	3270	5097.3	8250	1276.8	22
	60	5340	10042.5	22860	3103.7	40
TP4	15	6030	8785.9	18885	2567.2	29
	30	8280	9718.1	11910	914.8	47
	60	12240	14891.0	18360	1176.4	49
TP5	15	1230	7727.4	59250	11973.9	31
	30	2100	18210.0	271350	44027.4	37
	60	3120	11877.3	66600	12168.4	44
TP6	15	–	–	–	–	0
	30	15090	16395.0	17700	1305.0	2
	60	21840	25116.0	30300	2171.2	20
TP7	15	1365	1896.7	2685	244.8	47
	30	2370	3272.4	4350	336.7	49
	60	4620	5616.0	6720	401.1	50
TP8	15	17325	29893.8	41955	7954.4	13
	30	19110	24277.9	35370	4173.8	19
	60	26100	30172.7	33840	1770.7	33
TP9	15	2070	8546.0	48555	10426.9	30
	30	4740	17416.6	179490	29615.7	38
	60	6300	39581.1	427680	91879.1	38

Test Problem 12 (Hock and Schittkowski 1981). This problem is defined by

$$F_{12}(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \tag{16}$$

subject to

$$\begin{aligned} &-127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0,\\ &-282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0,\\ &-196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0,\\ &4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0,\\ &-10 \le x_i \le 10, \quad i = 1, \ldots, 7. \end{aligned}$$

Table 7 Results of PSOI for the unconstrained problems

Problem	*SS*	Min	Mean	Max	StD	Suc
TP1	15	6195	8467.5	10335	907.7	50
	30	13200	16716.0	20250	1573.7	50
	60	24780	34054.8	40920	3769.3	50
TP2	15	3795	8004.9	38940	6441.3	50
	30	7440	14337.6	38100	6673.5	50
	60	14040	26443.2	76740	10755.0	50
TP3	15	2820	10759.0	81195	12830.5	45
	30	3720	16848.0	141600	22935.6	50
	60	10020	24465.6	58560	9765.0	50
TP4	15	6105	8006.4	10890	1042.3	50
	30	11850	16132.8	20940	2203.4	50
	60	23520	31506.0	39600	3796.8	50
TP5	15	45	27170.0	117975	28745.1	45
	30	90	28363.2	201450	38115.6	50
	60	180	27240.0	135960	27689.4	50
TP6	15	9420	12733.5	39990	4123.7	50
	30	17670	24231.6	45810	3791.3	50
	60	39120	47902.8	60420	4924.5	50
TP7	15	2175	2686.2	3540	312.1	50
	30	3180	4657.8	5550	520.9	50
	60	7020	8769.6	10800	714.3	50
TP8	15	15840	19353.8	27060	2027.4	43
	30	30300	36608.6	42300	2478.8	49
	60	64080	74986.8	120360	7833.0	50
TP9	15	4080	14011.7	58575	13167.1	45
	30	9570	21287.1	97830	19328.5	49
	60	17280	25918.8	59520	6634.4	50

Test Problem 13 (Hock and Schittkowski 1981). This problem is defined by

$$F_{13}(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141, \tag{17}$$

subject to

$$0 \le 85.334407 + 0.0056858T_1 + T_2x_1x_4 - 0.0022053x_3x_5 \le 92,$$
$$90 \le 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \le 110,$$
$$20 \le 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \le 25,$$
$$78 \le x_1 \le 102, \qquad 33 \le x_2 \le 45, \qquad 27 \le x_i \le 45, \quad i = 3, 4, 5,$$

where $T_1 = x_2x_5$ and $T_2 = 0.0006262$.

