Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 179 (2021) 194–212
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

GEPSO: A new generalized particle swarm optimization algorithm

Davoud Sedighizadeh[a,*], Ellips Masehian[b], Mostafa Sedighizadeh[c], Hossein Akbaripour[d]

[a] *Department of Industrial Engineering, College of Technical and Engineering, Saveh Branch, Islamic Azad University, (IAU), Saveh, Iran*
[b] *Industrial and Manufacturing Engineering Department, California State Polytechnic University at Pomona, CA, USA*
[c] *Faculty of Electrical Engineering, Shahid Beheshti University, Tehran, Iran*
[d] *Industrial Engineering Department, Sharif University of Technology, Tehran, Iran*

Received 9 October 2018; received in revised form 29 July 2020; accepted 6 August 2020
Available online 20 August 2020

Abstract

Particle Swarm Optimization (PSO) algorithm is a nature-inspired meta-heuristic that has been utilized as a powerful optimization tool in a wide range of applications since its inception in 1995. Due to the flexibility of its parameters and concepts, PSO has appeared in many variants, probably more than any other meta-heuristic algorithm. This paper introduces the Generalized Particle Swarm Optimization (GEPSO) algorithm as a new version of the PSO algorithm for continuous space optimization, which enriches the original PSO by incorporating two new terms into the velocity updating equation. These terms aim to deepen the interrelations of particles and their knowledge sharing, increase variety in the swarm, and provide a better search in unexplored areas of the search space. Moreover, a novel procedure is utilized for dynamic updating of the particles' inertia weights, which controls the convergence of the swarm towards a solution. Also, since parameters of heuristic and meta-heuristic algorithms have a significant influence on their performance, a comprehensive guideline for parameter tuning of the GEPSO is developed. The computational results of solving numerous well-known benchmark functions by the GEPSO, original PSO, Repulsive PSO (REPSO), PSO with Passive Congregation (PSOPC), Negative PSO (NPSO), Deterministic PSO (DPSO), and Line Search-Based Derivative-Free PSO (LS-DF-PSO) approaches showed that the GEPSO outperformed the compared methods in terms of mean and standard deviation of fitness function values and runtimes.

Keywords: Heuristic algorithms; Particle Swarm Optimization (PSO); Parameter tuning; Swarm intelligence

1. Introduction

1.1. Motivation and incitement

With the advancement of engineering and social applications, we witness a surge in the emergence of various multi-dimensional problems with incomplete or noisy data, ultra-large size, and high computational complexity, which are expected to be solved effectively and efficiently by current computing technologies [26]. However, conventional exact and global optimization algorithms are not capable of solving such challenging real-world problems due to their somewhat inflexible structure and high computational burden [40]. On the other hand, the

* Corresponding author.
E-mail address: d.sedighizadeh@iau-saveh.ac.ir (D. Sedighizadeh).

https://doi.org/10.1016/j.matcom.2020.08.013

Nomenclature

Sets

N_{iter}	Set of the function evaluations
N_p	Set of the particles
N_k	Set of the coordinates

Variables

$prtpos^i{}_j$	Position of the particle j in the function evaluation i
$prtvel^i{}_j$	Velocity of the particle j in the function evaluation i
$pbest^i{}_j$	Best position of the particle j in the function evaluation i
$gbest^i$	Best position within the swarm until function evaluation i
$pbest^i{}_{\text{rand}}$	Best position of a randomly chosen particle within the swarm until function evaluation i
$f(gbest^i)$	Best fitness function value in the function evaluation i in the whole swarm
$pworst^i{}_j$	Worst position of the particle j in the function evaluation i
$gworst^i$	Worst position within the swarm until function evaluation i
μ	Mean
σ	Standard deviation
Δf	Associated normalized distance in the image space
Δx	Normalized Euclidean distance between the minimum position found by the algorithm and the analytical minimum position
Δt	Combination of Δx and Δf used for the overall assessment

Parameters

${r_1}^i$, ${r_2}^i$, ${r_3}^i$, ${r_4}^i$	Random numbers between 0 and 1 for the function evaluation i
$w^i{}_1$	Inertia weight in the GEPSO
w_2, w_3, w_4, w_5	Control weights
c_1, c_2	Cognitive and collective acceleration factors
c_3, c_4	Acceleration factors
α_1, α_2, α_3	Convergence factors
χ	Constriction factor
$V_{\min}$	Minimum velocity of the particles
$V_{\max}$	Maximum velocity of the particles
$prtvel_{\text{rand}}$	Random velocity vector between $V_{\min}$ and $V_{\max}$
$w_{\max}$	Maximum inertia weight
$w_{\min}$	Minimum inertia weight
$iter_{\max}$	Maximum number of function evaluations
γ_k	Modification factor for coordinate k

Abbreviations

ACO	Ant colony optimization
ANN	Artificial neural network
CTB-PSO	Continuous trait-based PSO
DEA	Differential evolution algorithm

bunch of heuristic methods under the paradigm of Natural Computing (NC) has been proved a suitable replacement for traditional methods in solving such problems [27]. The essence of these methods lies in simple elements which,

DPSO	Deterministic PSO
EA	Evolutionary algorithm
ELPSO	Enhanced leader PSO
EM	Electromagnetism-like mechanism
EP	Evolutionary programming
FF	Fitness function
FLPRFNS	Functional-link based Petri recurrent fuzzy neural system
FODPSO	Fractional-order Darwinian PSO
GA	Genetic algorithm
GEPSO	General particle swarm optimization
GP	Genetic programming
IPSO	Immune PSO
LS-DF-PSO	Line search-based derivative-free PSO
NC	Natural computing
NPSO	Negative PSO
PID	Proportional–integral–derivative
PSO	Particle swarm optimization
PSOPC	PSO with passive congregation
SPSO	Species-based PSO
RHDPSO	Rotary hybrid discrete PSO
REPSO	Repulsive PSO

when working together, can solve complicated problems powerfully. In general, NC methods can be divided into three categories [27]: (1) *Epigenesis*; which is related to situations where a complicated structure is desired and it is necessary to perform tentative learning, such as ANN in which the human brain is simulated as a complicated system, (2) *Phylogeny*; which is related to EAs where competition between agents on survival of the fittest occurs, as in GA [21], EP [2], GP [9], and DEA [43], and (3) *Ontogeny*; which is related to algorithms where a special organism is adopted in its environment and its elements exhibit cooperative behavior. PSO is an example of this type [8]. NC algorithms can be easily developed and customized for various applications, as they do not need previous knowledge about the problem space. However, their main drawback is that they do not guarantee to find the optimal solution and have non-deterministic nature in presenting and approximating solutions. Among the abovementioned methods, the PSO algorithm has a simple but efficient nature and is able to accommodate novel concepts related to multi-agent cooperation. It is proved to be a powerful rival for EAs, while benefiting from the same interactive population notion as the GA. In many applications, PSO has been shown to be faster in converging to near-optimum solutions than the GA [5]. Due to the high flexibility and adaptability of the PSO, it has been used as the basis of numerous optimization methods. In fact, many researchers have tried to enhance the original PSO through a wide variety of techniques, such as by improving the method of parameter tuning and updating, adopting efficient strategies for accelerating convergence, introducing additional parameters into the system, and of course by hybridization [6,29]. Through hybridization, various algorithms are combined to compensate for the drawbacks and reinforce the strengths of each other, and PSO has been extensively integrated in hybrid optimization methods [6].

The PSO algorithm is currently one of the most frequently used meta-heuristics in solving optimization problems such that the number of papers published on PSO highly increases each year. This population-based algorithm was first introduced by Kennedy and Eberhart in 1995 [7], who used the idea of swarms in nature such as birds and fish. PSO has some particles (solutions) which combine self-experiences with social experiences. The particles start their 'fly' from random positions in a search area and move in the search space towards a promising area to (hopefully) reach the global optimum. The PSO algorithm has a main nested loop that terminates when the total number of function evaluations (iterations) exceeds a certain limit or a minimum error threshold is achieved. In each function evaluation, particles update their positions and fly to another position affected by a fitness function that assesses the quality of each solution. The main variables in PSO include the best fitness value of each particle (*pbest*) and the

best fitness value of the whole swarm (*gbest*), as well as some parameters including the inertia weight (for regulating the global search), balancing factors (for balancing self- and social knowledge of particles), the constriction factor (for limiting the velocity of the particles), and random numbers.

This paper proposes a new variant to the original PSO, called GEPSO, to enhance the capability of the PSO algorithm by incorporating two new terms in order to efficiently update the velocities of the particles. These new terms are inspired by REPSO and PSOPC algorithms. In addition, a new mechanism for adjusting the inertia weights is presented. Extensive computational experiments showed the effectiveness of the adopted new terms and the relative performance of the GEPSO compared to the original PSO and five variants of it, namely, Repulsive PSO (REPSO), PSO with Passive Congregation (PSOPC), Negative PSO (NPSO), Deterministic PSO (DPSO), and Line Search-Based Derivative-Free PSO (LS-DF-PSO).

