Particle Swarm Methods

Konstantinos E. Parsopoulos

Contents

K.E. Parsopoulos (✉)
Department of Computer Science & Engineering, University of Ioannina, Ioannina, Greece
e-mail: kostasp@cs.uoi.gr

R. Martí et al. (eds.), *Handbook of Heuristics*,
DOI 10.1007/978-3-319-07153-4_22-1

Abstract

Particle swarm optimization has gained increasing popularity in the past 15 years. Its effectiveness and efficiency has rendered it a valuable metaheuristic approach in various scientific fields where complex optimization problems appear. Its simplicity has made it accessible to the non-expert researchers, while the potential for easy adaptation of operators and integration of new procedures allows its application on a wide variety of problems with diverse characteristics. Additionally, its inherent decentralized nature allows easy parallelization, taking advantage of modern high-performance computer systems. The present work exposes the basic concepts of particle swarm optimization and presents a number of popular variants that opened new research directions by introducing novel ideas in the original model of the algorithm. The focus is placed on presenting the essential information of the algorithms rather than covering all the details. Also, a large number of references and sources is provided for further inquiry. Thus, the present text can serve as a starting point for researchers interested in the development and application of particle swarm optimization and its variants.

Keywords

Particle Swarm Optimization • Swarm Intelligence • Metaheuristics • Nature-Inspired Algorithms • Stochastic Search • Optimization • Computational Intelligence

Introduction

Particle swarm optimization (PSO) was introduced in the pioneering works of Russell C. Eberhart and James Kennedy in [33, 60]. At that time, the wide success of Evolutionary Algorithms (EAs) motivated researchers worldwide to develop and experiment with novel nature-inspired methods. Thus, besides the interest in evolutionary procedures that governed EAs, new paradigms from nature were subjected to investigation. In this context, the hierarchically organized societies of simple organisms such as ants, bees, and fishes, which have limited range of individual responses but fascinating collective behaviors, immediately attracted scientific interest.

Simulations conducted with modeled populations of such organisms exhibited traits of intelligent behavior and remarkable problem solving capabilities [12]. The underlying mathematical models shared concepts with particle physics, enriched with elements from probability theory and stochastic processes. The theoretical background along with the potential for developing powerful optimization metaheuristics resulted in the emergence of a new category of algorithms under the name of *Swarm Intelligence* (SI) [12, 34, 62].

PSO is placed in a salient position among SI algorithms. Its inspiration stemmed from simulators of social behavior that implement rules such as neighbor velocity matching and acceleration by distance. These properties were shown to produce swarming motives in groups of simple artificial agents. The early models were prop-

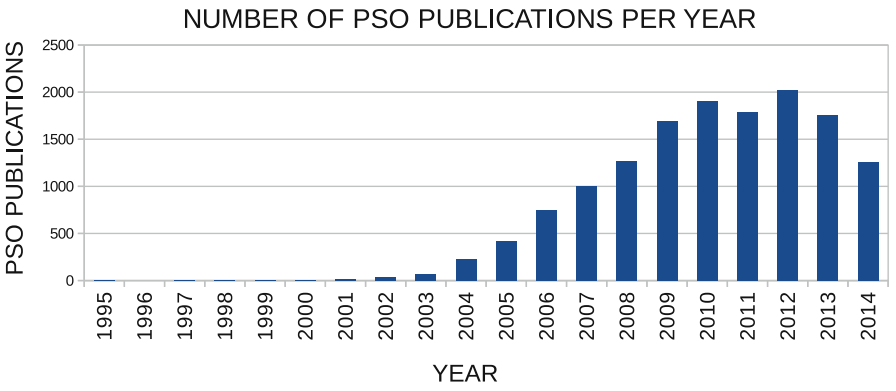

Fig. 1 Number of publications with the term "particle swarm" in the title (Source: Scopus search engine, November 2014)

erly refined to fit the framework of optimization algorithms. The first PSO models introduced the novelty of using difference vectors among population members to sample new points in the search space. This novelty diverged from the established procedures of EAs, which were mostly based on sampling new points from explicit probability distributions [126]. Nevertheless, it proved to be appealing and, almost concurrently with PSO, another algorithm with similar structure, namely, Differential Evolution [131], appeared in the literature. Additional advantages of PSO were its potential for easy adaptation of operators and procedures to match the specific requirements of a given problem, as well as its inherent decentralized structure that promoted parallelization.

PSO has gained wide recognition due to its effectiveness, efficiency, and easy implementation. This is illustrated in Fig. 1, which reports the number of scientific works that include the term "Particle Swarm" in the titles for the years 1995–2014, as returned by the Scopus search engine on a search conducted in November 2014. The illustrated numbers of papers include all sources provided by the Scopus search engine (journals, conferences, books, etc.).

The present work aims at introducing the basic concepts of PSO and presenting a number of its most influential variants, based on the criteria of novelty at the time of development, popularity, and potential for opening new research directions. Naturally, neither the selection of variants can be complete nor the presentation can be thorough within the limited space of a book chapter. For this reason, only basic information is provided, while further details can be acquired in the reported references and sources. Thus, the present text can serve as the "Ariadne's thread" for researchers with interest in PSO.

The rest of the present chapter is organized as follows: section "Basic Model" introduces a basic PSO model that is used as reference point for the presented variants. Section "Convergence and Parameter Setting" outlines the theoretical properties of PSO and probes interesting aspects such as convergence, parameter setting, special features of the algorithm, as well as performance-enhancing techniques. Section "Enhanced and Specialized PSO Variants" presents a number

of enhanced and specialized PSO variants that constitute the state-of-the-art, while section "Applications" briefly comments on applications. Finally, the paper concludes in section "Conclusions". The Appendix at the end of the text offers additional sources for further inquiry and experimentation with PSO.

Basic Model

PSO was primarily developed for optimization in real-valued search spaces. For this reason, the general d-dimensional bound-constrained optimization problem,

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{1}$$

where,

$$\mathcal{X} = \left[x_1^{\min}, x_1^{\max}\right] \times \cdots \times \left[x_d^{\min}, x_d^{\max}\right] \subset \mathbb{R}^d, \tag{2}$$

is henceforth considered as the reference problem for the presentation of the algorithm. In case of different search spaces, explicit definitions will be given. The only necessary assumption regarding the objective function f is the availability of its value $f(\mathbf{x})$ for every point $\mathbf{x} \in \mathcal{X}$. Smoothness properties such as continuity and differentiability are not required.

The main search mechanism of PSO is based on a group of *search agents*, called *particles*, which iteratively change their position in the search space $\mathcal{X}$. The particles retain in an external memory the best positions they have ever visited in $\mathcal{X}$. The move of each particle is stochastically biased toward its own findings as well as promising regions of the search space discovered by the rest of the particles. This implicit information-exchange scheme is responsible for the emergent convergence properties of the whole group, which is accordingly called a *swarm*.

Putting it formally, let A_i denote the i-th particle (search agent) and,

$$\mathcal{S} = \{A_1, A_2, \ldots, A_N\}, \tag{3}$$

be a swarm of size N (the ordering of the particles is irrelevant). Also, let,

$$I = \{1, 2, \ldots, N\}, \qquad D = \{1, 2, \ldots, d\},$$

be the sets of indices of the particles and the coordinate directions, respectively, and t denote the algorithm's iteration counter (this notation will be henceforth used in the present work). Then, each particle can be defined by four essential elements,

$$A_i^{(t)} = \left\langle \mathbf{x}_i^{(t)}, \mathbf{v}_i^{(t)}, \mathbf{p}_i^{(t)}, NB_i^{(t)} \right\rangle, \qquad i \in I.$$

The first vector,

$$\mathbf{x}_i^{(t)} = \left(x_{i1}^{(t)}, x_{i2}^{(t)}, \ldots, x_{id}^{(t)}\right)^\top \in \mathcal{X},$$

defines the *current position* of the particle in $\mathcal{X}$ at iteration t. In many works in literature, the term "particle" is used to define solely the current position $\mathbf{x}_i$. However, in the author's opinion, the use of this term to define the search agent with all its components promotes a more compact presentation, and it is aligned with notation in similar SI algorithms (e.g., the term "ant" is similarly used to define search agents in ant colony optimization [12]). The second vector,

$$\mathbf{v}_i^{(t)} = \left(v_{i1}^{(t)}, v_{i2}^{(t)}, \ldots, v_{id}^{(t)}\right)^\top,$$

is an adaptable position shift, also called *velocity*, which is responsible for the particle's move. The third vector,

$$\mathbf{p}_i^{(t)} = \left(p_{i1}^{(t)}, p_{i2}^{(t)}, \ldots, p_{id}^{(t)}\right)^\top \in \mathcal{X},$$

is the particle's *best position*, i.e., the best position it has ever visited in $\mathcal{X}$ up to iteration t. For the reference optimization problem of Eq. (1), the best position corresponds to the particle's previous position with the smallest objective value, i.e.,

$$\mathbf{p}_i^{(t)} = \arg \min_{\left\{\mathbf{x}_i^{(k)},\, k \leqslant t\right\}} f\left(\mathbf{x}_i^{(k)}\right).$$

The best position plays the role of attractor that biases the velocity toward the discovered promising regions of $\mathcal{X}$. However, this sole information would render the particle an isolated agent with limited search capability that produces trajectories by oscillating around its best visited points.

For this reason, in addition to its own discoveries, each particle has an implicit information-exchange mechanism with a subset of the swarm, called its *neighborhood*. The neighborhood can be formally denoted as a subset of the indices set I,

$$NB_i^{(t)} \subseteq I,$$

and its cardinality is usually called the *neighborhood size*. In most PSO variants, the best position in the neighborhood of the particle, i.e.,

$$\mathbf{p}_{g_i}^{(t)} = \arg \min_{\left\{\mathbf{p}_j^{(t)},\, j \in NB_i^{(t)}\right\}} f\left(\mathbf{p}_j^{(t)}\right),$$

is used as a second attractor for biasing its velocity.

Based on the definitions above, at each iteration of the algorithm, the particles componentwisely update their current positions and velocities as follows:

$$v_{ij}^{(t+1)} = \chi\, v_{ij}^{(t)} + \mathcal{C}_1\left(p_{ij}^{(t)} - x_{ij}^{(t)}\right) + \mathcal{C}_2\left(p_{g_i j}^{(t)} - x_{ij}^{(t)}\right), \tag{4}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{5}$$

Algorithm 1: Canonical PSO

Require: Initialize PSO algorithm.

```
while (not stopping condition) do
    for i = 1 to N do
        for j = 1 to d do
            Update v_ij by using Eq. (4).
            Check for velocity boundaries violation and correct if needed.
            Update x_ij using Eq. (5).
            Check for search space boundaries violation and correct if needed.
        end for
    end for
    for i = 1 to N do
        Update p_i according to Eq. (7).
    end for
end while
Report best solution.
```

where $i \in I$ and $j \in D$. The scalar parameter χ is called the *inertia weight* or *constriction coefficient*, and its role is discussed in section "Convergence and Parameter Setting". The stochastic terms $\mathcal{C}_1$ and $\mathcal{C}_2$ follow continuous uniform distributions,

$$\mathcal{C}_1 \sim \mathcal{U}(0, c_1), \qquad \mathcal{C}_2 \sim \mathcal{U}(0, c_2), \tag{6}$$

where c_1 and c_2 are user-defined parameters, called the *acceleration constants*, which control the magnitude of attraction toward $\mathbf{p}_i^{(t)}$ and $\mathbf{p}_{g_i}^{(t)}$, respectively. The first difference term $\left(p_{ij}^{(t)} - x_{ij}^{(t)}\right)$ is called the *cognitive term* because it involves only the particle's own information. Correspondingly, the second difference term $\left(p_{g_i j}^{(t)} - x_{ij}^{(t)}\right)$ is called the *social term* since it involves information provided by other particles.

After the current positions and velocities, the best positions are also updated as follows:

$$\mathbf{p}_i^{(t+1)} = \begin{cases} \mathbf{x}_i^{(t+1)}, & \text{if } f\left(\mathbf{x}_i^{(t+1)}\right) \leq f\left(\mathbf{p}_i^{(t)}\right), \\ \mathbf{p}_i^{(t)}, & \text{otherwise,} \end{cases} \qquad i \in I. \tag{7}$$

The algorithm is executed until a predefined stopping condition is fulfilled. The presented basic PSO model is also known as the *Canonical Particle Swarm* [59], and it is summarized in Algorithm 1. The discussion on initialization, boundary violation, and stopping conditions is postponed until the next section.

A number of issues arise regarding essential decisions that need to be made prior to the application of the basic PSO algorithm. Specifically, the practitioner shall address the following questions:

1. How shall the parameters N, χ, c_1, c_2, be set?
2. How shall the neighborhoods be defined?
3. How shall the boundaries violations be handled?
4. How shall the swarm be initialized?
5. What stopping conditions shall be used?

Apparently, each decision can have significant impact on the algorithm's performance. For this reason, careful settings that take into consideration the potential impact on PSO's dynamic are of vital importance.

A number of PSO variants have been developed on the basis of using alternative techniques for configuring the algorithm. These issues are discussed in the next section. Canonical PSO has constituted the starting point for further developments and enhancements for almost two decades. Yet, it still remains a popular piece of the state-of-the-art due to its verified effectiveness and efficiency in a plethora of applications [106, 112, 127].

Convergence and Parameter Setting

During the last two decades of PSO's development, a large number of variants have been proposed. In almost all cases, the motivation for further research has stemmed from the necessity for more efficient algorithms that tackle the weaknesses of previous variants. In the core of these developments lie the answers of the basic questions posed in section "Basic Model" with respect to PSO's configuration and parameter setting.