Test Problem 14 (Hock and Schittkowski 1981). This problem is defined exactly as Test Problem 13, but with

$$T_1 = x_2 x_3, \qquad T_2 = 0.00026.$$

Test Problem 15 (Michalewicz 1996). This problem is defined by

$$F_{15}(x) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10x_6 - 0.5\sum_{i=1}^{5} x_i^2, \tag{18}$$

subject to

$$\begin{aligned} &6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 - 6.5 \le 0,\\ &10x_1 + 10x_3 + x_6 \le 20,\\ &0 \le x_i \le 1, \quad i = 1, \ldots, 5, \quad 0 \le x_6 \le 50. \end{aligned}$$

For these test problems, the non-stationary penalty function employed in (Parsopoulos and Vrahatis 2002b) was adopted. More specifically, the penalty function is defined as (Yang et al. 1997),

$$f(x) = F(x) + h(t)H(x), \tag{19}$$

where $F(x)$ is the original objective function of the constrained problem; $h(t)$ is a dynamically modified penalty value, where t is the algorithm's current iteration number; and $H(x)$ is a penalty factor defined as

$$H(x) = \sum_{i=1}^{m} \theta\big(q_i(x)\big) q_i(x)^{\gamma(q_i(x))}, \tag{20}$$

where $q_i(x) = \max\{0, g_i(x)\}$, $i = 1, \ldots, m$, and $g_i(x)$ are the problem's constraints (assuming they are in the form $g_i(x) \le 0$). The function $q_i(x)$ is a relative violated function of the constraints; $\theta(q_i(x))$ is a multi-stage assignment function (Homaifar et al. 1994); and $\gamma(q_i(x))$ is the power of the penalty function.

The functions $h(\cdot)$, $\theta(\cdot)$ and $\gamma(\cdot)$, are problem dependent. We used the same values that are reported in (Yang et al. 1997), i.e., the relative violated function of the constraints was,

$$\gamma(q_i(x)) = \begin{cases} 1, & \text{if } q_i(x) < 1,\\ 2, & \text{otherwise,} \end{cases}$$

the assignment function was

$$\theta(q_i(x)) = \begin{cases} 10, & \text{if } q_i(x) < 0.001,\\ 20, & \text{if } 0.001 \le q_i(x) < 0.1,\\ 100, & \text{if } 0.1 \le q_i(x) < 1,\\ 300, & \text{otherwise,} \end{cases}$$

and

$$h(t) = \begin{cases} \sqrt{t}, & \text{for Test Problem 10,}\\ t\sqrt{t}, & \text{otherwise.} \end{cases}$$

Table 8 Parameter setting for the constrained optimization problems

Problem	Algorithm	Iter	Step
TP10	RWMPSOg	5	2×10^0
	RWMPSOl	5	2×10^0
TP11	RWMPSOg	5	10^{-4}
	RWMPSOl	5	10^{-3}
TP12	RWMPSOg	10	10^{-1}
	RWMPSOl	10	10^{-1}
TP13	RWMPSOg	8	2×10^{-5}
	RWMPSOl	4	10^{-8}
TP14	RWMPSOg	5	10^{-8}
	RWMPSOl	5	10^{-8}
TP15	RWMPSOg	4	5×10^{-5}
	RWMPSOl	8	2×10^{-5}

A constraint of the form $g_i(x) \leq 0$, was assumed to be violated only if $g_i(x) > 10^{-5}$. In all test problems, a swarm of size 100 was used. For each test problem, each algorithm was executed until it reached 10^5 function evaluations. Then, the best feasible detected solution was reported. For each test problem, 30 independent experiments were performed. An experiment was considered to be successful only if a feasible solution was detected.

Here, we must point out a difference between PSO's implementation in constrained and unconstrained problems. Penalty functions may assign quite low function values to unfeasible solutions. These solutions can be stored as particles' best positions, thereby attracting the swarm towards them. In order to prevent the swarm from being attracted to unfeasible regions, the indices g_i of the best particles in PSO were selected after looking at the current positions of the particles, instead of their best positions. This approach was adopted in Parsopoulos and Vrahatis (2002b) with promising results. The alternative approach of accepting only feasible solutions as particles' best positions is not valid unless there is a mechanism that can ensure that each particle will take at least one feasible position during the experiment, otherwise no best position for some or all particles can be determined. Even in the case that such a mechanism exists, the best positions' change rate is usually very slow, thereby leading to search stagnation. Thus, it is not recommended.

In Table 8, the values of the parameters used by RWMPSOg and RWMPSOl, respectively, are given. All results for the constrained problems are reported in Table 9. More specifically, the number of successes in 30 experiments, the mean and the standard deviation of the function value of the obtained feasible solutions (for the successful experiments only) are reported. For the RWMPSO variants, RWDE was applied at each iteration of the algorithm on the best detected feasible solution, if any, otherwise, each particle of the current swarm was selected for local search with probability 0.1.