1.2. Literature review

During the last two decades, numerous variations of the original PSO method have been introduced for various applications, and regarding the weaknesses and strengths of each algorithm, their success in reaching optimal or near-optimal solutions has been different from a problem to another. Some examples of such algorithms and applications are Adaptive fuzzy PSO algorithm by [34]; a hybrid of PSO and a definite selection procedure (EPSO) [23]; a vector-based PSO [28]; Set-based PSO [25] to determine ribonucleic acid secondary structure; IPSO [20] for solving prediction and control problems; a hybrid of PSO and EM for designing FLPRFNS [19]; Chaos-embedded PSO [3] in order to tune the parameters of a PID controller; and CTB-PSO [16], in which individuals within a swarm, as opposed to discrete behavior grouping, have traits based on a continuous scale. Also, the hybrid GA-PSO [10] for curve fitting in manufacturing and the hybrid bare-bones PSO [41] for dynamic economic dispatch are among some PSO-based hybridization algorithms. In addition, [15] has introduced ELPSO for mitigating the premature convergence problem based on a five-staged successive mutation strategy. In 2015, a new PSO algorithm with adaptive inertia weight used a Bayesian technique [42]. In the same year, a modified PSO technique based maximum power point tracking for uniform and under partial shading condition [4]. In 2016, novel stability based adaptive inertia weight for PSO was developed [35]. In 2017, a PSO algorithm with a new update mechanism was developed in [17] and a novel improved PSO algorithm based on individual difference evaluation was proposed in [12]. In 2018, correlation feature selection based improved-binary PSO for gene selection and cancer classification was devised [14], and also a meta-optimization analysis of PSO velocity update equations for watershed management learning was presented in [22]. Recently, Serani and Diez [31] studied the effects of using random coefficients in PSO for ship hydrodynamics simulation-based design optimization.

In [30], we presented a new taxonomy of the attributes of PSO-based methods that covers 22 classes in four general categories of variables, particles, swarm, and process. A general view of that taxonomy is presented in Table 1, the details of which can be found in [30]. Also, in [38] we developed an Expert System to identify the best-fitting variant of PSO (among 100 PSO-based algorithms) given the specifications of the problem at hand (e.g., type of search space, type of variables, hybridization, the purpose of the application, etc.). Indeed, the new PSO-based algorithm presented in the current paper was inspired after the evaluation of the different perspectives of PSO-based algorithms in [30] and [38] and aims to overcome some of the weaknesses of current variants.

The problem of determining which existing algorithms with what parameters are able to solve a specific problem more efficiently is a concern for researchers. According to the *No Free Lunch (NFL)* theory, two optimization algorithms are equivalent whenever their performance is averaged across all possible problems. In other words, assigning each problem to a particular algorithm reduces the burden cost than using a fixed algorithm for all of the problems [38].

Many papers have been published so far in which a limited number of PSO-based methods were compared for a specific problem. For instance, Tao et al. [37] compared the ability of RHDPSO in overcoming the premature convergence and local optimum issues with the discrete PSO algorithm and the results showed the advantage of RHDPSO. In order to evaluate the effectiveness of FODPSO in solving multilevel image segmentation problems, [11] compared the FODPSO with SPSO and original PSO methods. In [33], a parameter selection is proposed in synchronous and asynchronous DPSO for ship hydrodynamics problems. It shows that the performance of the DPSO is mainly dependent on the selected parameters for the implementation of the algorithm. In [32], a globally convergent hybridization of PSO using line search-based derivative-free techniques is proposed that is referred to as LS-DF-PSO. It uses exact and heuristic derivative-free methods for global unconstrained optimization problems.

Table 1
Taxonomy of the attributes of PSO-based algorithms [30].

Category	Attribute
Variables	*Constrainment* (Constrained | Unconstrained); *Stochasticity* (Deterministic | Stochastic), *Type* (Continuous | Integer | Continuous + Integer); *VelocityType* (Restricted | Unrestricted Velocity | Vertical Velocity | Limited Velocity | Escape Velocity | Adaptive Velocity); *Fuzziness* (Fuzzy | Crisp); *Space Continuity* (Continuous | Discrete | Binary).
Particles	*Accordance* (Adaptive | Dissipative | Adaptive + Dissipative); *Attraction* (Attractive | Repulsive | Attractive + Repulsive); *Association* (Aggregation | Passive | Active | Congregation | Social); *Dynamics* (Newtonian | Quantum); *Hierarchy* (Hierarchical | Non-hierarchical); *Mobility* (Static | Dynamic); *Synchronicity* (Synchronous | Asynchronous); *Trajectory* (Positive | Negative);
Swarm	*Cooperation* (Cooperative | Non-cooperative); *Topology* (Gbest | Lbest | Pyramid | Star | Small | VISbest | Random Graphs); *Activity* (Active | Passive); *Divisibility* (Divided | Undivided)
Process	*Recursiveness* (Recursive | Sideway); *Hybridization* (Genetic Algorithms | Ant Colony Optimization | Differential Evolution | Immune Systems | Neural Networks); *Objective* (Single | Multiple); *User Interaction* (Interactive | Non-Interactive)

1.3. Contributions and organization

The proposed new GEPSO method benefits from an improved velocity update formula that enhances diversification in the search space and prevents premature convergence. In addition, a new dynamically evolving inertia weight is proposed which intensifies the search procedure toward the objective in later function evaluations. Also, we have extensively compared the GEPSO with six PSO-based methods, namely, original PSO, REPSO, PSOPC, NPSO, DPSO, and LS-DF-PSO. Finally, the results are analyzed for 16 benchmark optimization functions.

The main advantages of GEPSO can be summarized as follows:

- A stronger interrelation between the particles is maintained compared to the original PSO,
- A better exploration of the swarm in various unexplored regions of the search space caused by random velocities is realized,
- Additional parameters are introduced as factors in the velocity updating equation for performance enhancement,
- The particle velocity updating formula is improved and a dynamic and adaptive inertia weight adjustment mechanism is proposed.

The rest of the paper is organized as follows: the components of the proposed GEPSO method and the effects of the terms and parameters are described in Section 2. Section 3 introduces five PSO-based algorithms that are selected for comparison with the GEPSO, and describes the experimental results and comparisons between seven PSO-based algorithms for 16 benchmark functions. Finally, conclusions and future research directions are presented in Section 4.

2. The GEPSO algorithm

2.1. Original PSO

In the original PSO proposed by Eberhart and Kennedy, the position and velocity of any particle are updated as follows [7], the symbols of which are defined in the Nomenclature:

$$Prtpos_j^i = Prtpos_j^{i-1} + Prtvel_j^i \qquad \forall i \in N_{iter}, \forall j \in N_p \tag{1}$$

$$Prtvel_j^i = wPrtvel_j^{i-1} + c_1 r_1^i \left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + c_2 r_2^i \left(gbest^{i-1} - prtpos_j^{i-1}\right) \qquad \forall i \in N_{iter}, \forall j \in N_p \tag{2}$$

The cognitive and collective acceleration factors (c_1, c_2) balance the effect of self-knowledge and social knowledge on the particles' movement towards the goal. The inertia weight w regulates the global search behavior, usually preset at the beginning of the search process with a range suggested to be 0.2 to 0.4 and dynamically

reduced during the optimization (which means a deeper local search behavior). Dynamic adjustment of the inertia weight has some advantages: first, it causes faster convergence to an optimal solution, and second, it controls the effect of preceding part velocities on current velocities, hence adjusting the tradeoff between the capability of the swarm in local and global exploration.

2.2. GEPSO

In this paper, in order to improve the performance of the original PSO and in fact, to increase its efficiency, the Generalized PSO (called GEPSO) is proposed which enhances the particle velocity updating formula and provides a dynamic inertia weight adjustment mechanism. While the position updating equation in GEPSO is the same as in the original PSO, the velocity updating equation is modified by adding two new terms, as follows:

$$Prtvel_j^i = \psi\left[w_1^i Prtvel_j^{i-1} + w_2 c_1 r_1^i\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + w_3\alpha_1 c_2 r_2^i\left(gbest^{i-1} - prtpos_j^{i-1}\right)\right.$$
$$\left. + w_4\alpha_2 c_3 r_3^i\left(pbest_{rand}^{i-1} - prtpos_j^{i-1}\right) + w_5\alpha_3 c_4 r_4^i Prtvel_{rand}\right] \qquad \forall i \in N_{iter}, \forall j \in N_p \tag{3}$$

The first term of (3) denotes the previous velocity of a particle, which produces the momentum needed for its flying across the search space. The second term, which is known as the *cognitive* component, simulates the 'personal' experience of a particle and encourages it to fly towards the best position ($pbest$) it has found up until the current function evaluation. The third term, called the *collective* component, simulates the effect of particles' cooperation in finding the global optimum and drives a particle towards the best position ever found among all of the members of the swarm ($gbest$). The fourth term, which we have added as a new term and call it *random self-cognition*, pushes a particle towards a randomly-selected best position found by other particles ($pbest_{rand}^{i-1}$), which in fact provides for other particles an opportunity to randomly share their knowledge during velocity updating. The fifth term is again a new term introduced here and incorporates the effect of a random velocity parameter ($Prtvel_{rand}$), thus leading to an increased variety in the swarm on one hand, and a more effective movement of the swarm in narrow and complex search spaces on the other hand.