In the following paragraphs, the most significant developments are highlighted along with insights and suggestions for PSO's configuration.

Early Precursors

The early PSO precursors [33, 60] were not as sophisticated as the Canonical PSO approach presented in section "Basic Model". In fact, they were based on a simplified version of Eq. (4) with $\chi = 1$ and $NB_i = I$, i.e.,

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + \mathcal{C}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \mathcal{C}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right), \tag{8}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{9}$$

with $i \in I$, $j \in D$, and $\mathbf{p}_g^{(t)}$ be the best position discovered by any particle (also called the swarm's *overall best*) up to iteration t, i.e.,

$$\mathbf{p}_g^{(t)} = \arg \min_{\left\{\mathbf{p}_k^{(t)},\, k \in I\right\}} f\left(\mathbf{p}_k^{(t)}\right). \tag{10}$$

Obviously, the magnitude of the position shifts depends on the values of the parameters c_1 and c_2 that determine the stochastic terms $\mathcal{C}_1$ and $\mathcal{C}_2$ according to Eq. (6).

In [56] the trajectories of the particles were studied by simplifying the system. The use of the previous velocity $\mathbf{v}_i^{(t)}$ in the update equation of the current velocity results in an oscillatory move of the particle around the weighted average of the two best positions,

$$\bar{\mathbf{p}}_i^{(t)} = \frac{1}{\mathcal{C}_1 + \mathcal{C}_1}\left(\mathcal{C}_1 \mathbf{p}_i^{(t)} + \mathcal{C}_2 \mathbf{p}_g^{(t)}\right). \tag{11}$$

The swarm's overall best particle, for which it holds that $\mathbf{p}_i^{(t)} = \mathbf{p}_g^{(t)}$, updates its velocity as follows:

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + \mathcal{C}\left(p_j^{(t)} - x_{ij}^{(t)}\right), \tag{12}$$

where $j \in D$, $\mathcal{C} = \mathcal{C}_1 + \mathcal{C}_2$, and $\mathbf{p}^{(t)} = \mathbf{p}_g^{(t)}$.

If we consider the trivial case of a single particle and a fixed best position $\mathbf{p}$, it was shown in [56] that the particle's trajectory becomes highly dependent on the values of $\mathcal{C}$, as shown in Fig. 2. Values higher than $\mathcal{C} = 4.0$ are detrimental for the algorithm's convergence, since the velocity can grow arbitrarily large (notice the scaling difference in the vertical axis for $\mathcal{C} = 4.0$ in Fig. 2). This problem was called the *swarm explosion effect*, and it was verified also in [87, 88], emphasizing the necessity for a mechanism to control the amplitude of the velocities.

Velocity Clamping and Inertia Weight

The swarm explosion effect led to the first significant improvements of PSO, namely *velocity clamping* and the introduction of the *inertia weight*. Since the main problem was the arbitrary growth of the velocities, a straightforward solution was their explicit restriction in acceptable bounds. Specifically, a maximum value $v_j^{\max}$ was imposed on the absolute value of each velocity component, i.e.,

$$-v_j^{\max} \leqslant v_{ij} \leqslant v_j^{\max}, \quad \text{for all } i \in I, j \in D.$$

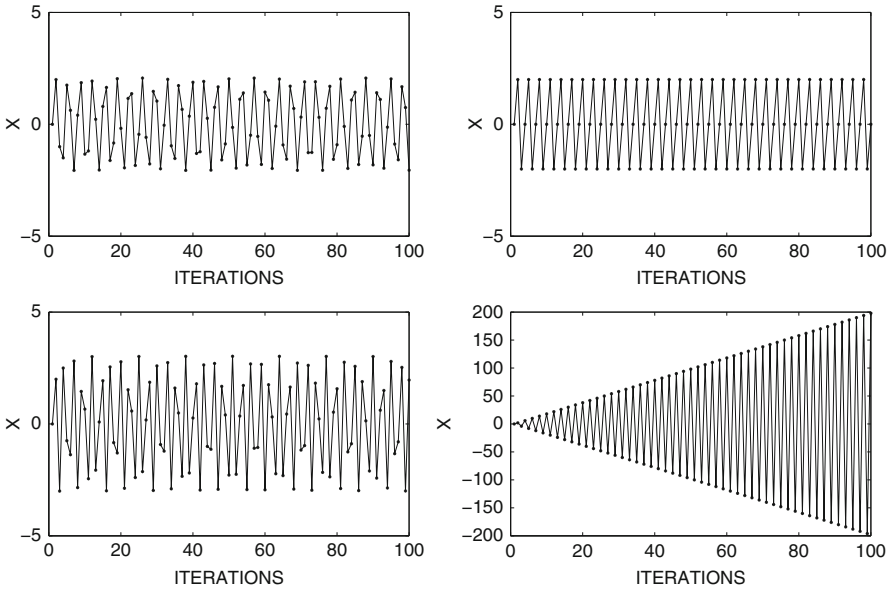

Fig. 2 Particle trajectory for fixed value $\mathcal{C} = 2.5$ (*upper left*), 3.0 (*upper right*), 3.5 (*lower left*), and 4.0 (*lower right*)

This way, if a component v_{ij} violates one of the boundaries, it is set equal to the violated boundary's value, prohibiting the particle from taking large steps away from the weighted average of best positions of Eq. (11). The user-defined value $v_j^{\max}$ is usually equal to a fraction of the search space along the j-th coordinate direction, i.e.,

$$v_j^{\max} = \lambda_j \left(x_j^{\max} - x_j^{\min} \right), \qquad j \in D, \ \lambda_j \in (0, 1). \tag{13}$$

Naturally, prior information regarding the search space facilitates proper setting of the maximum velocity. For example, in cases of extremely large number of minimizers or very narrow regions of attraction around them, smaller velocities can offer better search accuracy [129]. Equation (13) is used also in modern PSO variants, usually assuming identical λ_j values for all $j \in D$. Velocity clamping is applied in line 5 of the Canonical PSO in Algorithm 1.

Although velocity clamping was effective in hindering divergence, it was proved to be inadequate to produce convergent behavior of the particles. Obviously, convergence to a point in the search space requires the gradual decrease of velocities such that the particles can perform decreasing oscillations around the attractors and, eventually, settle on a point. This was achieved by introducing an inertia weight w [128] on the velocity update of Eq. (8), i.e.,

$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + \mathcal{C}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \mathcal{C}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right), \tag{14}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}. \tag{15}$$

The parameter w shall be properly set such that the impact of the previous velocity $\mathbf{v}_i^{(t)}$ declines during the execution of the algorithm. This can be achieved by setting either a fixed value $w \in (0, 1)$ or assuming a decreasing value between two extreme values $w_{\max}$ and $w_{\min}$, during the algorithm's run. For example, if $t_{\max}$ is a predefined maximum number of iterations, the most common linearly decreasing inertia weight takes values as follows,

$$w^{(t)} = w_{\max} - \frac{t}{t_{\max}} \left(w_{\max} - w_{\min} \right).$$

Figure 3 illustrates the swarm's diversity in terms of the average of the particles' standard deviations per coordinate direction, without inertia weight as well as with a linearly decreasing inertia weight, for a well known optimization problem.

Velocity clamping and the introduction of inertia weight boosted PSO research due to the improved convergence properties of the derived PSO variants. Both these improvements are still used in modern PSO approaches [106]. Further information and enhancements can be found in recent works such as [15, 155].

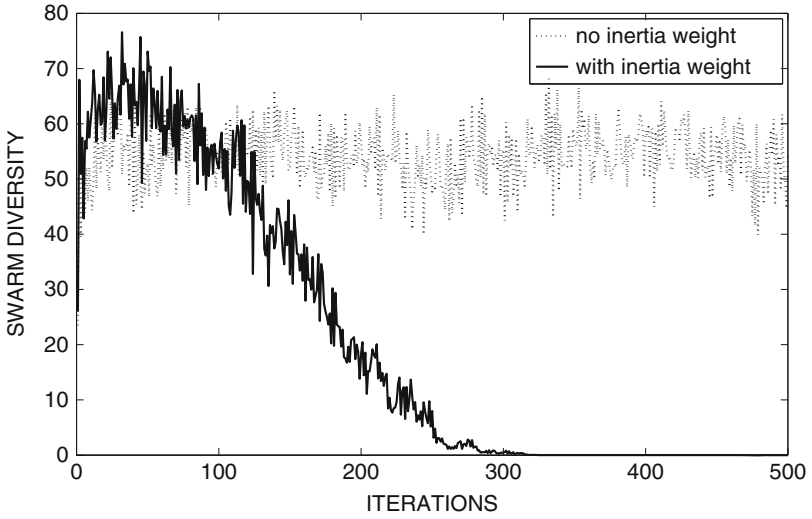

Fig. 3 Swarm's diversity with (*solid line*) and without (*dotted line*) inertia weight while minimizing the 2-dimensional Rosenbrock function using 20 particles with $c_1 = c_2 = 2$, $v^{\max} = 50$, $w_{\max} = 1.2$, and $w_{\min} = 0.1$

Stability Analysis

The empirical analyses of particles' trajectories in [56,87,88] and their implications in PSO's modeling and parameter setting motivated the systematic theoretical investigation of the algorithm. The first breakthrough appeared in 2002 due to M. Clerc and J. Kennedy who conducted a convergence and stability analysis of the algorithm in multidimensional search spaces [21].

The authors used as starting point the early model of Eqs. (8) and (9) with velocity clamping. The initial model was manipulated according to Eq. (12) and further simplified by assuming 1-dimensional particles without stochasticity, i.e., the two stochastic acceleration terms were assumed to be equal and fixed, $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$. After dropping some indices and making proper algebraic manipulations, the 1-dimensional swarm's update rules can be rewritten in the form [21],

$$v^{(t+1)} = v^{(t)} + \mathcal{C}\, y^{(t)}, \tag{16}$$

$$y^{(t+1)} = -v^{(t)} + (1 - \mathcal{C})\, y^{(t)}, \tag{17}$$

where $y^{(t)} = \bar{p} - x^{(t)}$, with $\bar{p}$ being the (currently fixed) aggregate best position defined in Eq. (11) (note that all vectors currently collapse to scalars). This discrete system can be written also in matrix form,

$$P_{t+1} = M\ P_t = M^t\ P_0,$$

where,

$$P_t = \begin{pmatrix} v^{(t)} \\ y^{(t)} \end{pmatrix}, \qquad M = \begin{pmatrix} 1 & \mathcal{C} \\ -1 & 1 - \mathcal{C} \end{pmatrix}.$$

Then, the behavior of the system depends on the eigenvalues of M, which are given by,

$$\lambda_{1,2} = 1 - \frac{\mathcal{C}}{2} \pm \frac{\sqrt{\mathcal{C}^2 - 4\mathcal{C}}}{2}.$$

The previously identified critical value $\mathcal{C} = 4$ (recall Fig. 2) [56,88], appears again as the limit between the case of two different real eigenvalues, one eigenvalue of multiplicity 2, and two complex conjugate eigenvalues $\lambda_{1,2}$.

For the case of $\mathcal{C} \in (0, 4)$, the eigenvalues become complex [21],

$$\lambda_1^t = \cos(t\theta) + i\ \sin(t\theta), \qquad \lambda_2^t = \cos(t\theta) - i\ \sin(t\theta),$$

and the system exhibits cyclic behavior for $\theta = (2\kappa\pi)/t$. On the other hand, values of $\mathcal{C} > 4$ produce no cyclic behavior and it is proved that P_t has monotonically increasing distance from the origin [21]. Finally, the limit case $\mathcal{C} = 4$ produces

either oscillatory behavior, i.e., $P_{t+1} = -P_t$, if P_0 is an eigenvector of M, or linearly increasing or decreasing $\|P_t\|$ for $y_0 > 0$ or $y_0 < 0$, respectively.

The investigation was further extended to the continuous case by transforming the simplified model into the recurrent equation [21],

$$\mathbf{v}^{(t+2)} + (\mathcal{C} - 2)\mathbf{v}^{(t+1)} + \mathbf{v}^{(t)} = 0,$$

which in turn becomes a second-order differential equation,

$$\frac{\partial^2 \mathbf{v}}{\partial t^2} + \ln(\lambda_1\lambda_2)\frac{\partial \mathbf{v}}{\partial t} + \ln(\lambda_1)\ln(\lambda_2)\mathbf{v} = 0,$$

where λ_1 and λ_2 are the solutions of the polynomial,

$$\lambda^2 + (\mathcal{C} - 2)\lambda + 1 = 0.$$

Thus, the quantities $\mathbf{v}^{(t)}$ and $\mathbf{y}^{(t)}$ assume the general form [21],

$$\begin{aligned}\mathbf{v}^{(t)} &= c_1\lambda_1^t + c_2\lambda_2^t,\\ \mathbf{y}^{(t)} &= \left(c_1\lambda_1^t(\lambda_1 - 1) + c_2\lambda_2^t(\lambda_2 - 1)\right)/\mathcal{C}.\end{aligned}$$

The parameters c_1 and c_2 depend on the initial vectors $\mathbf{v}^{(0)}$ and $\mathbf{y}^{(0)}$. Similar analysis with the discrete case shows that the system's explosion depends on whether the condition,

$$\max\{|\lambda_1|, |\lambda_2|\} > 1,$$

holds or not [21].

After the analysis above, the employed simplified model was generalized in order to approximate the actual model of PSO. A number of extended models were accordingly developed and studied in [21]. The outcome of the study was an effective model, namely the Canonical PSO model of Eqs. (4) and (5), accompanied with a proposed parameter setting [21],

$$\chi = 0.729, \quad c_1 = c_2 = 1.49, \tag{18}$$

derived from closed-form formulae in order to retain the convergent behavior of the system. This parameter setting has become the most common choice for off-the-shelf PSO approaches in numerous applications [106].