With the exception of Test Problem 13, where the performance of all algorithms was equal, in all test problems RWMPSO exhibited a better performance than PSO. More specifically, RWMPSOg had a better success rate than PSOg in Test Problems 10 and 11, while the quality of its solutions was superior in all problems. The same holds for RWMPSOl, which had a better performance than PSOl in all test problems. Moreover, the standard deviations of the RWMPSO variants were always smaller than the corresponding standard deviations of the PSO variants, indicating its robustness.

Table 9 Results for the constrained optimization problems

Problem		RWMPSOg	RWMPSOl	PSOg	PSOl
TP10	Suc	30/30	30/30	24/30	22/30
	Mean	−6961.283	−6960.717	−6960.668	−6939.627
	StD	0.380	1.798	1.043	58.789
TP11	Suc	25/30	30/30	24/30	30/30
	Mean	1.832	1.427	2.042	1.454
	StD	0.474	0.061	0.865	0.078
TP12	Suc	30/30	30/30	30/30	30/30
	Mean	680.915	680.784	681.254	680.825
	StD	0.178	0.062	0.245	0.077
TP13	Suc	30/30	30/30	30/30	30/30
	Mean	−30665.550	−30665.550	−30665.550	−30665.550
	StD	0.000	0.000	0.000	0.000
TP14	Suc	30/30	30/30	30/30	30/30
	Mean	−31021.173	−31026.435	−31021.140	−31026.440
	StD	11.506	0.000	12.617	0.000
TP15	Suc	30/30	30/30	30/30	30/30
	Mean	−212.616	−213.047	−211.833	−212.933
	StD	1.043	0.002	1.840	0.365

3.4 Minimax problems

The general form of the minimax problem is (Xu 2001):

$$\min_x F(x), \tag{21}$$

where

$$F(x) = \max_{i=1,\ldots,m} f_i(x), \tag{22}$$

with $f_i(x) : S \subset \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$. Also, nonlinear programming problems of the form:

$$\begin{aligned} &\min_x F(x), \\ &g_i(x) \geq 0, \quad i = 2, \ldots, m, \end{aligned}$$

can be transformed and solved as minimax problems,

$$\min_x \max_{1 \leq i \leq m} f_i(x), \tag{23}$$

where

$$\begin{aligned} &f_1(x) = F(x), \\ &f_i(x) = F(x) - \alpha_i g_i(x), \\ &\alpha_i > 0, \end{aligned} \tag{24}$$

for $2 \leq i \leq m$. It can be proved that for large values of α_i, the optimal points of the two problems coincide (Bandler and Charalambous 1974). The benchmark problems that were used in our experiments are:

Test Problem 16 (Xu 2001). This problem is defined by

$$\begin{aligned} &\min_x F_{16}(x), \\ &F_{16}(x) = \max\{f_i(x)\}, \quad i = 1, 2, 3, \\ &\quad f_1(x) = x_1^2 + x_2^4, \\ &\quad f_2(x) = (2 - x_1)^2 + (2 - x_2)^2, \\ &\quad f_3(x) = 2\exp(-x_1 + x_2), \end{aligned} \tag{25}$$

and the desired error goal is $F_{16}(x^*) = 1.9523$.

Test Problem 17 (Xu 2001). This is a nonlinear programming problem that can be transformed to a minimax problem according to (23) and (24), and it is defined by

$$\begin{aligned} F_{17}(x) &= x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4, \\ g_2(x) &= -x_1^2 - x_2^2 - x_3^3 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8, \\ g_3(x) &= -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10, \\ g_4(x) &= -x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 + 5. \end{aligned} \tag{26}$$

The desired error goal is $F_{17}(x^*) = -40.10$.

Test Problem 18 (Xu 2001). This is a nonlinear programming problem that can be transformed to a minimax problem according to (23) and (24), and it is defined by

$$\begin{aligned} F_{18}(x) &= (x_1 - 10)^2 + 5(x_2 - 12)^2 + 3(x_4 - 11)^2 + x_3^4 \\ &\quad + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \\ g_2(x) &= -2x_1^2 - 3x_3^4 - x_3 - 4x_4^2 - 5x_5 + 127, \\ g_3(x) &= -7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 + 282, \\ g_4(x) &= -23x_1 - x_2^2 - 6x_6^2 + 8x_7 + 196, \\ g_5(x) &= -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7. \end{aligned} \tag{27}$$

The desired error goal is $F_{18}(x^*) = 247$.