In the proposed algorithm, the inertia weight factor is employed in all terms in (3) in the form of w_1 to w_5. Also, in all terms, except for the first term (velocity in previous function evaluation), acceleration constants c_1 to c_4 are utilized. The procedure described in (4) is employed to dynamically update the inertia weight w_1 in each function evaluation:

$$w_1^i = \min\left\{w_{min}, w_1^{i-1} - \left[\frac{(w_{max} - w_{min})}{iter_{max}} i(f\left(gbest^{i-1}\right) - f\left(gbest^{i-2}\right))\right]\right\} \qquad \forall i \in N_{iter} \tag{4}$$

Note that (4) is proposed as such for minimization problems and the (first) minus sign therein must change to plus sign for maximization problems. The equation guarantees that during the search process, w_1^i will always be greater than or equal to a minimal inertia value (w_{min}). Through this updating mechanism, $w^i{}_1$ increases proportional to the amount of the improvement of the swarm's best fitness function compared with its previous function evaluation (that is, when $f\left(gbest^{i-1}\right) - f\left(gbest^{i-2}\right) < 0$). As a result, the effect of the current velocity direction ($Prtvel_j^i$ in the first term in (3)) augments when the $gbest$ improves, which leads to deeper exploitation of the current solution. On the other hand, w_1^i decreases whenever a worse fitness function value is obtained compared with the previous function evaluation (i.e., when $f\left(gbest^{i-1}\right) - f\left(gbest^{i-2}\right) > 0$), which in turn discourages the particle to continue along its previous direction $Prtvel_j^i$, leading to more exploration of the search space.

It is also noted that the magnitude of the change in successive w_1's grows steadily since the term ($f\left(gbest^{i-1}\right) - f\left(gbest^{i-2}\right)$) is multiplied by a normalized factor $\frac{(w_{max}-w_{min})}{iter_{max}} \times i$ that gradually increases as the number of elapsed iterations (i) increments. This means that the factor is small in early iterations (as i is small), and so the algorithm's behavior is more exploration-like (diversification), whereas in later iterations (when i approaches $iter_{max}$) the factor becomes larger, which scales up the amount of fitness improvement and encourages the particle to move along its current velocity vector, leading to an exploitation-like (intensification) behavior. For the first iteration, the value of w_1^0 is set to w_{min} (suggested to be 0.2).

Furthermore, we introduce an updated formula for the constriction parameter ψ as in (5), and the particles' positions and velocities are initialized as in (6) and (7) in which the $x_{k,min}$ and $x_{k,max}$ are respectively the minimum

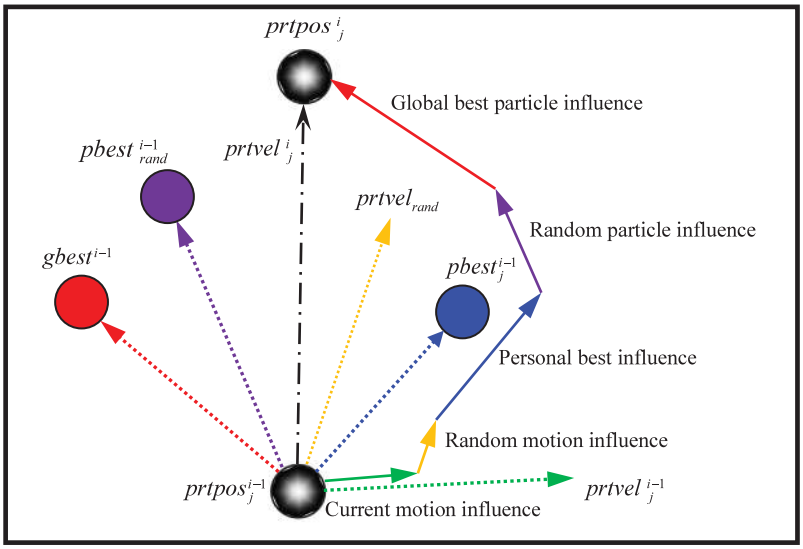


Fig. 1. Depiction of a particle's velocity update in GEPSO, which is the sum of five vectors. Note the parallel vectors.

and maximum values of the kth coordinate of the particles [24].

$$\psi = \frac{2}{\left|2 - (c_2{+}c_3)^2 - 5(c_2{+}c_3)\right|} \tag{5}$$

$$Prtpos_j^0\,(x_k) = x_{k,min} + rand x_{k,max} \qquad \forall j \in N_p, k \in N_k \tag{6}$$

$$Prtvel_j^0(x_k) = Prtpos_j^0\,(x_k) + \gamma_k \qquad \forall j \in N_p, k \in N_k \tag{7}$$

Fig. 1 illustrates a schematic view of updating the position of a particle in two successive function evaluations (iterations) of the GEPSO algorithm. Also, the overall procedure of the GEPSO is presented in Fig. 2.

2.3. The velocity terms in GEPSO

A notable feature of the GEPSO algorithm is the stronger interrelations between the particles than in the original PSO (due to the fourth term in (3)), which causes the swarm to converge toward better quality solutions more rapidly. Also, the effect of random velocities (due to the fifth term in (3)) causes a better exploration of the swarm in various unexplored regions of the search space. Although the last two terms of velocity updating formula are the most visible novelty in the GEPSO, the introduced additional parameters multiplied in the first three terms also contribute to performance enhancement of the algorithm.

In the following, the role of each term in the velocity update formula of GEPSO is described:

The first term: The w_1^i in this term is the inertia weight, which changes dynamically from a large value to a small value, thus causing better global search (exploration) in early function evaluations, and then better local search (exploitation) in later function evaluation. As a result, a tradeoff between global and local searches is obtained which leads to convergence towards better, near-optimal solutions. A similar tradeoff view has been provided in BPSO [42]. Another function of w_1^i is controlling the impact of particle's previous velocity $Prtvel_j^{i-1}$ on its current velocity, therefore providing the momentum needed for flying the particle in the whole search space.

The second term: This term, also known as the cognitive component, simulates the accumulation of 'experience' in a particle and encourages the particles to fly towards the best position they have found so far. Particularly, c_1 in this term is known as 'cognitive acceleration factor' and has a role of balancing the impact of each particle's individual intelligence. The parameter w_2, which is used in the cognitive part for the first time, represents the momentum of a particle's individual intelligence in determining its next position. It is worth noting that by multiplying the random

```
Procedure GEPSO
While maximum number of function evaluations or minimum error criteria is not attained Do
    For each particle
        Initialize particles according to (6) and (7)
    End
    For each particle Do
        Calculate fitness value
        If the fitness value is better than the best fitness value (pbest) in history Then
            Set the current value as the new pbest
        End
        Select randomly a pbest
        Find in the particle's neighborhood the particle with the best fitness (gbest)
        Select randomly a velocity for the particle (prtvel_rand)
        Select the best value of a random particle (pbest_rand)
        Update the inertia weight using equation (4)
        Calculate particle velocity according to the velocity update equation (3)
        Apply the velocity constriction
        Update particle position according to the position update equation (1)
        Apply the position constriction
    End
End
```

Fig. 2. Pseudocode of the GEPSO.

value of $r_1^i \in (0, 1)$, which is different in various function evaluations, this paper tries to increase the diversification ability of the algorithm and search for more unexplored regions of the search space.

The third term: This is the impact of the whole swarm's intelligence on determining the next positions of individual particles. The parameter w_3 is used here as the momentum of this impact and c_2 plays the role of regulating the impact of swarm intelligence on the particle. The α_1 regulates the probability of using this term. The role of the parameter $r_2^i \in (0, 1)$ is the same as r_1^i in the preceding term.

The fourth term: This newly-introduced term relates the last position of a particle with the best position of random particles. As a result, a stronger interrelation between particles, and hence in the whole swarm, is created in a positive way, meaning that each individual particle benefits from the best cognitive experience of another particle in each function evaluation. The resulting effect, on one hand, is a higher probability in converging toward good solutions, and higher diversification of the search process on the other hand. The parameter w_4 determines the momentum of this term in determining the next position of the particle. The probability of utilizing this effect is regulated by α_2. Also, $r_3^i \in (0, 1)$ has the same role as the previous ones.

The fifth term: This term incorporates a random velocity that contributes to increasing the power of exploring the space and preventing the algorithm from being trapped in local optima. In this term, w_5 is the momentum of random velocity impact on the particle's global movement, c_4 has the role of balancing the impact of random velocity, α_3 is this term's impact percentage with respect to the impacts of the preceding two terms. The role of $r_4^i \in (0, 1)$ is the same as in [3].

2.4. Evaluating the effects of the parameters in them.

The parameters of meta-heuristic algorithms have a significant effect on the efficiency and effectiveness of the search for a particular problem [1,36]. There may be many options for such factors for a given problem. Therefore, using an appropriate parameter tuning to find the best values from among many alternatives can produce better solutions for a given problem. In Table 2 we have described the function of each of the parameters in the GEPSO and have provided their best ranges as well as some guidelines for tuning them according to the conditions of the problem and the search process after extensive experiments on various problems.