The first theoretically sound convergence analysis of PSO in [21] was succeeded by a number of theoretical studies [49, 54, 113, 124, 143, 147]. These developments increased our understanding of PSO's dynamic. Also, they added merit that, along with its easy implementation and the reported excellent results in various applications, eventually placed PSO in a salient position among other

established population-based optimization algorithms such as Genetic Algorithms and Evolution Strategies. Moreover, the dynamic of the Canonical PSO model motivated recent approaches that spare function evaluations by taking advantage of the particles' dynamic [148].

Concept of Neighborhood

Information exchange among the particles is a concept of major importance in PSO. In the Canonical PSO of Eqs. (4) and (5), the i-th particle adopts the best position $\mathbf{p}_{g_i}$ of its neighborhood NB_i as an attractor for its move, besides its own best position. This way, the neighborhood determines the communication channels among particles and, consequently, the information flow within the swarm. Without communication, collective behavior cannot emerge and the swarm is downgraded to a group of isolated search agents with limited search capabilities.

Obviously, the neighborhood's characteristics have direct impact on PSO's dynamic [57, 132]. Its structure defines the communication channels among particles, while its size controls the influence of the swarm on each particle. Since the particles are attracted toward the best positions of their neighbors, it is easily comprehended that neighborhoods with large number of particles and dense communication channels are more inclined toward intensification of search around the best detected positions in the search space. This is ascribed to the rapid diffusion of the discovered good solutions to the particles and their consequent attraction toward them.

However, this also renders the particles prone to get stuck in deep local minima that are possibly detected in early steps of the algorithm's execution. Such minima are typically updated less frequently than shallow ones. Hence, they can impose stronger attraction on the particles. On the other hand, less crowded neighborhoods with sparse connections among particles exhibit slower information diffusion in the swarm. In this case, the particles are gradually acquainted with good solutions, retaining higher diversity and exploration capability.

The discussion above suggests that neighborhood's configuration is highly responsible for the diversification/intensification (a.k.a. exploration/exploitation) trade-off of the algorithm. For this reason, it shall be carefully selected. To this end, prior knowledge on the studied problem can be valuable. For example, smooth functions with one or few minima can be properly handled through intensification-oriented neighborhoods. On the other hand, rugged landscapes with a plethora of minimizers dispersed at distant parts of the search space usually require exploration-oriented approaches.

In early PSO variants, each particle was assumed to be connected with all the rest, i.e.,

$$NB_i^{(t)} = I, \quad \text{for all } \; i \in I,$$

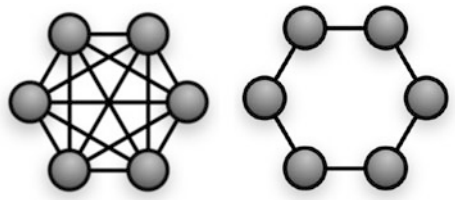

Fig. 4 Graphical representation of the fully connected (*left*) and the ring (*right*) neighborhood topology

and the best position for all neighborhoods was the overall best of the swarm, as defined in Eq. (10). Also, the neighborhoods were time-invariant, i.e., they remained unchanged throughout the execution of the algorithm (hence we can neglect t in $NB_i^{(t)}$ notation). This PSO model is also called the *global PSO model* or simply the *gbest model*. The connection scheme among the particles can be elegantly represented by undirected graphs, where nodes denote the particles and edges denote communication channels. The corresponding structure is called the *neighborhood's topology*. Apparently, the gbest model corresponds to a fully connected graph since all particles communicate with each other. This topology is illustrated in the left part of Fig. 4.

The tendency of the best model to rapidly converge toward the most promising detected solutions and easily get stuck in local minima led to further experimentation with sparsely connected models. The outcome of these efforts was the *local PSO model* or *lbest model*, which assumes neighborhoods that are proper subsets of I, i.e.,

$$NB_i \subset I, \quad \text{for all } \; i \in I.$$

The most effective and easily implemented topology of this form is the *ring*, which is depicted in the right part of Fig. 4. In this scheme, the particles are assumed to lie on a ring according to their indices, i.e, neighboring particles have neighboring indices. Then, each particle is set to communicate only with its immediate neighbors on the ring. The number of neighbors for each direction (front and back) is called the *neighborhood's radius*. Thus, a neighborhood of radius $r < N$ of the i-th particle is defined as the set,

$$NB_i = \{i - r, i - r + 1, \ldots, i - 1, i, i + 1, \ldots, i + r - 1, i + r\},$$

where the indices are assumed to recycle after index N, i.e.,

$$j = \begin{cases} j, & \text{if } \; 1 \leqslant j \leqslant N, \\ j \bmod N, & \text{if } \; j > N, \\ N - |j|, & \text{if } \; j < 1, \end{cases} \qquad \text{for all } \; j \in NB_i.$$

The diversity differences between the gbest model and the lbest with ring topology of radius $k = 1$ is indicatively depicted in Fig. 5 for a simple run on a well-known optimization test problem.

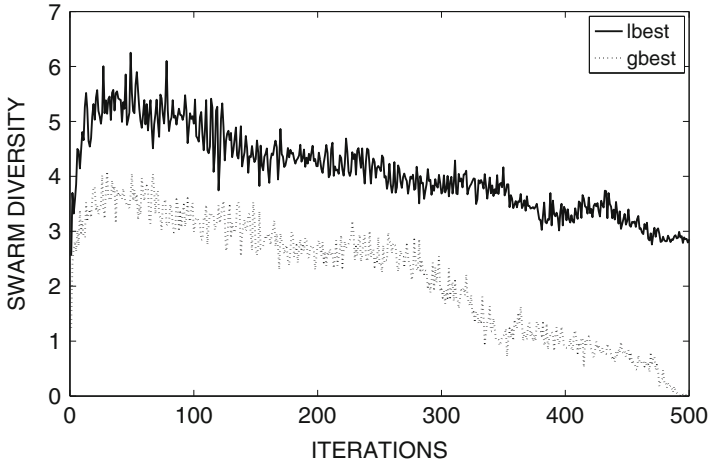

Fig. 5 Swarm's diversity for the lbest (*solid line*) and gbest (*dotted line*) PSO model on the 10-dimensional Rastrigin function using 20 particles

The ring topology has become a standard for lbest PSO implementations due to its successful application in various problems [106]. There is also a multitude of alternative topologies that have drawn attention, such as random [132], hierarchical [46], and dynamic [18], but with limited number of applications. Also, the concept of neighborhood has recently offered the ground for the development of new PSO variants with sophisticated computational budget allocation based on information carried by neighborhoods rather than the particles [130].

Initialization, Stopping Conditions, and Boundaries Violation

A basic requirement in the design and development of stochastic optimization algorithms such as PSO is their tolerance on perturbations of the initial conditions. In practice, this implies that mild perturbations on the initial positions of the particles shall correspond to similar performance profiles of the algorithm.

In benchmarking studies, it is commonly assumed that there is no information available regarding the optimization problem at hand. This is usually called the *black-box optimization problem*. In such cases, there is no reason for purposely biasing the particles' initialization in specific areas of the search space. Consequently, random and uniform initialization of the particles in the search space is the typical procedure followed in most studies, i.e.,

$$x_{ij}^{(0)} = p_{ij}^{(0)} \sim \mathcal{U}\left(x_j^{\min}, x_j^{\max}\right), \qquad \text{for all } \; i \in I, j \in D.$$

Accordingly, the velocities are randomly initialized as,

$$v_{ij}^{(0)} \sim \mathcal{U}\left(-v_j^{\max}, v_j^{\max}\right), \qquad \text{for all } \; i \in I, j \in D.$$

Naturally, if there is prior information regarding promising regions of the search space, the distributions can be properly adapted to favor the sampling of particles in these regions.

The effect of initialization has been studied in a number of works for different PSO variants [25, 31, 36, 100, 137, 157], offering further insight and suggestions in specific applications. Nevertheless, random initialization remains the standard approach also due to its minor implementation effort and the availability of uniform random number generators in almost all hardware platforms.

In contrast to initialization, which is based on the problem's characteristics, the termination conditions are rather user- and resources-dependent [106]. The following are the most common termination criteria:

1. Convergence in search space.
2. Convergence in function values.
3. Limitations in computational budget.
4. Search stagnation.

The first two criteria entail information on the position of the global minimizer or its value, respectively. Naturally, this information is generally unavailable. However, in some cases there are estimated bounds for the solution (or its value) and the user can set the algorithm to stop as soon as it reaches these bounds with a prespecified tolerance. In this case, the underlying stopping condition takes the form,

$$\texttt{IF} \quad \left(\left\|\mathbf{p}_g^{(t)} - \mathbf{x}^*\right\| \leqslant \varepsilon_x \quad \texttt{OR} \quad \left|f_g^{(t)} - f^*\right| \leqslant \varepsilon_f\right) \texttt{ THEN STOP}$$

where $\mathbf{x}^*$ and f^* are the targets in the search space and function values, respectively, and ε_x, ε_f, are the corresponding user-defined tolerances.

The other two stopping criteria are more common in PSO's literature. Computational budget limitations are typically imposed by the maximum available time that can be spent for solving a problem. This quantity can be explicitly expressed either as wall-clock/CPU time or as the number of function evaluations performed by the algorithm. For benchmarking purposes, the latter is preferable since even the same algorithm, executed on the same machine at different time instances, may result in different running times due to irrelevant procedures that may be concurrently executed on the machine.

On the other hand, the search stagnation criterion can prematurely stop the algorithm even if the computational budget is not exceeded. Successful application of this criterion is based on the existence of a proper stagnation measure. Typically, the number of subsequent iterations without improvement of the best solution and/or

the dispersion of the particles' current (or best) positions in the search space have been used as indicators of search stagnation.

Frequently, the aforementioned termination criteria are combined in forms such as,

$$\texttt{IF} \quad \left(t \geqslant t_{\max} \quad \texttt{OR} \quad t_{\text{fail}} \geqslant t_{\text{maxfail}} \quad \texttt{OR} \quad d\,v^{(t)} < d\,v_{\min}\right) \texttt{THEN STOP}$$

where t stands for the iteration number, t_{fail} is the number of subsequent non-improving iterations, and $d\,v^{(t)}$ is a measure of dispersion of the particles (e.g., the average standard deviation per coordinate component). Moreover, the user shall pay special attention to the selection of stopping criteria in order to avoid unintentional premature termination of the algorithm. Recent developments on this issue can be found in [67].

Another topic of interest is the handling of search space boundaries violations. Specifically, after the update of a particle's current position with Eqs. (4) and (5), there is a possibility that some of the new position's components violate the corresponding boundaries of the search space. The most common approach to restrict the particle in the search space is to set the violated components of the particle equal to the value of the violated boundary, i.e.,

$$x_{ij}^{(t)} = \begin{cases} x_j^{\max}, & \text{if } x_{ij}^{(t)} > x_j^{\max}, \\ x_j^{\min}, & \text{if } x_{ij}^{(t)} < x_j^{\min}, \\ x_{ij}^{(t)}, & \text{otherwise}, \end{cases} \tag{19}$$

while simultaneously setting the corresponding velocity component $v_{ij}^{(t)}$ to zero. Different boundary handling techniques (absorbing, bouncing, cyclic search spaces) have been proposed, although with less popularity. A recent survey on such techniques can be found in [89].

Performance-Enhancing Techniques

The Canonical PSO has offered satisfactory performance in various problems. However, it may exhibit declining performance in special problems such as the detection of multiple local/global minima, constrained, or discrete problems. In such cases, the user can either change the algorithm by introducing new, specialized operators or incorporate external techniques to tackle the problem's peculiarities.

There are established techniques that have been successfully used with Canonical PSO. Transformations of the objective function have been used for the alleviation of local minima and the detection of multiple minimizers. Rounding has been used for solving discrete optimization problems, while penalty functions can address constrained optimization problems. In the following paragraphs, basic techniques that have been combined with the Canonical PSO model are presented. Specialized

variants of the algorithm with ad hoc operators for similar purposes are presented in subsequent sections.

Alleviating Local Minimizers

In problems with a multitude of local and/or global minimizers, stochastic optimization algorithms such as PSO may approximate a different minimizer in each independent run. Frequently, the detected solutions are sub-optimal, while the user either needs more than one such solution or requires the globally best one. *Multistart techniques* where the algorithm is subsequently restarted from different initial conditions have been proposed for these cases and discussed in classical optimization texts [142]. However, these approaches cannot guarantee that an already detected minimizer will be avoided after restarting the algorithm.

An alternative approach consists of transforming the objective function into a new one that excludes the already detected solutions. Well-known examples of this type are the *filled functions* [37]. Similar techniques have been recently developed and successfully used with PSO [95, 100]. The *Stretching* technique consists of a two-stage transformation of the objective function [95],

$$F_1(\mathbf{x}) = f(\mathbf{x}) + \gamma_1 \|\mathbf{x} - \mathbf{x}^*\| \left[1 + \text{sign}\left(f(\mathbf{x}) - f(\mathbf{x}^*)\right)\right], \tag{20}$$

$$F_2(\mathbf{x}) = F_1(\mathbf{x}) + \gamma_2 \frac{1 + \text{sign}\left(f(\mathbf{x}) - f(\mathbf{x}^*)\right)}{\tanh\left(\mu\left(F_1(\mathbf{x}) - F_1(\mathbf{x}^*)\right)\right)}, \tag{21}$$

where $f(\mathbf{x})$ is the original objective function, $\mathbf{x}^*$ is the best detected solution of the algorithm so far and,

$$\text{sign}(\mathbf{z}) = \begin{cases} -1, & \text{if } \mathbf{z} < 0, \\ 0, & \text{if } \mathbf{z} = 0, \\ +1, & \text{if } \mathbf{z} > 0, \end{cases}$$

is the three-valued sign function. As soon as a local minimizer (or generally a sub-optimal solution) is detected, the transformation $F_1(\mathbf{x})$ stretches the objective function upward, while $F_2(\mathbf{x})$ transforms the detected solution $\mathbf{x}^*$ into a local maximizer. Under proper parameter setting, Stretching has the ability to remove higher local minima than the detected solution $\mathbf{x}^*$, while leaving unchanged all lower minima as well as the global one. This is illustrated in Fig. 6 for an 1-dimensional instance of a well-known test function. Thus, it can be very useful in problems with a multitude of local minima that can mislead the algorithm after its re-initialization. Of course, if a better solution is found in a subsequent application of the algorithm, it can simply replace $\mathbf{x}^*$ in Eqs. (20) and (21).