Test Problem 19 (Schwefel 1995). This problem is defined by

$$\min_x F_{19}(x),$$

subject to

$$\begin{aligned} F_{19}(x) &= \max\{f_i(x)\},\ i = 1, 2, \\ f_1(x) &= |x_1 + 2x_2 - 7|, \\ f_2(x) &= |2x_1 + x_2 - 5|. \end{aligned} \tag{28}$$

The desired error goal is $F_{19}(x^*) = 10^{-6}$.

Test Problem 20 (Schwefel 1995). This problem is defined by

$$\begin{aligned} &\min_x F_{20}(x), \\ &F_{20}(x) = \max\{f_i(x)\}, \\ &f_i(x) = |x_i|, \quad i = 1, \ldots, 10, \end{aligned} \tag{29}$$

and the desired error goal is $F_{20}(x^*) = 10^{-6}$.

Test Problem 21 (Lukšan and Vlček 2000). This problem is defined by

$$\begin{aligned} &\min_x F_{21}(x), \\ &F_{21}(x) = \max\{f_i(x)\}, \\ &f_1(x) = \left(x_1 - \sqrt{x_1^2 + x_2^2}\cos\sqrt{x_1^2 + x_2^2}\right)^2 + 0.005(x_1^2 + x_2^2), \\ &f_2(x) = \left(x_2 - \sqrt{x_1^2 + x_2^2}\sin\sqrt{x_1^2 + x_2^2}\right)^2 + 0.005(x_1^2 + x_2^2), \end{aligned} \tag{30}$$

and the desired error goal is $F_{21}(x^*) = 10^{-6}$.

Test Problem 22 (Lukšan and Vlček 2000). This problem is defined by

$$\begin{aligned} &\min_x F_{22}(x), \\ &F_{22}(x) = \max\left\{|f_i(x)|\right\}, \quad i = 1, \ldots, 21, \\ &f_i(x) = x_1 \exp(x_3 t_i) + x_2 \exp(x_4 t_i) - \frac{1}{1 + t_i}, \\ &t_i = -0.5 + \frac{i - 1}{20}, \end{aligned} \tag{31}$$

and the desired error goal is $F_{22}(x^*) = 0.1$.

For each test problem, 50 experiments were performed with a swarm size equal to 20, and the particles were constrained in the range $[-50, 50]^n$, where n is the dimension of the problem. An experiment was considered successful if the desired error goal was achieved within the maximum number of function evaluations.

In Table 10, the values of the parameters used by MPSO in minimax problems are reported. These parameters are same with those described in the corresponding tables for the unconstrained problems. All results are reported in Table 11. More specifically, the number

Table 10 Parameter setting for the minimax problems

Problem	Algorithm	Iter	Step	Best	Prob	Freq
TP16	RWMPSOg	8	0.01	yes	–	1
	RWMPSOl	8	0.01	yes	–	1
TP17	RWMPSOg	5	0.5	yes	–	50
	RWMPSOl	8	0.5	yes	–	1
TP18	RWMPSOg	3	1.0	yes	–	2
	RWMPSOl	5	0.5	yes	–	2
TP19	RWMPSOg	5	1.0	yes	–	20
	RWMPSOl	3	0.5	yes	–	20
TP20	RWMPSOg	3	1.0	yes	–	2
	RWMPSOl	10	1.0	yes	–	2
TP21	RWMPSOg	8	0.01	yes	–	1
	RWMPSOl	8	0.01	yes	–	1
TP22	RWMPSOg	5	0.5	–	0.3	1
	RWMPSOl	5	0.5	–	0.3	1

of successes is reported along with the maximum, minimum, mean and standard deviation of the required number of function evaluations for the successful cases. In all cases, the RWMPSO variants outperformed the corresponding standard PSO variants, having also higher success rate in Test Problem 22, were PSOg succeeded only in 36 experiments. RWMPSOg outperformed all algorithms, exhibiting the best performance. This was also verified through t-tests. Also, the worst behavior (maximum number of function evaluations) of the RWMPSO variants was far lower than the corresponding PSO variants in most cases.