Table 2
Parameters of the GEPSO and guidelines for tuning them.

Symbol	Function	Suggested range	Conditions for increase (▲) and decrease (▼)
c_1 c_2 c_3 c_4	Attracts the particle's position towards its *pbest* Attracts the particle's position towards *gbest* Attracts the particle's position towards the *pbest* of a particle selected randomly Controls the randomness of the *prtvel*	[1.5, 4]	▲ High velocity and acceleration of particle movements are required ▼ Low velocity and acceleration of particle movements are required
$w^i{}_1$	Maintains the particle's current velocity	[0.2, 2]	Tuned dynamically according to Eq. (4)
$w^i{}_2$ to $w^i{}_4$	Control the effects of their multiplicands in the overall new *prtvel*	[0.4, 0.9]	▲ Local exploitation is emphasized ▼ Global exploration is emphasized
$w^i{}_5$	Controls the effect of a random *prtvel*	[0.4, 0.9]	▲ Global exploration is emphasized ▼ Local exploitation is emphasized
α_1	Controls the influence of the *gbest*	[0, 5]	▲ Greedy convergence for solving non-complicated objective functions is required. ▼ Cautious convergence when solving complicated objective functions is required.
α_2	Controls the influence of the *pbest* of a particle selected randomly	[0, 3]	▲ Cautious convergence is required when solving complicated objective functions ▼ Greedy convergence for solving non-complicated objective functions
α_3	A random factor for *prtvel*	[0, 3]	▲ Moving in farther distances from the primary position of particles (progressive motions) is required. ▼ Moving in closer distances to the primary position of particles (conservative motions) is required.
N_p	Number of particles	[10, 10^3]	▲ More accuracy in reaching the goal in more time ▼ Less accuracy in reaching the goal in less time
$iter_{\max}$	Maximum number of function evaluations	[10, 10^4]	▲ More accuracy in optimizing the selected function spending more time ▼ Less accuracy in optimizing the selected function but with higher speed

3. Comparison with other algorithms

The effectiveness and efficiency of the proposed GEPSO have been compared with the original PSO and five other PSO-based methods, REPSO, PSOPC, NPSO, DPSO, and LS-DF-PSO, which are briefly explained below:

REPSO [18]: The Repulsive PSO algorithm belongs to the class of stochastic evolutionary optimizers. There are several different realizations of REPSO, and common to all realizations is the repulsion between particles. In the repulsion mechanism, particles move away from positions that are seen as best and thus explore new areas of the search space. This can prevent the swarm from being trapped in local optima, which would cause a premature convergence and would lead the algorithm to fail in finding the global optimum. When the desired diversity level is reached, the algorithm tries to switch back to the attraction phase to exploit the newly-explored areas. The particles' velocities in REPSO are updated as indicated in Table 3.

PSOPC [13]: Swarms in nature keep their collective shape under two types of grouping forces: *Aggregation*, and/or *Congregation*. Aggregation may be either (1) Passive, in which a passive (not self-moving) swarm moves under a physical force (like a swarm of planktons floating on the water such that the flow of water keeps them together), and (2) Active, which is realized by an absorbent source such as food or water. On the other hand, the Congregation is the absorbent supply or the group force by self, which is not by external and physical factors. The Congregation too may be either (1) Passive, in which there is an attraction from one particle to others but is not shown through a social

Table 3
The comparison of the velocity updating in original PSO, REPSO, PSOPC, NPSO, DPSO, and LS-DF-PSO.

Variants of the PSO	Velocity updating equation, $Prtvel_j^i$
Original PSO	$\left[wPrtvel_j^{i-1} + c_1r_1\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + c_2r_2\left(gbest^{i-1} - prtpos_j^{i-1}\right)\right]$
REPSO	$\left[wPrtvel_j^{i-1} + c_1r_1\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + wc_2r_2\left(gbest^{i-1} - prtpos_j^{i-1}\right) + wc_3r_3Prtvel_{rand}\right]$
PSOPC	$\left[wPrtvel_j^{i-1} + c_1r_1\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + c_2r_2\left(gbest^{i-1} - prtpos_j^{i-1}\right) + c_3r_3(prtpos_{rand}^{i} - prtpos_j^{i-1})\right]$
NPSO	$\left[wPrtvel_j^{i-1} + c_1r_1\left(prtpos_j^{i-1} - pworst_j^{i-1}\right) + c_2r_2\left(prtpos_j^{i-1} - gworst_j^{i-1}\right)\right]$
DPSO[a]	$\psi_1\left[Prtvel_j^{i-1} + c_1\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + c_2\left(gbest^{i-1} - prtpos_j^{i-1}\right)\right]$
LS-DF-PSO[b]	$\psi_1\left[Prtvel_j^{i-1} + c_1\left(pbest_j^{i-1} - prtpos_j^{i-1}\right) + c_2\left(gbest^{i-1} - prtpos_j^{i-1}\right)\right]$

[a]Details of the calculation of the ψ_1 can be found in [33].
[b]Details of the velocity updating are explained in [32].

behavior, and (2) Social, in which there is a social behavior among the particles that strongly relates them to each other. When in some groups there is a selfish behavior in information sharing (like in fish school), that may lead to forming a passive group. A passive swarm model can be added to the PSO in order to increase its efficiency, which results in the PSO with Passive Congregation (PSOPC) algorithm. The velocity in PSOPC is updated as shown in Table 3.

NPSO [39]: Negative PSO (NPSO) is a modified form of the original PSO with a strategy to avoid a particle's previous worst solution and its group's previous worst based on similar formulas of the original PSO. These terms, however, are utilized with negative signs in the velocity updating equation. In other words, this process tries to get farther from the worst instead of getting closer to the best. The velocity updating equation in NPSO is shown in Table 3.

DPSO [33]: Among deterministic methods, Deterministic PSO (DPSO) has numerous attractive characteristics such as the simplicity of the heuristics, the ease of implementation, and its often fairly remarkable effectiveness. The performances of DPSO depend on four main setting parameters: the number of swarm particles, their initialization, the set of coefficients defining the swarm behavior, and (for box-constrained optimization) the method to handle the box constraints. This version of the PSO is more applied in simulation-based designs and does not use random numbers in the velocity updates formula, as displayed in Table 3.

LS-DF-PSO [32]: This hybrid algorithm combines the attractive properties of the Line Search-based Derivative-Free (LS-DF) exact method with the PSO meta-heuristic method. LS-DF is capable of finding the global best solution while PSO finds a local optimum in shorter times, and thus hybridizing those two creates a powerful algorithm that is both effective and efficient. This algorithm uses a direct method for updating the particles' positions, without computing their velocities first.

The reasons for selecting the abovementioned algorithms for comparing with the GEPSO are as follows:

(1) The basic PSO originally developed by [7] is a benchmark algorithm usually tested against any new PSO-based algorithm developed afterward,
(2) The particle velocity update formula in REPSO has the term ($Prtvel_{rand}$) among others which is also used in GEPSO (as its fifth term), and by comparing the GEPSO and REPSO, we are able to assess the effect of the proposed fourth term and other introduced parameters,
(3) The PSOPC uses a term for attracting a particle toward the position of a random particle in the previous function evaluation, while in GEPSO there is a term for attracting a particle toward the best position of a random particle so far, thus comparisons are made to test the effect of this difference among others,
(4) While in the GEPSO diversification and exploration of unknown regions of the search space is secured by using the fourth and fifth terms in the velocity update formula, diversification in NPSO is done through a 'reverse' mechanism, realized by repelling the particles away from their worst positions, which causes the particles not to converge toward *gbest*s and *pbest*s, and in this way explore other regions of the search space. We wanted to

compare the diversification power of the GEPSO with that of the NPSO algorithm, which particularly focuses on this issue,

(5) The simplicity and speed of convergences of the DPSO are two important motivations for comparing GEPSO with it, and through that, the effect of random elements in GEPSO are also compared to the case when no randomness is introduced into the PSO, and

(6) Comparing the GEPSO with LS-DF-PSO can test its efficacy against a hybrid and comprehensive algorithm.

4. Computational experiments and analysis

The computational experiments were done by solving minimization problems of 16 standard benchmark functions, namely, Sphere, Griewank, Rastrigin, Schaffer, Ackley, Quintic, Bukin's, Goldstein–Price, Modified RCOS, Generalized Schwefel, Alpine, Dixon–Price, Pathological, Powell, Vincent, and Zacharov [33]. Table 4 shows the details of these benchmark functions. All algorithms were coded in Matlab™ and run on an Intel Core™ 2 Duo 2 GHz CPU with 4 GB of RAM.

Before the experimentation, the parameters of the algorithms were tuned and set for each benchmark function, as shown in Table 5. For all algorithms (except for DPSO and LS-DF-PSO algorithms), the population size was set to 100 particles. Considering the stochastic behavior of the methods (except for DPSO and LS-DF-PSO algorithms), each benchmark function was solved up to 10, 100, 1000, and 10,000 fitness function evaluations (iterations) by each algorithm, after which a number of performance metrics were measured and reported. For DPSO and LS-DF-PSO algorithms, the population size and other parameters are set up based on [33] and [32], respectively.