However, if Stretching is applied on a global minimizer, all other minimizers vanish. For this reason, Stretching is inappropriate for detecting multiple global minimizers (see next section for a relevant technique that tackles this problem). Also, similarly to its filled functions predecessors, Stretching may introduce new local minima around the point of application $\mathbf{x}^*$ [101]. This is also known as the

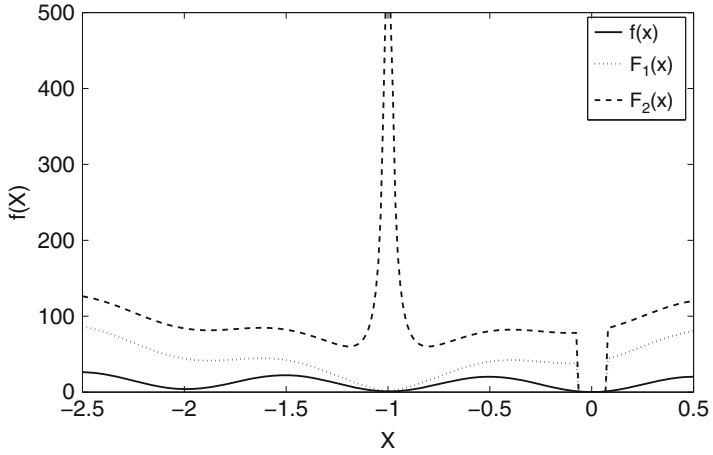

Fig. 6 The Stretching technique applied on the 1-dimensional Rastrigin function for the local minimizer $\mathbf{x}^* = -1$ and parameters $\gamma_1 = \gamma_2 = 20$, $\mu = 0.1$

mexican hat effect and has been addressed in [153] through proper parameterization. Naturally, the application of Stretching is not limited to the Canonical PSO. It can be incorporated to any PSO variant or even different algorithms [42].

Detecting Multiple Minimizers

Deflection [78] is a technique that works similarly to Stretching but it has only local effect on the objective function. Thus, it can be applied in cases where multiple (global or local) solutions are needed. Deflection has been used with PSO with promising results in detecting multiple global and/or local minimizers [101].

Let $f(\mathbf{x})$ be the objective function and $\mathbf{x}_1^*, \ldots, \mathbf{x}_k^*$, be k previously detected solutions. Then, Deflection transforms the objective function as follows [101],

$$F(\mathbf{x}) = \frac{f(\mathbf{x})}{\prod_{i=1}^{k} T_i\left(\mathbf{x}, \mathbf{x}_i^*, \lambda_i\right)}, \tag{22}$$

where T_i is defined as,

$$T_i\left(\mathbf{x}, \mathbf{x}_i^*, \lambda_i\right) = \tanh\left(\lambda_i \|\mathbf{x} - \mathbf{x}_i^*\|\right), \tag{23}$$

and λ_i are relaxation parameters, $i = 1, 2, \ldots, k$. The idea behind this transformation is the same as in Stretching, i.e., the transformation of detected minimizers into local maximizers. However, Deflection changes the objective function only locally, around the point of application. The magnitude of change depends on the parameters λ_i that need proper tuning. Also, Deflection requires strictly positive objective functions in order to achieve the desirable effect. In cases where the problem at

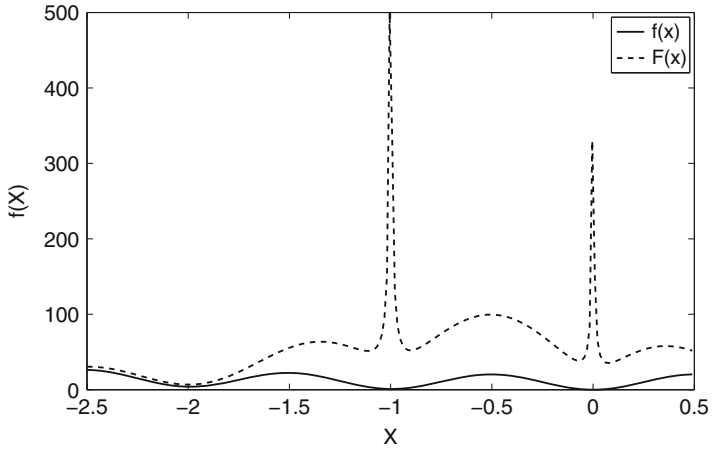

Fig. 7 The Deflection technique applied on the 1-dimensional Rastrigin function for the local minimizer $\mathbf{x}_1^* = -1$ as well as the global minimizer $\mathbf{x}_g^* = 0$ and parameters $\lambda_1 = \lambda_g = 1$

hand is not strictly positive, it can be simply shifted to positive values by using $f(\mathbf{x}) = f(\mathbf{x}) + c$, with a sufficiently large bias $c > 0$, prior to the application of Deflection.

Figure 7 illustrates Deflection for the 1-dimensional instance of the Rastrigin test function. The mexican hat effect appears also in Deflection, although its impact can be controlled with proper selection of the parameters λ_i. Moreover, Deflection can be combined with a *repulsion* technique that prevents the particles from visiting the neighborhoods of detected solutions and possibly get trapped in the artificially introduced local minima [101, 106]. Combined with PSO, Deflection offered previously undetected solutions in computationally demanding optimization problems [129].

Penalty Functions for Constrained Optimization Problems

Constrained optimization problems are accompanied by a set of constraints that need to be satisfied at the final solution. In general, such a problem can be defined as,

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \;\text{ subject to }\; C_i(\mathbf{x}) \leqslant 0, \quad i = 1, 2, \ldots, k, \tag{24}$$

where $C_i(\mathbf{x})$ are inequality constraints. Different forms of constraints can be equivalently given in the form above as follows,

$$C_i(\mathbf{x}) \geqslant 0 \Leftrightarrow -C_i(\mathbf{x}) \leqslant 0,$$
$$C_i(\mathbf{x}) = 0 \Leftrightarrow C_i(\mathbf{x}) \geqslant 0 \;\text{ and }\; C_i(\mathbf{x}) \leqslant 0.$$

Penalty functions have been widely used in constrained problems. The main goal is to penalize all solutions that lie in the infeasible region, such that the algorithm will be directed again into the search space.

The general form of a penalty function for the problem of Eq. (24) is defined as,

$$f(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x}),$$

where,

$$P(\mathbf{x}) = \begin{cases} \alpha > 0, & \text{if } C_i(\mathbf{x}) > 0 \text{ for at least one } i, \\ 0, & \text{otherwise.} \end{cases}$$

In the simplest case, the penalty term $P(\mathbf{x})$ can be constant for all infeasible solutions. However, this choice is not always the most suitable one, since it neglects the degree of violation and does not provide any information to the algorithm regarding the distance of the infeasible solutions from the feasible ones. Thus, the penalty term is recommended to take into consideration the number of violated constraints as well as the degree of violation for every infeasible solution. Moreover, the magnitude of penalties can be time-varying, starting with mild penalties in early stages and becoming strict in the last phase of the optimization procedure.

The Canonical PSO has been combined with such a penalty function defined as follows [98, 159],

$$f(\mathbf{x}) = f(\mathbf{x}) + h(t)H(\mathbf{x}), \tag{25}$$

where,

$$H(\mathbf{x}) = \sum_{i=1}^{k} \theta\left(q_i(\mathbf{x})\right) q_i(\mathbf{x})^{\gamma(q_i(\mathbf{x}))}, \tag{26}$$

and,

$$q_i(\mathbf{x}) = \max\left\{0, C_i(\mathbf{x})\right\}, \qquad i = 1, 2, \ldots, k.$$

The weight $h(t)$ controls the impact of the penalty term $H(\mathbf{x})$ with the number of iterations t. The degree of violation is accounted in $q_i(\mathbf{x})$ and manipulated with the power function $\gamma(q_i(\mathbf{x}))$ as well as with the multi-stage assignment function $\theta\left(q_i(\mathbf{x})\right)$. An alternative penalty function was proposed in [23],

$$f(\mathbf{x}) = f(\mathbf{x}) + H(\mathbf{x}),$$

with,

$$H(\mathbf{x}) = w_1 H_{\mathrm{NVC}}(\mathbf{x}) + w_2 H_{\mathrm{SVC}}(\mathbf{x}),$$

where $H_{\text{NVC}}(\mathbf{x})$ is the number of violated constraints and,

$$H_{\text{SVC}}(\mathbf{x}) = \sum_{i=1}^{k} \max\{0, C_i(\mathbf{x})\},$$

is the sum of violations. The weights w_1 and w_2 can be either fixed or time-varying and they determine the importance of each penalty term.

The Canonical PSO updates its best positions according to Eq. (7) by comparing objective values, solely. However, it is commonly desirable to allow only feasible best positions in order to avoid oscillations of the particles in the infeasible space. On top of that, the final solution shall be feasible and, thus, the particles shall eventually concentrate their search efforts in the feasible region. Yet, in the penalty functions defined above, it is still possible that an infeasible solution attains lower value than a feasible one and, therefore, be preferable for inclusion in the best positions.

In order to prevent such undesirable inclusions, a set of rules can be imposed in the selection procedure:

1. Between feasible solutions, the one with the smallest objective value is preferable.
2. Between a feasible and an infeasible solution, the feasible one is always preferable.
3. Between two infeasible solutions, the one with the smallest penalty term $H(\mathbf{x})$ is preferable.

These selection rules along with the penalty function approaches have been used with PSO in various constrained optimization problems with promising results [82, 111].

Tackling Discrete Problems

The Canonical PSO was initially introduced as a continuous optimization method and its operators are designed to work on real-valued search spaces. In Mathematical Programming literature, integer optimization problems can be tackled with continuous algorithms by extending the discrete problem to a continuous one and rounding the solutions to the nearest integers. Such approaches have been used with Branch and Bound algorithms combined with quadratic programming solvers [69, 79].

A similar approach was used also with PSO for solving integer problems [68]. Specifically, the particles are let to assume real vectors in their current positions, although they are rounded to the nearest integer vector when evaluated with the objective function. Also, the rounded integer vectors are preferable as best positions, since the final solution shall be integer. Putting it formally, if $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^\top$ is the current position of the i-th particle then its objective value $f(\mathbf{x}_i) = f(\mathbf{z}_i)$ is evaluated on the integer vector $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{id})^\top$ defined as,

$$z_{ij} = \lfloor x_{ij} + 0.5 \rfloor, \qquad i \in I,\ j \in D.$$

Rounding has effectively worked for various problems of integer and mixed integer type [68, 82, 92, 111]. Note that in mixed integer problems, no additional representation scheme is required for the real and the discrete parameters. The algorithm works by simply rounding the coordinate components that correspond to integer variables.

Yet, if PSO's velocities become small enough, it is probable that rounding will result in search stagnation due to the inability of producing different integer components. This potential deficiency can be tackled either by restarting the current positions of the particles (and possibly also the best positions except the overall best one) as soon as stagnation is identified or by applying gradual truncation of the decimal digits of the position vectors of the particles as the number of iterations increases [68].

Another problem category that involves discrete search spaces comprises of *permutation problems*. In such problems, the main goal is the detection of an optimal permutation of fixed elements (e.g., numbers, items, indices, etc.) and they are frequently met in Combinatorics and Operations Research [11]. Such problems can be tackled through real-valued algorithms by using the *smallest position value* (SPV) representation [138]. Specifically, let,

$$\mathcal{Z} = \{Z_1, \ldots, Z_d\}$$

be an ordered set of the elements of interest Z_i, $i \in D$. Then, the components of a real-valued particle are defined as numerical weights that denote the priority of the corresponding discrete elements according to a predefined mapping. For example, the components of the i-th particle $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^\top$ can be mapped as follows,

$$\begin{array}{rccccc} \text{ordered list of (discrete) elements:} & Z_1 & Z_2 & \cdots & Z_d \\ & \updownarrow & \updownarrow & & \updownarrow \\ \text{particle's components (weights):} & x_{i1} & x_{i2} & \cdots & x_{id} \end{array}$$

Then, the weights (particle's components) are sorted in ascending order and the corresponding permutation is received by re-arranging the discrete elements accordingly, i.e.,

$$\begin{array}{rccccc} \text{sorted particle's components (weights):} & x_{ik_1} & x_{ik_2} & \cdots & x_{ik_d} \\ & \updownarrow & \updownarrow & & \updownarrow \\ \text{corresponding permutation:} & Z_{k_1} & Z_{k_2} & \cdots & Z_{k_d} \end{array}$$

Thus, each particle corresponds to a permutation received after sorting its components and PSO's goal is to find the appropriate weights that produce optimal or near-optimal permutations. The most common search space in such problems

is $\mathcal{X} = [0,1]^d$, i.e., the weights are all assumed to lie in the range $[0,1]$. Obviously, there is an infinite number of weight vectors that correspond to a specific permutation, since it depends only on the relative ordering of the weights and not their actual values. SPV has been successfully combined with PSO in various permutation problems [65, 103, 139, 140].