3.5 Integer programming problems

This is a very interesting class of test problems, since most evolutionary algorithms that work by rounding the real variables to integers suffer from search stagnation. The problems that were used are:

Test Problem 23 (Rüdolph 1994). This problem is defined by

$$F_{23}(x) = \|x\|_1 = |x_1| + \cdots + |x_n|, \tag{32}$$

where n is the dimension, and $x \in [-100, 100]^n$. The global minimum is $F_{23}(x^*) = 0$.

Test Problem 24 (Rüdolph 1994). This problem is defined by

$$F_{24}(x) = x^\top x = (x_1 \quad \ldots \quad x_n) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \tag{33}$$

where n is the dimension, and $x \in [-100, 100]^n$. The global minimum is $F_{24}(x^*) = 0$.

Table 11 Results for the minimax problems

Algorithm	Problem	Min	Mean	Max	StD	Suc
RWMPSOg	TP16	722	2415.3	6893	1244.2	50
	TP17	1665	3991.3	16421	2545.2	50
	TP18	4290	7021.3	10332	1241.4	50
	TP19	2430	2947.8	3541	257.0	50
	TP20	15760	18520.1	19958	776.9	50
	TP21	135	1308.8	2359	505.5	50
	TP22	2028	4404.0	24704	3308.9	50
RWMPSOl	TP16	665	2686.9	5874	1320.7	50
	TP17	2771	5948.4	10827	1902.8	50
	TP18	7813	11165.0	16590	2145.1	50
	TP19	2799	3463.7	3968	295.8	50
	TP20	28400	32167.4	36200	1775.2	50
	TP21	104	1599.7	3938	797.7	50
	TP22	1963	4593.6	11042	1680.5	50
PSOg	TP16	1540	4347.2	15720	3643.0	50
	TP17	1960	4050.4	12260	1932.8	50
	TP18	4650	7098.0	14750	1966.1	50
	TP19	2600	3018.4	3540	209.4	50
	TP20	16400	18465.0	20600	932.4	50
	TP21	80	1658.0	2340	321.6	50
	TP22	620	5976.6	87960	15572.9	36
PSOl	TP16	1560	3669.2	13860	2526.4	50
	TP17	2480	6820.8	27820	4831.0	50
	TP18	7650	11289.0	16700	1990.2	50
	TP19	2760	3475.2	4360	387.9	50
	TP20	30500	33687.0	36950	1641.7	50
	TP21	100	2572.4	4460	931.0	50
	TP22	1020	7530.0	101600	18817.0	50

Test Problem 25 (Glankwahmdee et al. 1979). This problem is defined by

$$F_{25}(x) = -(15 \quad 27 \quad 36 \quad 18 \quad 12)\,x + x^{\top}\begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix} x, \tag{34}$$

where $x \in [-100, 100]^n$. The global minimum is $F_{25}(x^*) = -737$.

Test Problem 26 (Glankwahmdee et al. 1979). This problem is defined by

$$F_{26}(x) = \left(9x_1^2 + 2x_2^2 - 11\right)^2 + \left(3x_1 + 4x_2^2 - 7\right)^2. \tag{35}$$

The global minimum is $F_{26}(x^*) = 0$.

Test Problem 27 (Glankwahmdee et al. 1979). This problem is defined by

$$F_{27}(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4. \tag{36}$$

The global minimum is $F_{27}(x^*) = 0$.

Test Problem 28 (Glankwahmdee et al. 1979). This problem is defined by

$$F_{28}(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2. \tag{37}$$

The global minimum is $F_{28}(x^*) = -6$.

Test Problem 29 (Glankwahmdee et al. 1979). This problem is defined by

$$\begin{aligned} F_{29}(x) = {} & -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 \\ & + 203.64x_2^2 + 182.25x_1x_2. \end{aligned} \tag{38}$$

The global minimum is $F_{29}(x^*) = -3833.12$.