Following a trial-and-error parameter tuning, the parameters of the GEPSO algorithm are set for each benchmark function. Table 5 shows the parameters of the used algorithms in this paper.

Considering the stochastic behavior of the algorithms (except DPSO and LS-DF-PSO), each benchmark function was solved 10 000 times (runs). To better assessment of the efficiency of the algorithms, the various number of the function evaluations (iterations) are taken into account that are 10, 100, 1000, and 10 000. For computational results, in addition to the average runtime of each algorithm per each benchmark function, means (μ) and standard deviations (σ) of the best fitness obtained by each algorithm per each problem was calculated. Also, three other performance criteria (metric), i.e., Δx, Δf, and Δt which were adapted from [33] and are defined as follows were computed:

$$\Delta x = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\frac{x_{i,\min} - x^*_{i,\min}}{|u_i - l_i|}\right)^2} \tag{8}$$

$$\Delta f = \frac{f_{\min} - f^*_{\min}}{f^*_{\max} - f^*_{\min}} \tag{9}$$

$$\Delta t = \sqrt{\frac{\Delta x^2 + \Delta f^2}{2}} \tag{10}$$

in which Δx represents the *normalized Euclidean distance* between the minimum position found by the algorithm ($\mathbf{x}_{\min}$) and the global minimum position ($\mathbf{x}^*{}_{\min}$), $x_{i,\min}$ is the position of the ith variable at the best-found solution, $x^*{}_{i,\min}$ is the position of the ith variable at the global minimum solution, d is the dimension of the problem (i.e., the number of decision variables), and u_i and l_i are respectively the upper and lower limits of the ith variable. Δf is the *associated normalized distance* in the solution landscape, in which $f_{\min}$ is the minimum found by the algorithm, and $f_{\min}{}^*$ and $f_{\max}{}^*$ are respectively the global minimum and maximum of the function $f(\mathbf{x})$ in its domain. Finally, Δt is a combination of Δx and Δf and is used for overall assessment.

Table 6 presents the average runtimes, means, standard deviations, as well as the Δx, Δf, and Δt criteria for all algorithms and 16 benchmark functions. The overall average and standard deviation performance among benchmarks for each metric per each algorithm are also shown in the last two columns of Table 6. As the results in Table 6 show, the GEPSO algorithm is better than other algorithms in the term of runtime. For instance, for the Quintic function, was relatively better than original PSO, RESPO, PSOPC, NPSO, DPSO, and LS-DF-PSO by 18.3%, 18.1%, 33%, 26%, 22.5%, and 12%, respectively. These results for average of runtime among all benchmark functions were 16%, 9%, 14.8%, 14.9%, 12% and 12.2%. For Δt term, GEPSO was relatively better than original PSO, RESPO,

Table 4
The solved benchmark functions.

Function	Mathematical equation	Global minimum
Sphere	$f_1 = \sum_{i=1}^{d} x_i^2$	$f_1(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Griewank	$f_2 = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \left(1 + \cos(\frac{x_i}{\sqrt{i}})\right)$	$f_2(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Rastrigin	$f_3 = 10d + \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i)\right)$	$f_3(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Schaffer 6	$f_4 = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	$f_4(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, 0)$
Ackley	$f_5 = 20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}$	$f_5(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Quintic	$f_6 = \sum_{i=1}^{d} \left\lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4\right\rvert,\ d = 50$	$f_6(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (-1, \ldots, -1)$
Bukin 6	$f_7 = 100\sqrt{\lVert x_2 - 0.01x_1^2\rVert} + 0.01\lVert x_1 + 10\rVert$	$f_7(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (-10, 1)$
Goldstein–Price	$f_8 = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]\cdot$ $[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$f_8(\mathbf{x}^*) = 3$ at $\mathbf{x}^* = (0, -1)$
Branin's RCOS 2	$f_9 = \frac{-1}{(g_1 + g_2 + g_3 + 10)};\ g_1 = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5x_1}{\pi} - 6\right)^2;$ $g_2 = 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1)\cos(x_2);\ g_3 = \log(x_1^2 + x_2^2 + 1).$	$f_9(\mathbf{x}^*) = -0.179891239$ at $\mathbf{x}^* = (-3.19698842, 12.52625788)$
Generalized Schwefel 2.26	$f_{10} = -\left(x_1 \sin\sqrt{\lvert x_1\rvert}\right) - \left(x_2 \sin\sqrt{\lvert x_2\rvert}\right)$	$f_{10}(\mathbf{x}^*) = -837.9658$ at $\mathbf{x}^* = (420.9687, 4209687)$
Alpine 1	$f_{11} = \sum_{i=1}^{d} \lvert x_i \sin x_i + 0.1x_i\rvert,\ d = 50$	$f_{11}(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Dixon–Price	$f_{12} = (x_1 - 1)^2 + \sum_{i=2}^{d} i\left(2x_i^2 - x_{i-1}\right)^2,\ d = 50$	$f_{12}(\mathbf{x}^*) = 0$ at $x_i^* = 2^{-\left(1 - \frac{1}{2^i - 1}\right)}$
Pathological	$f_{13} = \sum_{i=1}^{d-1}\left[0.5 + \frac{\sin^2\left(\sqrt{100x_i^2 + x_{i+1}^2}\right) - 0.5}{1 + 0.001\left(x_i^2 - 2x_ix_{i+1} + x_{i+1}^2\right)^2}\right],\ d = 50$	$f_{13}(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$
Powell Singular	$f_{14} = \sum_{i=1}^{n/4}\left[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\right],\ n = 48$	$f_{14}(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)$
Vincent	$f_{15} = -\frac{1}{d}\sum_{i=1}^{d} \sin(10\log x_i),\ d = 50$	$f_{15}(\mathbf{x}^*) = -1$ at $x_i^* = m \cdot e^{0.1\sin^{-1}(1)},\ m = 1, 5, 9, \ldots$
Zakharov	$f_{16} = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i \cdot x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i \cdot x_i\right)^4,\ n = 50$	$f_{16}(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$

PSOPC, NPSO, DPSO, and LS-DF-PSO by 4%, 7%, 6%, 8%, 65 and 4.5% respectively, which shows that LS-DF-PSO also proved to be comparably effective. Considering the standard deviation of results among benchmark functions, GEPSO was relatively better than original PSO, RESPO, PSOPC and DPSO by 20%, 29%, 29%, 7%. In addition, from the mentioned standard deviation, GEPSO was similar to NPSO and 9% worse than the LS-DF-PSO. Other similar comparisons can be extracted from Table 6.

For a more specific analysis of the performance of the GEPSO, consider the Generalized Schwefel function that is depicted in Fig. 3.

Table 5
Parameters of the compared algorithms.

PSO variant	Parameters
Original PSO	$c_1 = c_2 = 3,\ w = 0.5$
REPSO	$c_1 = c_2 = c_3 = 3,\ w = 0.5$
PSOPC	$c_1 = c_2 = c_3 = 3,\ w = 0.5$
NPSO	$c_1 = c_2 = 3,\ w = 0.5$
DPSO	$c_1 = 2.050,\ c_2 = 2.050,\ \psi_1 = 0.729$ as defined in [33]
LS-DF-PSO	$h_k = 1,\ \gamma = 10^{-3},\ \theta = 0.5,\ \alpha_k = 0.25$ as defined in [32]
GEPSO	$c_1 = c_2 = c_3 = c_4 = 3,\ \alpha_1 = 3,\ \alpha_2 = 2,\ \alpha_3 = 2,\ w_2 = w_3 = w_4 = w_5 = 0.5$

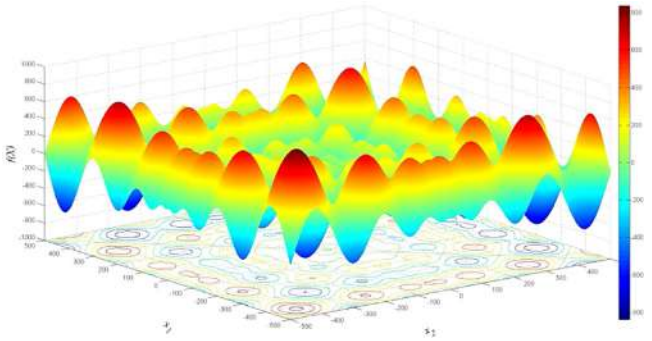


Fig. 3. The generalized Schwefel function in 2D.

This function has a uniform random nature and its local optima are scattered all over the search space. In order to effectively solve this problem that has a rugged landscape, a search algorithm must have proper intensification abilities to ensure finding a good local optimal solution quickly. On the other hand, the algorithm needs to implement an efficient diversification scheme for better investigation of unexplored regions. Thanks to better interrelations between the particles in the GEPSO (due to the random self-cognition component), it managed to converge to optimal solutions in a timely manner. Moreover, the effects of random velocities and the dynamic and adaptive inertia updating mechanism reinforced the exploratory ability of the GEPSO in narrow and complex areas of the function. By benefiting from such intensification and diversification abilities, GEPSO attained significantly lower errors compared to the other algorithms.