Enhanced and Specialized PSO Variants

This section is devoted to PSO variants that stemmed from the Canonical PSO as improvements or specialized modifications for specific problem types. The pluralism of PSO variants in literature renders a complete presentation impossible. For this reason, a selection is made based on the novelty introduced by each method as well as the influence for further developments. Chronological order of appearance of the methods is partially retained.

Binary PSO

The first *Binary PSO* (BPSO) variant was developed in 1997 [61]. Instead of using rounding for transforming the real-valued parameters into binary ones, the algorithm introduced a new interpretation of the velocities. Specifically, each component of the velocity's vector is considered as the input of a sigmoid that determines the state (0 or 1) of the corresponding particle's position component. Thus, the velocity $\mathbf{v}_i$ of the i-th particle is updated according to Eq. (4), while the current position is updated as follows [61],

$$x_{ij}^{(t)} = \begin{cases} 1, & \text{if } \mathcal{R} < S\left(v_{ij}^{(t)}\right), \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

for all $i \in I$, $j \in D$, and $\mathcal{R} \sim \mathcal{U}(0,1)$ is a random variable. The sigmoid is defined as,

$$S(x) = \frac{1}{1+\exp(-x)},$$

clamping the velocity in the range $[0,1]$ so it can be translated to probability. The rest of the algorithm follows the basic rules of the Canonical PSO.

The algorithm was demonstrated on a set of test problems with promising results. It also exhibits conceptual similarities with reinforcement learning approaches [73]. BPSO is still considered as the basis for the development of modern and ad hoc binary PSO variants for specific applications [9, 14, 52, 116, 158].

Guaranteed Convergence PSO

The *Guaranteed Convergence PSO* (GCPSO) was introduced in 2002 [145] based on the observation that small velocities can prohibit the convergence of the gbest PSO model with inertia weight. This is ascribed to the cancellation of the attraction forces of the overall best particle toward its best positions, along with small velocities that essentially immobilize it in the long run.

The problem was solved by modifying the position update of the overall best particle (denoted with the index g), as follows [145],

$$x_{gj}^{(t+1)} = p_{gj}^{(t)} + \chi\, v_{gj}^{(t)} + \rho^{(t)}(1 - 2\mathcal{R}), \tag{28}$$

where $j \in D$, $\rho^{(t)}$ is a *scaling factor*, and $\mathcal{R} \sim \mathcal{U}(0, 1)$ is a random variable. The rest of the particles are updated with the standard rules of Eqs. (4) and (5). The scaling factor is dynamically adjusted based on the number of consecutive successes and failures in improving the overall best, i.e.,

$$\rho^{(t+1)} = \begin{cases} 2\,\rho^{(t)}, & \text{if } m_{\text{suc}}^{(t)} > T_{\text{suc}}, \\ 0.5\,\rho^{(t)}, & \text{if } m_{\text{fail}}^{(t)} > T_{\text{fail}}, \\ \rho^{(t)}, & \text{otherwise}, \end{cases}$$

where $m_{\text{suc}}^{(t)}$ and $m_{\text{fail}}^{(t)}$ are counters of the consecutive successes and failures at iteration t, respectively, and T_{suc}, T_{fail}, are the corresponding user-defined thresholds. The counters are reset whenever the row of consecutive successes or failures is interrupted by a failure or success, respectively. Convergence of GCPSO was proved and the values $\rho^{(0)} = 1$, $T_{\text{suc}} = 15$, and $T_{\text{fail}} = 5$, were recommended in [145], where GCPSO was shown to be very promising on typical benchmark problems.

Bare Bones PSO

The *Bare Bones PSO* (BBPSO) was introduced in 2003 [58] as a simplification of the Canonical PSO. Its main feature is the elimination of the velocity update of Eq. (4) and its replacement with Gaussian sampling around the best positions. Specifically, the current position is updated by sampling the Gaussian distribution,

$$x_{ij}^{(t+1)} \sim \mathcal{N}\left(\mu_{ij}^{(t)}, \sigma_{ij}^{2\,(t)}\right), \tag{29}$$

where,

$$\mu_{ij}^{(t)} = \frac{1}{2}\left(p_{gj}^{(t)} + p_{ij}^{(t)}\right), \qquad \sigma_{ij}^{2\,(t)} = \left|p_{gj}^{(t)} - p_{ij}^{(t)}\right|,$$

and g stands for the index of the overall best particle. The rest of the Canonical PSO's procedures (such as best position update and boundary violations handling) are retained.

BBPSO aimed at offering a very simple, parameter-free PSO variant. Its convergence properties were studied in [10, 90, 115] and enhanced variants were proposed for applications in various scientific fields [66, 164–166].

Fully Informed PSO

The *Fully Informed PSO* (FIPS) was introduced in 2004 [80] as a variant of the Canonical PSO that extends the concept of neighbor influence. Specifically, in FIPS each particle is influenced by all its neighbors and not only the best one. The particles' update scheme of Eqs. (4) and (5) is modified in the following update rule,

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + \chi \left(x_{ij}^{(t)} - x_{ij}^{(t-1)} \right) + \sum_{k \in NB_i} \mathcal{C}_k \left(p_{kj}^{(t)} - x_{ij}^{(t)} \right), \tag{30}$$

where $i \in I$, $j \in D$, NB_i is the i-th particle's neighborhood of size s_i, and $\mathcal{C}_k \sim \mathcal{U}(0, c/s_i)$ is a random variable. Adapting the analysis of [21] in FIPS, the default parameter setting of Eq. (18) is used.

FIPS introduced a novel point of view for the concept of neighborhood. The social influence was enhanced, providing additional attractors to the particles. The outcome is their stochastic move around a stochastic average of the best positions of all neighbors. However, this also implies the dependency of the algorithm's performance on the selected neighborhood topology. For example, in the gbest PSO model where the whole swarm is the neighborhood of all particles, their move tends to be nearly random [59]. A multitude of neighborhood structures (more than 1000) were tested in [80] and interesting conclusions were derived regarding the potential of FIPS to improve the Canonical PSO.

Quantum PSO

Quantum PSO (QPSO) was introduced in 2004 [133], putting the algorithm in a new framework. In QPSO, the swarm is considered as a quantum system where each particle possesses a quantum state, while moving in a *Delta potential well* (DPW) toward a position $\mathbf{p}$. The quantum state depends on the employed wave function.

Borrowing from previous PSO analyses such as [21], the aforementioned position $\mathbf{p}$ is defined as in Eq. (11). Depending on the considered potential, different variants of QPSO can be defined. Three established approaches are the following [81, 133, 135],

$$\text{Delta Potential Well:} \quad \mathbf{x}_i^{(t+1)} = \mathbf{p}^{(t)} \pm \frac{\ln\left(\frac{1}{\mathcal{R}}\right)}{2\,q\,\ln(\sqrt{2})} \left\| \mathbf{x}_i^{(t)} - \mathbf{p}^{(t)} \right\|, \tag{31}$$

$$\text{Harmonic Oscillator:} \quad \mathbf{x}_i^{(t+1)} = \mathbf{p}^{(t)} \pm \frac{\sqrt{\ln\left(\frac{1}{\mathcal{R}}\right)}}{0.47694\,q} \left\| \mathbf{x}_i^{(t)} - \mathbf{p}^{(t)} \right\|, \tag{32}$$

$$\text{Square Well:} \quad \mathbf{x}_i^{(t+1)} = \mathbf{p}^{(t)} + \frac{0.6574}{\xi\,q} \cos^{-1}\left(\pm\sqrt{\mathcal{R}}\right) \left\| \mathbf{x}_i^{(t)} - \mathbf{p}^{(t)} \right\|, \tag{33}$$

where $i \in I$, $\mathcal{R} \sim \mathcal{U}(0,1)$ is a random number, and ξ, q, are user-defined parameters. Despite the identified parameter sensitivity of the algorithm, it was embraced by the scientific community and extended in a number of interesting applications [17, 22, 39, 45, 50, 76, 163].

Unified PSO

The *Unified PSO* (UPSO) was introduced in 2004 [102] as a PSO variant that harnesses the gbest and lbest PSO models in a unified scheme. The motivation behind its development lies in the good intensification (exploitation) properties of the gbest model and the corresponding good diversification (exprolartion) properties of the lbest model. Their combination can form variants with different trade-offs of these two properties.

UPSO is based on the update equations of the Canonical PSO with constriction coefficient. Specifically, let,

$$\mathcal{L}_{ij}^{(t+1)} = \chi\, v_{ij}^{(t)} + \mathcal{C}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)}\right) + \mathcal{C}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)}\right), \tag{34}$$

denote the velocity update of Eq. (4) for the lbest PSO model, where g_i is the index of the best neighbor of the i-th particle and $j \in D$. Also, let,

$$\mathcal{G}_{ij}^{(t+1)} = \chi\, v_{ij}^{(t)} + \mathcal{C}_3 \left(p_{ij}^{(t)} - x_{ij}^{(t)}\right) + \mathcal{C}_4 \left(p_{gj}^{(t)} - x_{ij}^{(t)}\right), \tag{35}$$

be the corresponding velocity update for the gbest PSO model, i.e., g denotes the overall best particle. Then, the basic UPSO scheme is defined as [102],

$$v_{ij}^{(t+1)} = u\,\mathcal{G}_{ij}^{(t+1)} + (1-u)\,\mathcal{L}_{ij}^{(t+1)}, \tag{36}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{37}$$

where $i \in I$, $j \in D$, and $u \in [0,1]$ is a user-defined parameter called the *unification factor*, which controls the influence of the gbest and lbest term. Obviously, $u = 0$ corresponds to the lbest model, while $u = 1$ corresponds to

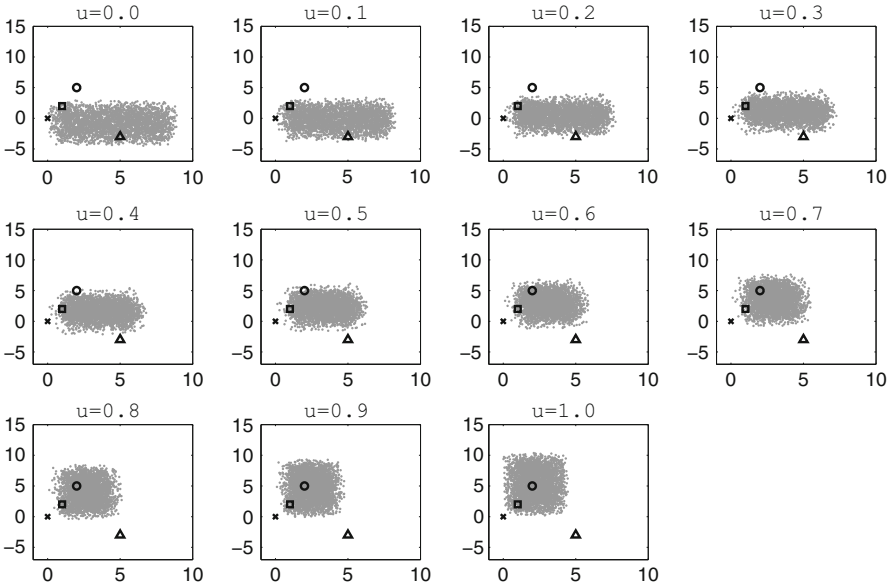

Fig. 8 The distribution of 3000 possible new positions (*light grey points*) of the 2-dimensional particle $\mathbf{x}_i = (0,0)^\top$ (*cross*) with own best position $\mathbf{p}_i = (1,2)^\top$ (*square*), neighborhood's best position $\mathbf{p}_{g_i} = (5,-3)^\top$ (*triangle*), and overall best $\mathbf{p}_g = (2,5)^\top$ (*circle*), for different values of $u \in [0,1]$. For simplicity, the velocity vector is set to $\mathbf{v}_i = (0,0)^\top$

the gbest model. All intermediate values define UPSO variants that combine the diversification/intensification properties of the two models. Figure 8 illustrates the distribution of new positions of a particle for different values of the unification factor.

In addition to the main UPSO model, an alternative with increased stochasticity was also proposed [102]. It came in two forms, namely,

$$v_{ij}^{(t+1)} = \mathcal{R}\, u\, \mathcal{G}_{ij}^{(t+1)} + (1-u)\, \mathcal{L}_{ij}^{(t+1)}, \tag{38}$$

which is mostly based on the lbest model, and,

$$v_{ij}^{(t+1)} = u\, \mathcal{G}_{ij}^{(t+1)} + \mathcal{R}\,(1-u)\, \mathcal{L}_{ij}^{(t+1)}, \tag{39}$$

which is based on the gbest model [102]. The stochastic parameter $\mathcal{R} \sim \mathcal{N}(\mu, \sigma^2)$ is normally distributed and imitates the mutation operators in Evolutionary Algorithms.

Theoretical properties of UPSO were studied in [102], while a thorough investigation of its parameter setting and adaptation was offered in [104]. Its performance has been studied on various problems [3, 38, 41, 65, 83, 84, 106, 144].

Cooperative PSO

Cooperative population-based algorithms [117] are based on the concept of cooperation between individuals toward a common goal. Cooperation can be either explicit through direct communication among them or implicit through a shared memory where information is deposited. Cooperation can be considered in a multi- or single-population framework. In the first case, each population usually operates on a subspace of the original search space, e.g., on one coordinate direction of the solution vector. Thus, its individuals carry partial solutions that are combined with those of the other populations, forming complete solutions. In the second case, the individuals usually carry complete solution information that is combined with the rest by using special recombination schemes [117].