For each test problem, 50 independent experiments were conducted with the particles constrained in $[-100, 100]^n$. The swarm size was problem dependent and equal to 100, 10, 70, 20, 20, 10, and 20, for Test Problems 23–29, respectively. An experiment was considered

Table 12 Parameter setting for the integer programming problems

Problem	Algorithm	Iter	Step	Best	Prob	Freq
TP23	RWMPSOg	8	2.0	yes	0.1	1
	RWMPSOl	3	2.0	yes	–	1
TP24	RWMPSOg	3	2.0	yes	–	1
	RWMPSOl	3	2.0	yes	–	1
TP25	RWMPSOg	3	2.0	yes	0.1	1
	RWMPSOl	3	2.0	yes	–	1
TP26	RWMPSOg	8	4.0	yes	–	1
	RWMPSOl	5	2.0	yes	–	1
TP27	RWMPSOg	3	2.0	yes	–	1
	RWMPSOl	3	2.0	yes	–	1
TP28	RWMPSOg	5	4.0	yes	–	1
	RWMPSOl	5	4.0	yes	–	1
TP29	RWMPSOg	5	2.0	yes	–	1
	RWMPSOl	5	2.0	yes	–	1

Table 13 Results for the integer programming problems

Algorithm	Problem	Min	Mean	Max	StD	Suc
RWMPSOg	TP23	17160	27176.3	74699	8656.9	50
	TP24	252	578.5	912	136.5	50
	TP25	1361	6490.6	41593	6912.8	50
	TP26	76	215.0	468	97.9	50
	TP27	687	1521.8	2439	360.7	50
	TP28	40	110.9	238	48.6	50
	TP29	72	242.7	620	132.2	50
RWMPSOl	TP23	24870	30923.9	35265	2405.0	50
	TP24	369	773.9	1931	285.5	50
	TP25	5003	9292.6	15833	2443.7	50
	TP26	73	218.7	620	115.3	50
	TP27	675	2102.9	3863	689.5	50
	TP28	40	112.0	235	48.7	50
	TP29	70	248.9	573	134.4	50
PSOg	TP23	14000	29435.3	261100	42039.1	34
	TP24	400	606.4	1000	119.0	50
	TP25	2150	12681.0	187000	35066.8	50
	TP26	100	369.6	620	113.2	50
	TP27	680	1499.0	3440	513.1	43
	TP28	80	204.8	350	62.0	50
	TP29	100	421.2	660	130.4	50
PSOl	TP23	27400	31252.0	35800	1817.8	50
	TP24	450	830.2	1470	206.0	50
	TP25	4650	11320.0	22650	3802.8	50
	TP26	120	390.0	920	134.6	50
	TP27	800	2472.4	3880	637.5	50
	TP28	70	256.0	520	107.5	50
	TP29	100	466.0	820	165.0	50

successful if the global minimum was obtained with an accuracy of 10^{-6}. In order to avoid search stagnation and possible deterioration of the algorithms' dynamics, the search points were rounded to the nearest integer only for function evaluation purposes, while they were considered as real numbers for all other operations. The best solution was also rounded after the termination of the algorithm.

In Table 12, the parameter setting of RWMPSO is reported. All results are reported in Table 13. More specifically, the number of successes is reported along with the minimum, mean, maximum, and standard deviation of the required number of function evaluations for the successful cases. Once again, RWMPSO was superior to standard PSO. RWMPSOg exhibited better performance than all the other algorithms with respect to the required mean number of function evaluations. Its standard deviations were also the smallest among the algorithms in the majority of the test problems, thereby verifying its robustness. The statistical significance of the RWMPSOg results was also verified through t-tests. Interestingly,

no algorithm suffered search stagnation. This finding was also noticed for the standard PSO in (Laskari et al. 2002).

4 Conclusions

A new Memetic Particle Swarm Optimization scheme that incorporates local search techniques to the standard Particle Swarm Optimization algorithm was proposed. Its performance was investigated on a plethora of test problems, including unconstrained, constrained, minimax and integer programming problems, employing the Random Walk with Direction Exploitation. Both the local and global variants of the proposed scheme were tested and compared with the corresponding variants of Particle Swarm Optimization. In almost all problems the memetic approach proved to be superior, increasing both the efficiency and the effectiveness of the algorithm.

Techniques for the self-adaptation of the local search parameters that could further enhance the algorithm's performance and different local search techniques will be included in future correspondence.