To perform a deeper analysis on the effects of the additional terms in GEPSO on finding the optimal solution, four scenarios were designed and evaluated: (1) GEPSO without adaptive inertia updating and without fourth and fifth terms in the velocity updating equation, (2) GEPSO with adaptive inertia updating but without the fourth and fifth terms in the velocity updating equation, (3) GEPSO with adaptive inertia updating, with the fourth term, but without the fifth term in the velocity updating equation, and (4) full GEPSO with adaptive inertia updating and all terms. Table 7 shows the means, standard deviations of the minimal (optimal) solutions, and other metrics of the Sphere function for the abovementioned scenarios obtained after 10,000 runs. The results show that the additional terms incorporated into the velocity update formula improved all of the performance metrics (smaller is better).

More specifically, the following observations were made:

- The net effect of adaptively changing the inertia weight ($w^i{}_1$) (i.e., Scenario 2 vs. 1) was an improvement of the μ, σ, Δx, Δf, and Δt by 10.7%, 5.9%, 0.5%, 1.3%, and 0.58%, respectively, with the highest impact on μ.
- The net effect of adding the fourth term to the velocity updating formula (i.e., Scenario 3 vs. 2) was an improvement of the μ, σ, Δx, Δf, and Δt by 6.5%, 19.8%, 1.04%, 2%, and 1.7%, respectively, with the highest impact on σ.
- The net effect of adding the fifth term to the velocity updating formula (i.e., Scenario 4 vs. 3) was an improvement of the μ, σ, Δx, Δf, and Δt by 13.7%, 7.3%, 1.05%, 0.68%, and 0.59%, respectively, with the highest impact on μ.

Table 6
The metrics of the solved benchmark functions. Reported times are seconds $\times 10^3$.

Method	Criteria	Benchmark functions															Average	SD
		Sphere	Griewank	Rastrigrin	Schaffer	Ackley	Quintic	Bukin's	Goldstein-Price	Modified RCOS	Alpine	Dixon–Price	Pathological	Powell	Vincent	Zacharov	Generalized Schwefel		
PSO	Runtime	2.281	2.113	1.822	2.376	2.521	2.171	2.127	2.159	1.964	2.867	2.333	2.891	2.379	1.952	2.972	2.181	2.319	0.342
	Δx	0.192	0.182	0.211	0.173	0.201	0.218	0.164	0.195	0.189	0.196	0.202	0.194	0.211	0.189	0.179	0.199	0.193	0.014
	Δf	0.152	0.157	0.161	0.135	0.162	0.173	0.129	0.157	0.137	0.147	0.167	0.155	0.197	0.149	0.134	0.147	0.154	0.017
	Δt	0.173	0.170	0.188	0.155	0.183	0.197	0.148	0.177	0.165	0.173	0.185	0.176	0.204	0.170	0.158	0.175	0.175	0.015
REPSO	Runtime	2.121	1.984	1.943	2.0157	2.657	2.166	1.896	2.324	2.136	3.014	1.899	2.191	2.455	2.014	1.473	1.862	2.134	0.356
	Δx	0.198	0.191	0.223	0.191	0.242	0.234	0.198	0.202	0.194	0.203	0.194	0.204	0.186	0.196	0.186	0.187	0.202	0.017
	Δf	0.149	0.154	0.192	0.143	0.201	0.194	0.157	0.167	0.168	0.169	0.149	0.169	0.142	0.178	0.143	0.145	0.164	0.019
	Δt	0.175	0.173	0.208	0.169	0.222	0.215	0.179	0.185	0.181	0.187	0.173	0.187	0.165	0.187	0.166	0.167	0.184	0.017
PSOPC	Runtime	1.989	1.883	2.171	2.650	1.885	2.653	2.177	2.233	2.178	2.681	1.883	2.333	2.982	2.979	1.762	2.135	2.286	0.392
	Δx	0.186	0.193	0.174	0.191	0.213	0.196	0.189	0.243	0.196	0.182	0.197	0.214	0.192	0.201	0.197	0.198	0.198	0.016
	Δf	0.144	0.154	0.132	0.144	0.198	0.157	0.137	0.187	0.147	0.142	0.156	0.196	0.154	0.191	0.153	0.149	0.159	0.022
	Δt	0.166	0.175	0.154	0.169	0.206	0.178	0.165	0.217	0.173	0.163	0.178	0.205	0.174	0.196	0.176	0.175	0.179	0.017
NPSO	Runtime	1.934	1.972	2.034	4.167	2.061	2.394	2.035	2.176	2.069	2.732	1.855	2.032	2.982	2.561	1.364	2.272	2.290	0.625
	Δx	0.196	0.203	0.198	0.193	0.214	0.197	0.201	0.199	0.195	0.193	0.223	0.201	0.196	0.206	0.196	0.187	0.200	0.009
	Δf	0.154	0.187	0.157	0.153	0.187	0.156	0.179	0.143	0.149	0.146	0.189	0.167	0.155	0.179	0.163	0.143	0.163	0.017
	Δt	0.176	0.195	0.179	0.174	0.201	0.178	0.190	0.175	0.174	0.171	0.207	0.185	0.177	0.193	0.180	0.166	0.182	0.012
DPSO	Runtime	2.351	2.189	2.097	2.594	2.324	2.287	2.374	2.587	1.863	2.517	1.995	1.446	2.217	1.891	2.406	2.297	2.215	0.301
	Δx	0.193	0.207	0.194	0.191	0.184	0.195	0.183	0.208	0.193	0.185	0.199	0.175	0.195	0.192	0.213	0.185	0.193	0.010
	Δf	0.162	0.183	0.161	0.152	0.162	0.163	0.159	0.184	0.147	0.142	0.183	0.141	0.170	0.156	0.196	0.141	0.163	0.017
	Δt	0.178	0.195	0.178	0.173	0.173	0.181	0.171	0.196	0.172	0.165	0.191	0.163	0.183	0.175	0.202	0.164	0.179	0.013
LS-DF-PSO	Runtime	1.912	2.256	2.184	2.434	2.586	2.025	2.228	2.191	2.326	2.847	1.995	1.857	2.306	1.487	2.534	2.312	2.218	0.322
	Δx	0.188	0.186	0.212	0.195	0.207	0.196	0.187	0.194	0.185	0.196	0.187	0.193	0.216	0.196	0.184	0.191	0.195	0.010
	Δf	0.156	0.143	0.180	0.151	0.183	0.152	0.140	0.151	0.144	0.152	0.141	0.155	0.182	0.152	0.145	0.154	0.155	0.014
	Δt	0.173	0.166	0.197	0.174	0.195	0.175	0.165	0.174	0.166	0.175	0.166	0.175	0.200	0.175	0.166	0.173	0.176	0.011
GEPSO	Runtime	2.086	1.894	1.887	2.029	2.332	1.772	1.715	2.050	2.076	2.129	1.578	1.673	2.009	1.592	2.372	1.951	1.947	0.238
	Δx	0.187	0.153	0.199	0.193	0.198	0.194	0.194	0.185	0.194	0.199	0.179	0.197	0.186	0.197	0.186	0.167	0.188	0.013
	Δf	0.146	0.113	0.151	0.149	0.156	0.158	0.154	0.142	0.147	0.145	0.139	0.155	0.145	0.157	0.144	0.124	0.145	0.012
	Δt	0.168	0.134	0.177	0.172	0.178	0.177	0.175	0.165	0.172	0.174	0.160	0.177	0.167	0.178	0.166	0.147	0.168	0.012

Table 7
Means, standard deviations, and other metrics of the Sphere function for 10,000 runs.

Scenario	Algorithm	Metrics				
		μ	σ	Δx	Δf	Δt
1	GEPSO − adaptive inertia updating − 4th term − 5th term in the velocity updating formula (equivalent to the original PSO)	11.721	12.391	0.192	0.152	0.173
2	GEPSO + adaptive inertia updating − 4th term − 5th term in the velocity updating formula	10.456	11.652	0.191	0.150	0.172
3	GEPSO + adaptive inertia updating + 4th term − 5th term in the velocity updating formula	9.774	9.342	0.189	0.147	0.169
4	GEPSO + adaptive inertia updating + 4th term + 5th term in the velocity updating formula	8.431	8.651	0.187	0.146	0.168

Algorithm	Δf		
	0.205	**0.175**	**0.145**
PSO	5020	5970	6540
REPSO	5430	6150	7240
PSOPC	5230	6100	6940
NPSO	5300	6050	7014
DPSO	5210	6035	7130
LS-DF-PSO	5130	6080	7230
GEPSO	5130	5760	6310

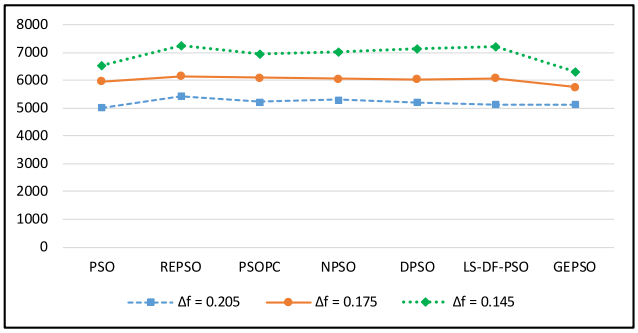


Fig. 4. The number of function evaluations (iterations) for the Sphere function at different levels of Δf.