In the context of PSO, one of the first notable attempts to design a *Cooperative PSO* (CPSO) took place in 2004 [146]. That version consists of a number d of swarms (equal to the problem's dimension), containing N_k particles each, $k \in D$, i.e., according to Eq. (3),

$$\mathcal{S}^{[1]} = \left\{A_1^{[1]}, \ldots, A_{N_1}^{[1]}\right\}, \quad \ldots, \quad \mathcal{S}^{[d]} = \left\{A_1^{[d]}, \ldots, A_{N_d}^{[d]}\right\}.$$

Each swarm probes only one coordinate direction of the solution vector, applying any PSO variant (Canonical PSO was used in the specific one). Yet, the evaluation of the particles with the objective function requires complete d-dimensional solutions. Thus, two main issues need to be addressed:

1. How shall particles from different swarms be selected to form complete solutions?
2. How shall particles be awarded or penalized for their contribution in solutions' quality?

The decisions on these crucial properties have direct impact on the algorithm's performance and, thus, require special attention.

In [146] two alternative schemes were proposed. The first scheme, denoted as CPSO-S$_k$, introduced a *context vector* $\mathbf{z}^*$ for the evaluation of the particles. This vector constitutes an external memory where each swarm $\mathcal{S}^{[k]}$ participates with its 1-dimensional overall best at the corresponding k-th direction component, i.e.,

$$\mathbf{z}^* = \left(\mathbf{p}_g^{[1]}, \mathbf{p}_g^{[2]}, \ldots, \mathbf{p}_g^{[d]}\right),$$

where $\mathbf{p}_g^{[k]}$ is the overall (1-dimensional) best of the k-th swarm [146]. Then, the evaluation of the i-th particle of the k-th swarm is done by replacing the k-th swarm's best in $\mathbf{z}^*$ with its own information, i.e.,

$$f\left(\mathbf{x}_i^{[k]}\right) = f\left(\mathbf{z}_{[i,k]}^*\right),$$

where,

$$\mathbf{z}^*_{[i,k]} = \left(\mathbf{p}_g^{[1]}, \ldots, \mathbf{p}_g^{[k-1]}, \mathbf{x}_i^{[k]}, \mathbf{p}_g^{[k+1]}, \ldots, \mathbf{p}_g^{[d]}\right),$$

where $\mathbf{x}_i^{[k]}$ is the current position of the i-th particle of the k-th swarm, which is under evaluation. Naturally, instead of the overall bests of the swarms, randomly selected best positions can be used in the context vector. Also, swarms of higher dimension can be used. However, both these alternatives can radically change the algorithm's performance. Obviously, the context vector $\mathbf{z}^*$ constitutes the best approximation of the problem's solution with CPSO-S_k.

The second variant presented in [146], denoted as CPSO-H_k, combines CPSO-S_k with the Canonical PSO and applies each algorithm alternatively in subsequent iterations. In addition, information exchange between the two algorithms was considered by sharing half of the discovered solutions between them. The experimental assessment revealed that both CPSO-S_k and CPSO-H_k are promising, opening the ground for further developments such as the ones in [48, 136, 152, 167].

Comprehensive Learning PSO

The *Comprehensive Learning PSO* (CLPSO) [72] was proposed in 2006 as an alternative for alleviating gbest PSO's premature convergence problem, which can be attributed to the use of the overall best position in the update equation of the velocities. In CLPSO, each particle can use the best position of any other particle to independently update its velocity, based on a probabilistic scheme.

Specifically, the velocity update of Eq. (4) is replaced with the following [72],

$$v_{ij}^{(t)} = \chi\, v_{ij}^{(t)} + \mathcal{C}\left(p_{q_{[i,j]}j} - x_{ij}^{(t)}\right), \tag{40}$$

where $j \in D$, $i \in I$, and $q_{[i,j]} \in I$ is the index of the particle that is used for the update of the j-th component of the i-th particle's velocity vector. Naturally, this particle can be either the i-th particle itself or another particle from the swarm. This decision is probabilistically made according to predefined probabilities $\rho_1, \rho_2, \ldots, \rho_d$, i.e.,

$$q_{[i,j]} = \begin{cases} i, & \text{if } \mathcal{R} \leqslant \rho_j, \\ \text{TOURN}\,(I'), & \text{otherwise,} \end{cases} \qquad \text{for all } j \in D,$$

where $\mathcal{R} \sim \mathcal{U}(0,1)$ is a uniformly distributed random variable, $I' = I \setminus \{i\}$, and TOURN (I') is an index selected from I' through *tournament selection* [72]. The latter procedure includes the random selection of two particles from the set I'. The best between them, i.e., the one with smallest objective value, is the winner and participates in the update of v_{ij}.

In case of $q_{[i,j]} = i$, for all $j \in D$, one of the components of v_{ij} is randomly selected and determined anew by using another particle. Also, the indices $q_{[i,j]}$ are

updated for each particle after a number of non-improving iterations. CLPSO has been extensively studied in [72], while a number of improvements, modifications, and applications in various fields have been proposed in relevant literature [40, 43, 154, 161].

TRIBES

The *TRIBES* algorithm was proposed in 2006 [19] as a novel PSO variant with self-adaptation capability. It is based on a special communication scheme between neighborhoods and admits the update rules of any PSO variant. In TRIBES, the i-th particle is called *informant* of the j-th particle if it shares its best position for the update of the latter. Accordingly, a *tribe* can be defined as a subset of the swarm, where each one of its members is informant of all the rest in the same tribe. Obviously, the swarm is the union set of all tribes.

Each tribe must have at least one communication channel with another tribe. In other words, between two particles there shall be at least one path in the neighborhood's graph that connects them [19]. Also, the algorithm is self-adaptive, i.e., existing tribes can be deleted and new tribes can be generated. Hence, the communication channels between tribes also change dynamically. The goodness criterion for the tribes is related to the performance of their members-particles, which are characterized as *neutral* if they have not improved their best position in the last iteration, *good* if improvement was achieved in the last iteration, and *excellent* if improvement was achieved for more than one consecutive iterations. Accordingly, a tribe TR that contains s_{TR} particles is characterized as follows,

$$TR \text{ is } \begin{cases} \text{good}, & \text{if } N_{TR}^{\text{good}} > \mathcal{R}, \\ \text{bad}, & \text{otherwise}, \end{cases}$$

where N_{TR}^{good} is the number of good particles in tribe TR, and $\mathcal{R}$ is a randomly selected integer in $\{0, 1, \ldots, s_{TR}\}$. Moreover, additional rules are applied for the generation/deletion of particles and tribes as follows:

1. The worst particle of the best tribe can be eliminated, inheriting its communication channels to the best particle of its tribe.
2. If a tribe consists of only one particle, it is eliminated if it has an informant with better performance.
3. Each tribe that was characterized as "bad" generates two new particles. The first one is randomly generated within the whole search space. The second one is uniformly generated in the sphere with center the best position of the best informant of the tribe's best particle, and radius equal to the distance between the latter and the sphere's center.

Adaptations were recommended to occur after a number of iterations so that the algorithm can deploy its dynamic [19]. Promising results were received with

TRIBES under various settings [19]. Although TRIBES is a rather controversial PSO variant, it has contributed toward the development of self-adaptation mechanisms [25–27] and has been applied on interesting problems [30].

Niching PSO

Niching algorithms are applied on multimodal optimization problems where the main goal is the identification of multiple global/local minima. In such problems, the algorithms must be capable of identifying minima and retaining them until the end of the search. Although transformation techniques such as the ones presented in section "Performance-Enhancing Techniques" can be used in these cases, alternative algorithmic models that do not use external procedures have been developed.

Two efficient *Niching PSO* approaches are the *Speciation-based PSO* (SPSO) [94] and the *Fitness Euclidean-distance Ratio PSO* (FERPSO) [70]. In SPSO, the swarm is divided into subswarms, which are considered as species represented by the dominant (best) particle in the subswarm. A niche radius is also specified to define the size of species. Special procedures are applied for determining species and their seeds, while the global best particle is replaced by the species best or species seed. Also, all particles in the same species use the same neighborhood best at each iteration.

On the other hand, FERPSO is based on the lbest PSO model of Eqs. (4) and (5) [70], where the neighborhood's best $\mathbf{p}_{g_i}$ is taken as the particle that maximizes the *fitness Euclidean-distance ratio* (FER), defined as,

$$\mathrm{FER}_{i,k} = \frac{\alpha\,(f\,(\mathbf{p}_i) - f\,(\mathbf{p}_k))}{\|\mathbf{p}_i - \mathbf{p}_k\|}, \qquad k \in I,$$

where the scaling factor α is defined as,

$$\alpha = \frac{\sqrt{\sum_{l=1}^{d} \left(x_l^{\max} - x_l^{\min}\right)^2}}{f\left(\mathbf{p}_g\right) - f\left(\mathbf{p}_w\right)},$$

with $\mathbf{p}_g$ and $\mathbf{p}_w$ being the swarm's best and worst particles, respectively [70]. The effectiveness of both SPSO and FERPSO has led to further enhancements such as the ones in [71, 118, 125].

Standard PSO

The *Standard PSO* (SPSO) was introduced in an attempt to define a baseline for the development and assessment of new PSO variants. Although there have been various versions (2006, 2007, and 2011), only the latest one, SPSO-2011 [20], is considered here since it cures a number of deficiencies identified in the previous versions.

A characteristic feature of SPSO-2011 is the independence on the coordinate system. Let,

$$\mathcal{P}_{ij}^{(t)} = x_{ij}^{(t)} + \mathcal{C}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right), \tag{41}$$

$$\mathcal{L}_{ij}^{(t)} = x_{ij}^{(t)} + \mathcal{C}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right), \tag{42}$$

with $j \in D$, define two points for the i-th particle that are a little "beyond" its own best position and its neighborhood's best position, respectively, at iteration t. Then, the center of gravity between the two points and the particle's current position is defined as,

$$\mathcal{G}_i^{(t)} = \frac{1}{3} \left(\mathbf{x}_i^{(t)} + \mathcal{P}_i^{(t)} + \mathcal{L}_i^{(t)} \right). \tag{43}$$

A new point $\mathbf{x'}_i^{(t)}$ is randomly (not necessarily uniformly) generated in the hypersphere $\mathcal{H}_i^{(t)}$ with center $\mathcal{G}_i^{(t)}$ and radius equal to its distance from the actual $\mathbf{x}_i^{(t)}$, i.e.,

$$\mathbf{x'}_i^{(t)} \in \mathcal{H}_i^{(t)} \left(\mathcal{G}_i^{(t)}, \left\| \mathcal{G}_i^{(t)} - \mathbf{x}_i^{(t)} \right\| \right). \tag{44}$$

Then, the update equations of SPSO-2011 are given as follows [20],

$$v_{ij}^{(t+1)} = \chi \, v_{ij}^{(t)} + x'^{(t)}_{ij} - x_i^{(t)}, \tag{45}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{46}$$

where $i \in I$ and $j \in D$. The rest of the algorithm follows the Canonical PSO approach. The particles are bounded in the search space according to Eq. (19). The algorithm's parameters were studied in [162], while a thorough theoretical analysis was provided in [13].

Memetic PSO

The term *Memetic Algorithm* [85] is used to describe hybrid algorithms that consist of population-based metaheuristics with additional local search or learning mechanisms. Early *Memetic PSO* (MPSO) schemes appeared in 2005 [74], hybridizing PSO with the Solis and Wets local search approach. Later, different schemes were proposed using alternative local search algorithms, such as the Random Walk with Direction Exploitation and the Hooke and Jeeves method [108, 109]. Recently, PSO was integrated with gradient-based optimization as well as direct search approaches in MEMPSODE, an efficient general-purpose software package [149]. Further enhancements and applications can be found in [5, 44, 150].

Besides the choice of an appropriate local search algorithm, which is mostly a problem-dependent decision, additional resolutions shall be made prior to the application of MPSO [106]:

1. When to apply local search?
2. Where to apply local search?
3. What computational budget shall be devoted to local search?

A straightforward choice is the application of local search on the best positions (if updated) and/or the current positions of the particles at each iteration, until a local minimum is found. Naturally, this approach would require excessive computational resources that are hardly available in practice.

For this reason, the following schemes were proposed in [108] after empirical evaluations on established test problems:

(S1) Application of local search on the overall best position $\mathbf{p}_g$, whenever it changes.
(S2) For each best position $\mathbf{p}_i$, $i \in I$, local search is applied with a predefined probability ρ.
(S3) Local search is applied both on $\mathbf{p}_g$ and some randomly selected best positions $\mathbf{p}_i$, $i \in I$.
(S4) Local search is applied on $\mathbf{p}_g$ as well as on the best positions $\mathbf{p}_i$ that lie in adequate distance from $\mathbf{p}_g$, e.g., $\|\mathbf{p}_g - \mathbf{p}_i\| > \varepsilon$, where $\varepsilon > 0$ is a predefined distance usually defined as a fraction of the search space diameter.

In addition, the *Shannon information entropy*, used as a measure of the swarm's information diversity, was employed in [107] along with the above schemes in order to make swarm-level decisions on the application of local search. Further details and extensive results are given in [106], where it is shown that MPSO outperforms the Canonical PSO (as expected) but also its gbest model can outperform the lbest model of Canonical PSO. The latter result suggests that local search applied as above can be beneficial both for PSO's intensification and diversification properties.