Acknowledgements We would like to thank the editors and the anonymous reviewers for their valuable comments and suggestions. We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS, for funding the above work.

References

Abido, M. A. (2002). Optimal design of power system stabilizers using particle swarm optimization. *IEEE Transactions on Energy Conversion*, *17*, 406–413.

Agrafiotis, D. K., & Cedeno, W. (2002). Feature selection for structure-activity correlation using binary particle swarms. *Journal of Medicinal Chemistry*, *45*, 1098–1107.

Angeline, P. J. (1998). Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In V. W. Porto, N. Saravanan, D. Waagen & A. E. Eiben (Eds.), *Evolutionary programming* (Vol. VII, pp. 601–610). Berlin: Springer.

Bandler, J. W., & Charalambous, C. (1974). Nonlinear programming using minimax techniques. *Journal of Optimization Theory and Applications*, *13*, 607–619.

Belew, R. K. (1990). Evolution, learning and culture: computational metaphores for adaptive algorithms. *Complex Systems*, *4*, 11–49.

Belew, R. K., McInerny, J., & Schraudolph, N. N. (1991). Evolving networks: using the genetic algorithm with connectionist learning. In C. Langton, C. Taylor, J. Farmer & S. Rasmussen (Eds.), *Proceedings of the second conference in artificial life* (pp. 511–548). Reading: Addison-Wesley.

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

Cockshott, A. R., & Hartman, B. E. (2001). Improving the fermentation medium for *Echinocandin B* production. Part II: particle swarm optimization. *Process Biochemistry*, *36*, 661–669.

Dawkins, R. (1976). *The selfish gene*. New York: Oxford University Press.

Floudas, C. A., & Pardalos, P. M. (1987). A collection of test problems for constrained global optimization algorithms. In P. M. Floudas (Ed.), *Lecture notes in computer science,* Vol. 455. Berlin: Springer.

Fourie, P. C., & Groenwold, A. A. (2002). The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, *23*, 259–267.

Geesing, R., & Stork, D. (1991). Evolution and learning in neural networks: the number and distribution of learning trials affect the rate of evolution. In R. Lippmann, J. Moody & D. and Touretzky (Eds.), *NIPS 3* (pp. 804–810). San Mateo: Morgan Kaufmann.

Glankwahmdee, A., Liebman, J. S., & Hogg, G. L. (1979). Unconstrained discrete nonlinear programming. *Engineering Optimization*, *4*, 95–107.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading: Addison-Wesley.

Hart, W. E. (1994). *Adaptive global optimization with local search*. Ph.D. thesis, University of California, San Diego, USA.

Himmelblau, D. M. (1972). *Applied nonlinear programming*. New York: McGraw-Hill.

Hinton, G. E., & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, *1*, 495–502.

Hock, W., & Schittkowski, K. (1981). Test examples for nonlinear programming codes. In *Lecture notes in economics and mathematical systems* (Vol. 187). Berlin: Springer.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: Ann Arbor University Press.

Homaifar, A., Lai, A. H. -Y., & Qi, X. (1994). Constrained optimization via genetic algorithms. *Simulation*, *2*, 242–254.

Hoos, H. H., & Stützle, T. (2004). *Stochastic local search: foundations and applications*. San Mateo: Morgan Kaufmann.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. San Mateo: Morgan Kaufmann.

Krasnogor, N. (2002). *Studies on the theory and design space of memetic algorithms*. Ph.D. thesis, University of the West of England, Bristol, UK.

Land, M. W. S. (1998). *Evolutionary algorithms with local search for combinatorial optimization*. Ph.D. thesis, University of California, San Diego, USA.

Laskari, E. C., Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization for integer programming. In *Proceedings of the IEEE 2002 congress on evolutionary computation* (pp. 1576–1581). Hawaii (HI), USA. New York: IEEE Press.

Lee, C. -Y., & Yao, X. (2004). Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, *8*, 1–13.

Lu, W. Z., Fan, H. Y., Leung, A. Y. T., & Wong, J. C. K. (2002). Analysis of pollutant levels in central Hong Kong applying neural network method with particle swarm optimization. *Environmental Monitoring and Assessment*, *79*, 217–230.