To evaluate the convergence speed of the GEPSO algorithm, the number of the function evaluations (iterations) are investigated at three different values (levels) of the Associated Normalized Distance (Δf) metric for the Sphere function, as shown in Fig. 4. It can be seen that GEPSO reached the optimal solution faster than other algorithms.

We also investigated the effect of the number of function evaluations that are 10, 100, 1000, and 10,000 on the quality of solutions of all algorithms and functions. In Fig. 5 the mean and standard deviation of the optimal solutions represent the vertical axis and the numbers of fitness function evaluations represent the horizontal axis. Fig. 5 shows how the algorithms did act in various function evaluations for selected benchmarks from 2 and n variables categories. As expected, the higher the numbers of function evaluations, the better are the obtained results for all algorithms.

Analysis of Fig. 5 also reveals that for the selected benchmark problems, no significant improvement could be achieved by further increasing the maximum number of function evaluations above 10 000, and the results with 1000 function evaluations are almost equally satisfactory.

All the plots in Fig. 5 can also be summarized in Figs. 6 and 7 in a different way, where metrics including Δx, Δf and Δt and runtimes (in seconds) of all 16 benchmark functions vs. the number of fitness evaluations (iterations) are depicted. Interestingly, in Fig. 6 the PSOPC significantly improved its results after increasing the function evaluations to 1000, while the GEPSO exhibited a consistent and robust behavior for different numbers of function evaluations. Also, Fig. 7 shows how the algorithms spent computational time in higher function evaluations where the performance of NPSO was particularly slowed down after 10,000 function evaluations whereas the other algorithms had similar time growth patterns.

5. Conclusion

While being inspired by the original PSO algorithm, GEPSO tries to improve the search efficiency and effectiveness through introducing some novelties in the velocity updating equation and an adaptive inertia weight updating mechanism which regulates the diversification and intensification of the search at any time according to

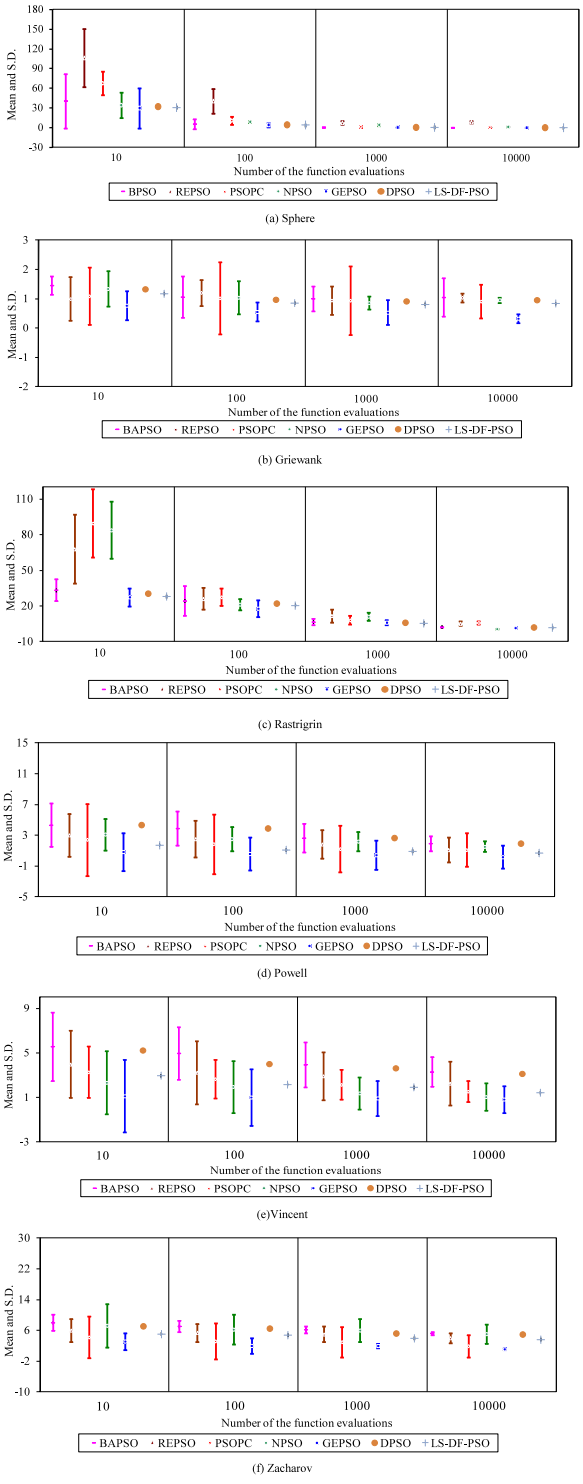
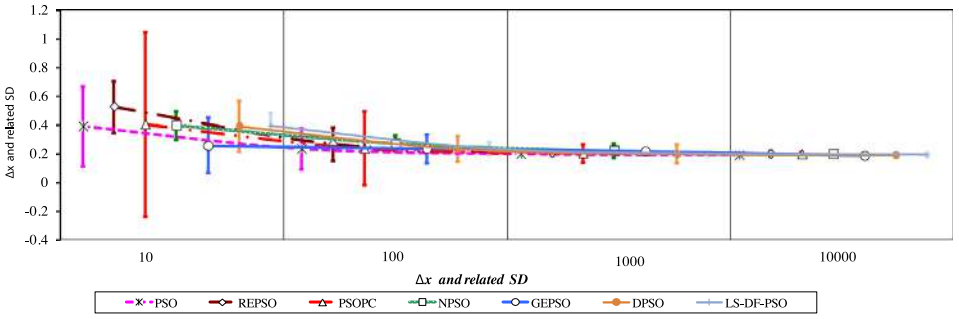
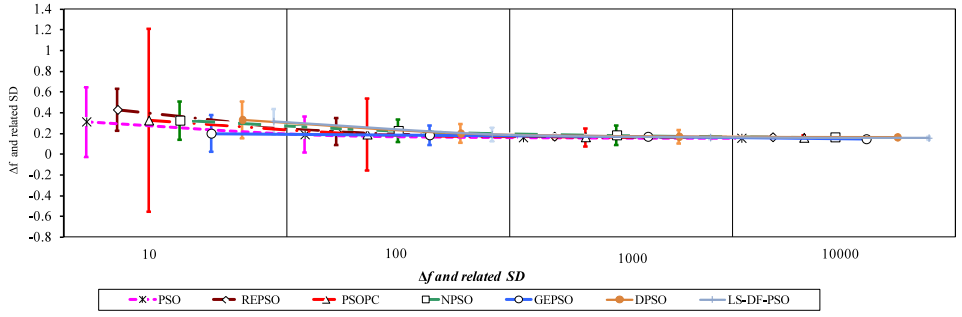


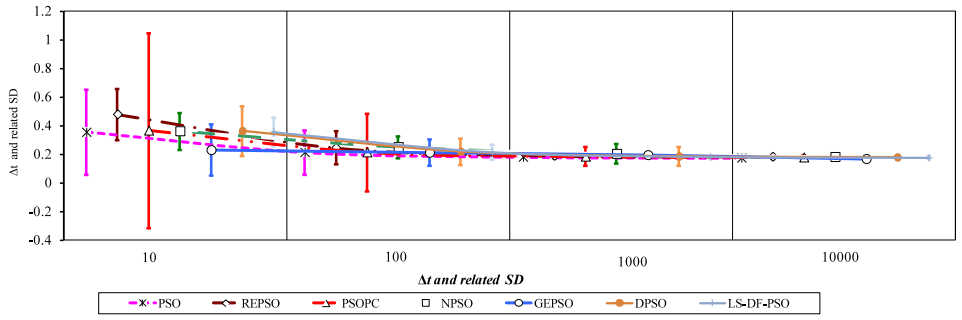
Fig. 5. Means and standard deviations of fitness function values of benchmark problems vs. number of function evaluations (iterations).



(a) Δ*x and* related *SD*



(b) Δ*f and* related *SD*



(c) Δt and related SD

Fig. 6. Comparison of the metrics (Δx, Δf, Δt)values of PSO, REPSO, PSOPC, NPSO, DPSO, LS-DF-PSO and GEPSO in various function evaluations (iterations).

the fitness value in the preceding two iterations. The newly added fourth and fifth terms increase the diversity of the swarm, enhance the interrelation among the particles, as well as the capability of doing a more effective search in complex (e.g., rugged and irregular) search spaces.

According to the computational results, GEPSO came up with better results in the performance metrics of fitness functions, as well as average runtimes for solving 16 standard benchmark continuous minimization problems compared to the PSO, REPSO, PSOPC, NPSO, DPSO, and LS-DF-PSO algorithms. As a result, GEPSO can be considered as an efficient optimization tool for a variety of continuous optimization problems. Our future research is focused on modifying the GEPSO to handle discrete combinatorial problems such as the Traveling Salesman Problem and the non-attacking n-queens problems. Also, dynamic and adaptive tuning mechanisms can be proposed for setting the values of control, acceleration, and convergence parameters as well.