Opposition-Based PSO

The opposition-based algorithms are grounded on the concept of *opposite point* [120]. If $\mathbf{x} = (x_1, \ldots, x_d)^\top$ is a point in the search space $\mathcal{X}$ defined as in Eq. (2), then its opposite is defined as a point $\mathbf{x'} = (x'_1, \ldots, x'_d)^\top$ with $x'_j = x_j^{\min} + x_j^{\max} - x_j$, for all $j \in D$. Evaluating both points simultaneously and keeping the best one can accelerate the optimization procedure according to the study in [120].

This scheme was recently adopted in the framework of PSO, producing the *Generalized Opposition-based PSO* (GOPSO) [151]. In GOPSO, there are two swarms, $\mathcal{S}$ and $\mathcal{S}'$, comprising the particles and their opposites, respectively. The

initial positions $\mathbf{x}_i^{(0)}$, $i \in I$, of $\mathcal{S}$ are randomly initialized, while for $\mathcal{S}'$ the initial positions $\mathbf{x'}_i^{(0)}$ are obtained as follows [151],

$$x'^{(0)}_{ij} = \mathcal{R}\left(\alpha_j^{(0)} + \beta_j^{(0)}\right) - x_{ij}, \tag{47}$$

where $i \in I$, $j \in D$, $\mathcal{R} \sim \mathcal{U}(0,1)$, and $\alpha_j^{(0)} = x_j^{\min}$, $\beta_j^{(0)} = x_j^{\max}$. Subsequently, both swarms are evaluated and merged. The N best particles are then selected to form the initial swarm.

At each iteration, a probabilistic decision is taken. The algorithm, with a user-defined probability ρ, either chooses to update the boundaries $\alpha_j^{(t)}$, $\beta_j^{(t)}$, as follows [151],

$$\alpha_j^{(t)} = \min_i \left\{x_{ij}^{(t)}\right\}, \qquad \beta_j^{(t)} = \max_i \left\{x_{ij}^{(t)}\right\}, \tag{48}$$

or applies the update equations of the Canonical PSO defined in Eqs. (4) and (5). The procedure continues with the best positions update of Eq. (7). The new overall best undergoes also mutation, where its components are perturbed with random numbers following a Cauchy distribution [151].

Experimental results have shown that GOPSO can be competitive to other PSO variants [151]. A number of different opposition-based approaches have been proposed in various application fields [28, 55, 77, 168].

PSO in Noisy Environments

Noisy problems arise very often in engineering applications. The use of measurement instruments or approximations based on inaccurate mathematical models impose uncertainty on the objective function values. Thus, *noise-tolerance* is a desirable property for metaheuristic optimization algorithms such as PSO. In [97] the gbest PSO model was studied on a number of test problems contaminated by Gaussian noise, exhibiting promising behavior. Various other studies followed in subsequent years [8, 91, 119].

A common technique for tackling noisy problems is the re-evaluation of the objective function at each point. Specifically, if the objective function is given as,

$$f'(\mathbf{x}) = f(\mathbf{x}) + \mathcal{R},$$

where $\mathcal{R}$ is a random variable following a (usually Gaussian) distribution, then PSO evaluates the particles by using,

$$F(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} f'^{(m)}(\mathbf{x}),$$

where $f'^{(m)}(\mathbf{x})$ is the m-th re-evaluation of $\mathbf{x}$ using $f'(\mathbf{x})$. Re-evaluation serves as a mean for approximating the expected value of the noisy objective function, i.e., $F(\mathbf{x}) \approx \mathbb{E}(f'(\mathbf{x}))$. Accuracy increases with the number M of re-evaluations, although it also increases the computational cost.

Thus, the trade-off between better estimations of the objective values and the corresponding computational burden shall be tuned. In such cases, specialized techniques such as the *Optimal Computing Budget Allocation* (OCBA) [16] have been used to optimally allocate the re-evaluations budget in order to provide reliable evaluation and identification of the promising particles [91]. These techniques can be used along with proper parameter tuning [8] or learning strategies [110] for improved results. Also, they do not require the modification of the algorithm. Alternatively, specialized operators have been proposed with remarkable success [47].

Multiobjective PSO

Multiobjective optimization (MO) problems consist of a number of objective functions that need to be simultaneously optimized. In contrast to the definition of single-objective problems in Eq. (1), an MO problem is defined as the minimization of a vector function [24],

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x}))^\top,$$

possibly subject to constraints $C_i(\mathbf{x}) \leqslant 0$, $i = 1, 2, \ldots, m$. Typically, the objective functions $f_k(\mathbf{x})$ can be conflicting. Thus, it is highly improbable that a single solution that globally minimizes all of them can be found.

For this reason, the main interest in such problems is concentrated on the detection of *Pareto optimal* solutions. These solutions are nondominated by any other point in the search space, i.e., they are at least as good as any other point for all the objectives $f_k(\mathbf{x})$. Formally, if $\mathbf{x}$, $\mathbf{y}$, are two points in the search space $\mathcal{X}$, then $\mathbf{f}(\mathbf{x})$ is said to dominate $\mathbf{f}(\mathbf{y})$, and we denote $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{y})$, if it holds that,

$$f_k(\mathbf{x}) \leqslant f_k(\mathbf{y}), \text{ for all } k = 1, 2, \ldots, K,$$

and,

$$f_{k'}(\mathbf{x}) < f_{k'}(\mathbf{y}), \text{ for at least one } k' \in \{1, 2, \ldots, K\}.$$

Thus, $\mathbf{x}^* \in \mathcal{X}$ is a Pareto optimal point if there is no other point $\mathbf{y} \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{y}) \prec \mathbf{f}(\mathbf{x}^*)$. Obviously, an (even infinite) set $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots\}$ of Pareto optimal solutions may exist. The set $\{\mathbf{f}(\mathbf{x}_1^*), \mathbf{f}(\mathbf{x}_2^*), \ldots\}$ is called the *Pareto front*.

There are two main approaches for tackling MO problems. The first one aggregates the objectives into a single one and solves the problem with the typical methodologies for single-objective optimization. The second approach requires

vector evaluated operators and it is based on the concept of Pareto dominance. In the context of PSO, early aggregation approaches appeared in 2002 [99], where the Canonical PSO was used for the minimization of a weighted aggregation of the objective functions,

$$F(\mathbf{x}) = \sum_{k=1}^{K} w_k \, f_k(\mathbf{x}), \qquad \sum_{k=1}^{K} w_k = 1.$$

Both a *conventional weighted aggregation* (CWA) approach with fixed weights as well as a *dynamic weighted aggregation* (DWA) approach [53] were investigated with promising results. Obviously, the detection of many Pareto optimal solutions through weighted aggregation requires multiple applications of PSO, since each run provides a single solution of $F(\mathbf{x})$. From the computational point of view, this is a drawback since the swarms can simultaneously evolve many solutions. Yet, it is still a popular approach in applications mostly due to its simplicity.

A *Vector Evaluated PSO* (VEPSO) was also proposed in [99] and parallelized later in [96]. VEPSO uses a number of K swarms, one for each objective f_k. The k-th swarm $\mathcal{S}_k$ is evaluated only with the corresponding objective f_k, $k = 1, 2, \ldots, K$. The swarms are updated according to the gbest model of the Canonical PSO, although with a slight modification. Specifically, the overall best that is used for the velocity update of the particles in the k-th swarm comes from another swarm. Clearly, this is a migration scheme aiming at transferring information among swarms. The donator swarm can be either a neighbor of the k-th swarm in a ring topology scheme as the one described in section "Concept of Neighborhood" or it can be randomly selected [99]. VEPSO was studied on standard MO benchmark problems with promising results [96].

There is a large number of new developments and applications on multiobjective PSO approaches in literature [1, 2, 27, 29, 32, 51, 75, 156, 160, 169]. The interested reader can find comprehensive surveys in [105, 121].

Applications

It would be futile to even try to enumerate all applications of PSO that have been published so far. From 2005 and on, more than 400 papers with PSO's applications appear every year, spanning various scientific and technological fields. Electrical Engineering concentrates the majority of these works, especially in the fields of power systems, control, antenna design, electromagnetics, sensors, networks and communications. Artificial Intelligence also hosts a large number of PSO-based applications, especially in robotics, machine learning, and data mining. Bioinformatics and Operations Research follow closely, with numerous works in modeling, health-care systems, scheduling, routing, supply chain management, and forecasting.

A number of applications is cited in the previous sections. In addition, the interested reader can refer to devoted survey papers such as [112], which was probably the first notable attempt to collect and categorize PSO's applications. An analytical survey was published in [127], where a huge number of PSO-based applications was categorized along with more than 100 PSO variants. Further applications are reported in relevant books such as [106]. The Appendix at the end of the present work contains a number of sources for further inquiry on PSO-based developments.

Conclusions

PSO has been established as one of the most popular metaheuristic optimization algorithms. Its popularity emanates from its nice performance and adequate simplicity that renders it usable even by non-expert researchers. In the previous sections, a number of variants and improvements were presented. This is only a small fraction of the existing PSO-related literature, which counts thousands of papers. Thus, a reasonable question can be put regarding the room left for further developments on PSO. In the author's opinion, the answer is: a lot.

Despite the numerous research contributions, there are still many issues that need improvement to achieve the main goal of intelligent behavior and self-adaptation. Moreover, the evolution of computer and web technologies always introduces new challenges on the design and development of algorithms that can take full advantage of their properties to solve problems of increasing complexity. For example, the GPU computing paradigm and modern multicore desktop systems can offer computational power comparable to small- and medium-size clusters. Cloud computing and ubuquitous computing environments are other examples. Also, new ad hoc operators and procedures for specific problems are expected to boost PSO's performance.

Closing this chapter, the author would like to quote Albert Einstein's words to motivate new researchers toward unconventional thinking, which was the main ingredient for the development of PSO so far:

> You will never solve problems using the same thinking you created them with.

Cross-References

- ▸ Adaptive and Multilevel Metaheuristics
- ▸ Ant Systems
- ▸ Automatic Tuning of Parameters
- ▸ General Concepts of Metaheuristics
- ▸ Hyper Heuristics
- ▸ Memetic Algorithms
- ▸ Multiobjective Optimization

- Multi Start Methods
- Parallel Search
- Statistical Tests to Compare Heuristics

Appendix

A number of sources for further inquiry and experimentation with PSO is reported below.
Books
[19, 34, 62–64, 86, 93, 106, 134]
Survey papers
[4, 6, 7, 35, 112, 114, 122, 123, 127, 141]
Webpages
Particle Swarm Central
http://www.particleswarm.info/
M. Clerc's PSO page
http://clerc.maurice.free.fr/pso/
Software
PSO in C (code published in [149])
http://www.cpc.cs.qub.ac.uk/summaries/AELM_v1_0.html
PSO in Matlab
http://www.mathworks.com/matlabcentral/fileexchange/7506
http://psotoolbox.sourceforge.net/
PSO in Java
http://jswarm-pso.sourceforge.net/
http://gundog.lbl.gov/GO/jdoc/genopt/algorithm/PSOCC.html

References

1. Abido MA (2010) Multiobjective particle swarm optimization with nondominated local and global sets. Nat Comput 9(3):747–766
2. Agrawal S, Panigrahi BK, Tiwari MK (2008) Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch. IEEE Trans Evol Comput 12(5):529–541
3. Ahmadi MA (2012) Neural network based unified particle swarm optimization for prediction of asphaltene precipitation. Fluid Phase Equilib 314:46–51
4. Aote AS, Raghuwanshi MM, Malik L (2013) A brief review on particle swarm optimization: limitations & future directions. Int J Comput Sci Eng 2(5):196–200
5. Aziz M, Tayarani-N M-H (2014) An adaptive memetic particle swarm optimization algorithm for finding large-scale latin hypercube designs. Eng Appl Artif Intell 36:222–237
6. Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part i: background and development. Nat Comput 6(4):467–484
7. Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124

8. T. Bartz-Beielstein, Blum D, Branke J (2007) Particle swarm optimization and sequential sampling in noisy environments. In: Doerner KF et al (ed) Metaheuristics: progress in complex systems optimization. Operations research/computer science interfaces series, vol 39. Springer, New York, pp 261–273
9. Bin W, Qinke P, Jing Z, Xiao C (2012) A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior. Eur J Oper Res 219(2):224–233
10. Blackwell T (2012) A study of collapse in bare bones particle swarm optimization. IEEE Trans Evol Comput 16(3):354–372
11. Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput J 11(6):4135–4151
12. Bonabeau E, Dorigo M, Théraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, New York
13. Bonyadi MR, Michalewicz Z (2014) SPSO2011 – analysis of stability, local convergence, and rotation sensitivity. In: GECCO 2014 – proceedings of the 2014 genetic and evolutionary computation conference, Vancouver, pp 9–15
14. Camci F (2009) Comparison of genetic and binary particle swarm optimization algorithms on system maintenance scheduling using prognostics information. Eng Optim 41(2):119–136
15. Chauhan P, Deep K, Pant M (2013) Novel inertia weight strategies for particle swarm optimization. Memet Comput 5(3):229–251
16. Chen C-H, Lin J, Yücesan E, Chick SE (2000) Simulation budget allocation for further enhancing the efficiency of ordinal optimization. Discr Event Dyn Syst Theory Appl 10(3):251–270
17. Chen J, Yang D, Feng Z (2012) A novel quantum particle swarm optimizer with dynamic adaptation. J Comput Inf Syst 8(12):5203–5210
18. Chen Z, He Z, Zhang C (2010) Particle swarm optimizer with self-adjusting neighborhoods. In: Proceedings of the 12th annual genetic and evolutionary computation conference (GECCO 2010), Portland, pp 909–916
19. Clerc M (2006) Particle swarm optimization. ISTELtd, London
20. Clerc M (2012) Standard particle swarm optimization. Technical report 2012, Particle Swarm Central
21. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73
22. Coelho LdS (2008) A quantum particle swarm optimizer with chaotic mutation operator. Chaos Solitons Fractals 37(5):1409–1418
23. Coello Coello CA (1999) Self-adaptive penalties for GA-based optimization. In: Proceedings of the 1999 IEEE congress on evolutionary computation, Washington, vol 1, pp 573–580
24. Coello Coello CA, Van Veldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, New York
25. Cooren Y, Clerc M, Siarry P (2008) Initialization and displacement of the particles in TRIBES, a parameter-free particle swarm optimization algorithm. Stud Comput Intell 136:199–219
26. Cooren Y, Clerc M, Siarry P (2009) Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm. Swarm Intell 3(2):149–178
27. Cooren Y, Clerc M, Siarry P (2011) MO-TRIBES, an adaptive multiobjective particle swarm optimization algorithm. Comput Optim Appl 49(2):379–400
28. Dai Y, Liu L, Feng S (2014) On the identification of coupled pitch and heave motions using opposition-based particle swarm optimization. Math Probl Eng 2014(3):1–10
29. Daneshyari M, Yen GG (2011) Cultural-based multiobjective particle swarm optimization. IEEE Trans Syst Man Cybern Part B Cybern 41(2):553–567
30. Daoudi M, Boukra A, Ahmed-Nacer M (2011) Adapting TRIBES algorithm for traveling salesman problem. In: Proceedings of the 10th international symposium on programming and systems (ISPS' 2011), pp 163–168
31. Davarynejad M, Van Den Berg J, Rezaei J (2014) Evaluating center-seeking and initialization bias: the case of particle swarm and gravitational search algorithms. Inf Sci 278:802–821