Lukšan, L., & Vlček, J. (2000). *Test problems for nonsmooth unconstrained and linearly constrained optimization*. Technical report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic.

Matyas, J. (1965). Random optimization. *Automatization and Remote Control*, *26*, 244–251.

Merz, P. (1998). *Memetic algorithms for combinatorial optimization. Fitness landscapes and effective search strategies*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs*. Berlin: Springer.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts. Towards memetic algorithms. Technical report C3P, Report 826, Caltech Concurrent Computation Program, California, USA

Moscato, P. (1999). Memetic algorithms. A short introduction. In D. Corne, M. Dorigo & F. and Glover (Eds.), *New ideas in optimization* (pp. 219–235). London: McGraw-Hill.

Muhlenbein, M., Gorges Schleiter, M., & Kramer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, *7*, 65–85.

Ourique, C. O., Biscaia, E. C., & Carlos Pinto, J. (2002). The use of particle swarm optimization for dynamical analysis in chemical processes. *Computers and Chemical Engineering*, *26*, 1783–1793.

Papageorgiou, E. I., Parsopoulos, K. E., Groumpos, P. P., & Vrahatis, M. N. (2004). Fuzzy cognitive maps learning through swarm intelligence. In: *Lecture notes in computer science* (Vol. 3070, pp. 344–349). Berlin: Springer.

Parsopoulos, K. E., & Vrahatis, M. N. (2002a). Initializing the particle swarm optimizer using the nonlinear simplex method. In A. Grmela & N. Mastorakis (eds.) *Advances in intelligent systems, fuzzy systems, evolutionary computation* (pp. 216–221). WSEAS Press.

Parsopoulos, K. E., & Vrahatis, M. N. (2002b). Particle swarm optimization method for constrained optimization problems. In P. Sincak, J. Vascak, V. Kvasnicka & J. and Pospichal (Eds.), *Intelligent technologies–theory and application (New trends in intelligent technologies). Frontiers in artificial intelligence and applications* (Vol. 76, pp. 214–220). Amsterdam: IOS Press.

Parsopoulos, K. E., & Vrahatis, M. N. (2002c). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, *1*, 235–306.

Parsopoulos, K. E., & Vrahatis, M. N. (2004). On the Computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*, 211–224.

Parsopoulos, K. E., Papageorgiou, E. I., Groumpos, P. P., & Vrahatis, M. N. (2004). Evolutionary computation techniques for optimizing fuzzy cognitive maps in radiation therapy systems. In *Lecture notes in computer science* (Vol. 3102, pp. 402–413). Berlin: Springer.

Petalas, Y. G., & Vrahatis, M. N. (2004a). Memetic algorithms for neural network training in bioinformatics. In *European symposium on intelligent technologies, hybrid systems and their implementation on smart adaptive systems (EUNITE 2004)* (pp. 41–46). Aachen, Germany.

Petalas, Y. G., & Vrahatis, M. N. (2004b). Memetic algorithms for neural network training on medical data. In *Fourth European symposium on biomedical engineering*, Patras, Greece.

Rao, S. S. (1992). *Optimization: theory and applications*. New Dehli: Wiley Eastern.

Ray, T., & Liew, K. M. (2002). A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, *34*(2), 141–153.

Rüdolph, G. (1994). An evolutionary algorithm for integer programming. In Y. Davidor, H.-P. Schwefel & R. Männer (Eds.), *Parallel problem solving from nature* (Vol. 3, pp. 139–148). Berlin: Springer.

Saldam, A., Ahmad, I., & Al-Madani, S. (2002). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, *26*, 363–371.

Schwefel, H. -P. (1995). *Evolution and optimum seeking*. New York: Wiley.

Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.

Trelea, I. C. (2003). The particle swarm optimization algorithm. Convergence analysis and parameter selection. *Information Processing Letters*, *85*, 317–325.

Xu, S. (2001). Smoothing method for minimax problems. *Computational Optimization and Applications*, *20*, 267–279.

Yang, J.-M., Chen, Y.-P., Horng, J.-T., & Kao, C.-Y. (1997). Applying family competition to evolution strategies for constrained optimization. In *Lecture Notes in Mathematics* (Vol. 1213, pp. 201–211). New York: Springer.