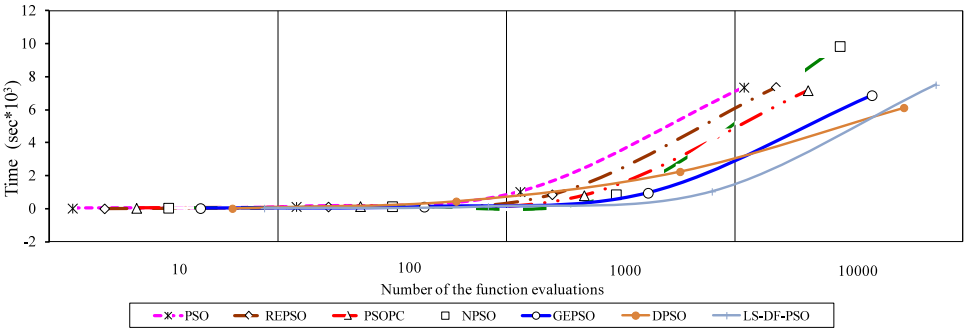


Fig. 7. Comparison of the runtimes of PSO, REPSO, PSOPC, NPSO, DPSO, LS-DF-PSO and GEPSO in various function evaluations (iterations).

References

[1] H. Akbaripour, E. Masehian, Efficient and robust parameter tuning for heuristic algorithms, 2013.
[2] J.A. Alfaro-Ayala, V. Ayala-Ramirez, A. Gallegos-Munoz, A.R. Uribe-Ramirez, Optimal location of axial impellers in a stirred tank applying evolutionary programing and CFD, Chem. Eng. Res. Des. 100 (2015) 203–211.
[3] O.T. Altinoz, A.E. Yilmaz, G.W. Weber, Application of chaos embedded PSO for PID parameter tuning, Int. J. Comput. Commun. Control 7 (2) (2012) 204–217.
[4] T.S. Babu, N. Rajasekar, K. Sangeetha, Modified particle swarm optimization technique based maximum power point tracking for uniform and under partial shading condition, Appl. Soft Comput. 34 (2015) 613–624.
[5] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: background and development, Nat. Comput. 6 (4) (2007) 467–484.
[6] J. Chuanwen, E. Bompard, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation, Math. Comput. Simulation 68 (1) (2005) 57–65.
[7] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Micro Machine and Human Science, 1995. MHS'95. Proceedings of the Sixth International Symposium on, IEEE, 1995, pp. 39–43.
[8] M. El-Abd, Cooperative models of particle swarm optimizers, 2008.
[9] R.S. Faradonbeh, M. Monjezi, D.J. Armaghani, Genetic programing and non-linear multiple regression techniques to predict backbreak in blasting operation, Eng. Comput. 32 (1) (2016) 123–133.
[10] A. GáLvez, A. Iglesias, A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing, Appl. Soft Comput. 13 (3) (2013) 1491–1504.
[11] P. Ghamisi, M.S. Couceiro, F.M. Martins, J.A. Benediktsson, Multilevel image segmentation based on fractional-order Darwinian particle swarm optimization, IEEE Trans. Geosci. Remote Sens. 52 (5) (2014) 2382–2394.
[12] J. Gou, Y.-X. Lei, W.-P. Guo, C. Wang, Y.-Q. Cai, W. Luo, A novel improved particle swarm optimization algorithm based on individual difference evolution, Appl. Soft Comput. 57 (2017) 468–481.
[13] S. He, Q. Wu, J. Wen, J. Saunders, R. Paton, A particle swarm optimizer with passive congregation, Biosystems 78 (1–3) (2004) 135–147.
[14] I. Jain, V.K. Jain, R. Jain, Correlation feature selection based improved-Binary Particle Swarm Optimization for gene selection and cancer classification, Appl. Soft Comput. 62 (2018) 203–215.
[15] A.R. Jordehi, Enhanced leader PSO (ELPSO): a new PSO variant for solving global optimisation problems, Appl. Soft Comput. 26 (2015) 401–417.
[16] E. Keedwell, M. Morley, D. Croft, Continuous trait-based particle swarm optimisation (CTB-PSO), in: International Conference on Swarm Intelligence, Springer, 2012, pp. 342–343.
[17] M.S. Kiran, Particle swarm optimization with a new update mechanism, Appl. Soft Comput. 60 (2017) 670–678.
[18] K.H. Lee, S.W. Baek, K.W. Kim, Inverse radiation analysis using repulsive particle swarm optimization algorithm, Int. J. Heat Mass Transfer 51 (11–12) (2008) 2772–2783.
[19] C.-H. Lee, Y.-C. Lee, F.-Y. Chang, A dynamic fuzzy neural system design via hybridization of EM and PSO algorithms, IAENG Int. J. Comput. Sci. 37 (3) (2010) 242–251.
[20] C. Lin, Y. Liu, C. Lee, An efficient neural fuzzy network based on immune particle swarm optimization for prediction and control applications, Int. J. Innovative Comput. Inf. Control 4 (7) (2008) 1711–1722.
[21] J. Luan, Z. Yao, F. Zhao, X. Song, A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization, Math. Comput. Simulation 156 (2019) 294–309.
[22] K. Mason, J. Duggan, E. Howley, A meta optimisation analysis of particle swarm optimisation velocity update equations for watershed management learning, Appl. Soft Comput. 62 (2018) 148–161.
[23] V. Miranda, N. Fonseca, EPSO-best-of-two-worlds meta-heuristic applied to power system problems, in: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, Vol. 2, IEEE, 2002, pp. 1080–1085.

[24] B.B. Naik, C.P. Raju, R.S. Rao, A Constriction Factor based Particle Swarm Optimization for Congestion Management in Transmission Systems, Int. J. Electr. Eng. Inform. 10 (2) (2018) 232–241.
[25] M. Neethling, A.P. Engelbrecht, Determining RNA secondary structure using set-based particle swarm optimization, in: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, IEEE, 2006, pp. 1670–1677.
[26] S.-H. Ong, W.-J. Lee, Y.-C. Low, A general method of computing mixed Poisson probabilities by Monte Carlo sampling, Math. Comput. Simulation 170 (2020) 98–106.
[27] G. Rozenberg, T. Bäck, J.N. Kok, Handbook of Natural Computing, Springer, 2012.
[28] I. Schoeman, A. Engelbrecht, Using vector operations to identify niches for particle swarm optimization, in: Cybernetics and Intelligent Systems, 2004 IEEE Conference on, Vol. 1, IEEE, 2004, pp. 361–366.
[29] M. Sedighizadeh, M. Esmaili, H. Parvaneh, Coordinated optimization and control of SFCL and SMES for mitigation of SSR using HBB-BC algorithm in a fuzzy framework, J. Energy Storage 18 (2018) 498–508.
[30] D. Sedighizadeh, E. Masehian, Particle swarm optimization methods, taxonomy and applications, Int. J. Comput. Theory Eng. 1 (5) (2009) 486.
[31] A. Serani, M. Diez, Are random coefficients needed in particle swarm optimization for simulation-based ship design, in: Proceedings of the 7th International Conference on Computational Methods in Marine Engineering (Marine 2017), 2017.
[32] A. Serani, M. Diez, E.F. Campana, G. Fasano, D. Peri, U. Iemma, Globally convergent hybridization of particle swarm optimization using line search-based derivative-free techniques, in: Recent Advances in Swarm Intelligence and Evolutionary Computation, Springer, 2015, pp. 25–47.
[33] A. Serani, C. Leotardi, U. Iemma, E.F. Campana, G. Fasano, M. Diez, Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems, Appl. Soft Comput. 49 (2016) 313–334.
[34] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, Vol. 1, IEEE, 2001, pp. 101–106.
[35] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. 38 (2016) 281–295.
[36] E.-G. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, 2009.
[37] Q. Tao, H. Chang, Y. Yi, C. Gu, Y. Yu, QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm, in: Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on, IEEE, 2009, pp. 153–159.
[38] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[39] C. Yang, D. Simon, A new particle swarm optimization technique, in: Null, IEEE, 2005, pp. 164–169.
[40] C. Yu, K.L. Teo, L. Zhang, Y. Bai, A new exact penalty function method for continuous inequality constrained optimization problems, J. Ind. Manag. Optim. 6 (2010) 895–910.
[41] Y. Zhang, D.-w. Gong, N. Geng, X.-y. Sun, Hybrid bare-bones PSO for dynamic economic dispatch with valve-point effects, Appl. Soft Comput. 18 (2014) 248–260.
[42] L. Zhang, Y. Tang, C. Hua, X. Guan, A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques, Appl. Soft Comput. 28 (2015) 138–149.
[43] T. Zheng, M. Yamashiro, Solving flow shop scheduling problems by quantum differential evolutionary algorithm, Int. J. Adv. Manuf. Technol. 49 (5–8) (2010) 643–662.