32. Dos Santos Coelho L, Ayala HVH, Alotto P (2010) A multiobjective gaussian particle swarm approach applied to electromagnetic optimization. IEEE Trans Mag 46(8):3289–3292
33. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings sixth symposium on micro machine and human science, Piscataway, pp 39–43. IEEE Service Center
34. Engelbrecht AP (2006) Fundamentals of computational swarm intelligence. Wiley, Chichester
35. Eslami M, Shareef H, Khajehzadeh M, Mohamed A (2012) A survey of the state of the art in particle swarm optimization. R J Appl Sci Eng Technol 4(9):1181–1197
36. Gao W-F, Liu S-Y, Huang L-L (2012) Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. Commun Nonlinear Sci Numer Simul 17(11):4316–4327
37. Ge RP, Qin YF (1987) A class of filled functions for finding global minimizers of a function of several variables. J Optim Theory Appl 54:241–252
38. Gholipour R, Khosravi A, Mojallali H (2013) Suppression of chaotic behavior in duffing-holmes system using backstepping controller optimized by unified particle swarm optimization algorithm. Int J Eng Trans B Appl 26(11):1299–1306
39. Gholizadeh S, Moghadas R (2014) Performance-based optimum design of steel frames by an improved quantum particle swarm optimization. Adv Struct Eng 17(2):143–156
40. Goudos SK, Moysiadou V, Samaras T, Siakavara K, Sahalos JN (2010) Application of a comprehensive learning particle swarm optimizer to unequally spaced linear array synthesis with sidelobe level suppression and null control. IEEE Antennas Wirel Propag Lett 9:125–129
41. He G, Wu B (2014) Unified particle swarm optimization with random ternary variables and its application to antenna array synthesis. J Electromag Waves Appl 28(6):752–764
42. He J, Dai H, Song X (2014) The combination stretching function technique with simulated annealing algorithm for global optimization. Optim Methods Softw 29(3):629–645
43. Hu Z, Bao Y, Xiong T (2014) Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression. Appl Soft Comput J 25:15–25
44. Huang K-W, Chen J-L, Yang C-S, Tsai C-W (2015) A memetic particle swarm optimization algorithm for solving the dna fragment assembly problem. Neural Comput Appl 26(3): 495–506
45. Jamalipour M, Gharib M, Sayareh R, Khoshahval F (2013) PWR power distribution flattening using quantum particle swarm intelligence. Ann Nucl Energy 56:143–150
46. Janson S, Middendorf M (2004) A hierarchical particle swarm optimizer for dynamic optimization problems. Lecture notes in computer science, vol 3005. Springer, Berlin/New York, pp 513–524
47. Janson S, Middendorf M (2006) A hierarchical particle swarm optimizer for noisy and dynamic environments. Genet Program Evol Mach 7(4):329–354
48. Jiang B, Wang N (2014) Cooperative bare-bone particle swarm optimization for data clustering. Soft Comput 18(6):1079–1091
49. Jiang M, Luo YP, Yang SY (2007) Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Inf Process Lett 102:8–16
50. Jiao B, Yan S (2011) A cooperative co-evolutionary quantum particle swarm optimizer based on simulated annealing for job shop scheduling problem. Int J Artif Intell 7(11 A):232–247
51. Jin N, Rahmat-Samii Y (2007) Advances in particle swarm optimization for antenna designs: real-number, binary, single-objective and multiobjective implementations. IEEE Trans Antennas Propag 55(3 I):556–567
52. Jin N, Rahmat-Samii Y (2010) Hybrid real-binary particle swarm optimization (HPSO) in engineering electromagnetics. IEEE Trans Antennas Propag 58(12):3786–3794
53. Jin Y, Olhofer M, Sendhoff B (2001) Evolutionary dynamic weighted aggregation for multiobjective optimization: why does it work and how? In: Proceedings GECCO 2001 conference, San Francisco, pp 1042–1049
54. Kadirkamanathan V, Selvarajah K, Fleming PJ (2006) Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Trans Evol Comput 10(3):245–255

55. Kaucic M (2013) A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. J Glob Optim 55(1):165–188
56. Kennedy J (1998) The behavior of particles. In: Porto VW, Saravanan N, Waagen D, Eiben AE (eds) Evolutionary programming, vol VII. Springer, Berlin/New York, pp 581–590
57. Kennedy J (1999) Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the IEEE congress on evolutionary computation, Washington, DC. IEEE Press, pp 1931–1938
58. Kennedy J (2003) Bare bones particle swarms. In: Proceedings of the IEEE swarm intelligence symposium, Indianapolis. IEEE Press, pp 80–87
59. Kennedy J (2010) Particle swarm optimization. In: Sammut C, Webb G (eds) Encyclopedia of machine learning. Springer, Boston, pp 760–766
60. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceeding of the IEEE international conference neural networks, Piscataway, vol IV. IEEE Service Center, pp 1942–1948
61. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of the conference on systems, man and cybernetics, Hyatt Orlando, pp 4104–4109
62. Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann Publishers, San Francisco
63. Kiranyaz S, Ince T, Gabbouj M (2014) Multidimensional particle swarm optimization for machine learning and pattern recognition. Springer, Berlin
64. Kishk A (2008) Particle swarm optimizaton: a physics-based approach. Morgan and Claypool Publishers, Arizona
65. Kotsireas IS, Koukouvinos C, Parsopoulos KE, Vrahatis MN (2006) Unified particle swarm optimization for Hadamard matrices of Williamson type. In: Proceedings of the 1st international conference on mathematical aspects of computer and information sciences (MACIS 2006), Beijing, pp 113–121
66. Krohling RA, Campos M, Borges P (2010) Bare bones particle swarm applied to parameter estimation of mixed weibull distribution. Adv Intell Soft Comput 75:53–60
67. Kwok NM, Ha QP, Liu DK, Fang G, Tan KC (2007) Efficient particle swarm optimization: a termination condition based on the decision-making approach. In: Proceedings of the 2007 IEEE congress on evolutionary computation (CEC 2007), Singapore, pp 3353–3360
68. Laskari EC, Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization for integer programming. In: Proceedings of the IEEE 2002 congress on evolutionary computation (IEEE CEC 2002), Honolulu. IEEE Press, pp 1582–1587
69. Lawler EL, Wood DW (1966) Branch and bound methods: a survey. Oper Res 14:699–719
70. Li X (2007) A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio. ACM, New York, pp 78–85
71. Li X (2010) Niching without Niching parameters: Particle swarm optimization using a ring topology. IEEE Trans Evol Comput 14(1):150–169
72. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295
73. Likas A, Blekas K, Stafylopatis A (1996) Parallel recombinative reinforcement learning: a genetic approach. J Intell Syst 6(2):145–169
74. Liu B-F, Chen H-M, Chen J-H, Hwang S-F, Ho S-Y (2005) MeSwarm: memetic particle swarm optimization. ACM, New York, pp 267–268
75. Liu DS, Tan KC, Huang SY, Goh CK, Ho WK (2008) On solving multiobjective bin packing problems using evolutionary particle swarm optimization. Eur J Oper Res 190(2):357–382
76. Liu R, Zhang P, Jiao L (2014) Quantum particle swarm optimization classification algorithm and its applications. Int J Pattern Recognit Artif Intell 28(2)
77. Lv L, Wang H, Li X, Xiao X, Zhang L (2014) Multi-swarm particle swarm optimization using opposition-based learning and application in coverage optimization of wireless sensor network. Sensor Lett 12(2):386–391

78. Magoulas GD, Vrahatis MN, Androulakis GS (1997) On the alleviation of local minima in backpropagation. Nonlinear Anal Theory Methods Appl 30(7):4545–4550
79. Manquinho VM, Marques Silva JP, Oliveira AL, Sakallah KA (1997) Branch and bound algorithms for highly constrained integer programs. Technical report, Cadence European Laboratories, Portugal
80. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8(3):204–210
81. Mikki SM, Kishk AA (2006) Quantum particle swarm optimization for electromagnetics. IEEE Trans Antennas Propag 54(10):2764–2775
82. Moustaki E, Parsopoulos KE, Konstantaras I, Skouri K, Ganas I (2013) A first study of particle swarm optimization on the dynamic lot sizing problem with product returns. In: XI Balkan conference on operational research (BALCOR 2013), Belgrade, pp 348–356
83. Nanda B, Maity D, Maiti DK (2014) Crack assessment in frame structures using modal data and unified particle swarm optimization technique. Adv Struct Eng 17(5):747–766
84. Nanda B, Maity D, Maiti DK (2014) Modal parameter based inverse approach for structural joint damage assessment using unified particle swarm optimization. Appl Math Comput 242:407–422
85. Neri F, Cotta C (2012) Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol Comput 2:1–14
86. Olsson AE (ed) (2011) Particle swarm optimization: theory, techniques and applications. Nova Science Pub Inc., New York
87. Ozcan E, Mohan CK Analysis of a simple particle swarm optimization. In: Intelligent engineering systems through artificial neural networks, vol 8. ASME Press, New York, pp 253–258
88. Ozcan E, Mohan CK (1999) Particle swarm optimization: surfing the waves. In: Proceedings of the 1999 IEEE international conference on evolutionary computation, Washington, DC, pp 1939–1944
89. Padhye N, Deb K, Mittal P (2013) Boundary handling approaches in particle swarm optimization. In: Proceedings of seventh international conference on bio-inspired computing: theories and applications (BIC-TA 2012), Gwalior, vol 201, pp 287–298
90. Pan F, Hu X, Eberhart R, Chen Y (2008) An analysis of bare bones particle swarm. In: Proceedings of the 2008 IEEE swarm intelligence symposium, St. Louis
91. Pan H, Wang L, Liu B (2006) Particle swarm optimization for function optimization in noisy environment. Appl Math Comput 181(2):908–919
92. Pandremmenou K, Kondi LP, Parsopoulos KE, Bentley ES (2014) Game-theoretic solutions through intelligent optimization for efficient resource management in wireless visual sensor networks. Signal Process Image Commun 29(4):472–493
93. Parasuraman D (2012) Handbook of particle swarm optimization: concepts, principles & applications. Auris reference, Nottingham
94. Parrott D, Li X (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans Evol Comput 10(4):440–458
95. Parsopoulos KE, Plagianakos VP, Magoulas GD, Vrahatis MN (2001) Objective function "stretching" to alleviate convergence to local minima. Nonlinear Anal Theory Methods Appl 47(5):3419–3424
96. Parsopoulos KE, Tasoulis DK, Vrahatis MN (2004) Multiobjective optimization using parallel vector evaluated particle swarm optimization. In: Hamza MH (ed) Proceedings of the IASTED 2004 international conference on artificial intelligence and applications (AIA 2004), Innsbruck, vol 2. IASTED/ACTA Press, pp 823–828
97. Parsopoulos KE, Vrahatis MN (2001) Particle swarm optimizer in noisy and continuously changing environments. In: Hamza MH (ed) Artificial intelligence and soft computing. IASTED/ACTA Press, Anaheim, pp 289–294
98. Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization method for constrained optimization problems. In: Sincak P, Vascak J, Kvasnicka V, Pospichal J (eds) Intelligent

technologies-theory and application: new trends in intelligent technologies. Frontiers in artificial intelligence and applications, vol 76. IOS Press, pp 214–220
99. Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization method in multiobjective problems. In: Proceedings of the ACM 2002 symposium on applied computing (SAC 2002), Madrid. ACM Press, pp 603–607
100. Parsopoulos KE, Vrahatis MN (2002) Recent approaches to global optimization problems through particle swarm optimization. Nat Comput 1(2-3):235–306
101. Parsopoulos KE, Vrahatis MN (2004) On the computation of all global minimizers through particle swarm optimization. IEEE Trans Evol Comput 8(3):211–224
102. Parsopoulos KE, Vrahatis MN (2004) UPSO: a unified particle swarm optimization scheme. In: Proceedings of the international conference of computational methods in sciences and engineering (ICCMSE 2004). Lecture series on computer and computational sciences, vol 1. VSP International Science Publishers, Zeist, pp 868–873
103. Parsopoulos KE, Vrahatis MN (2006) Studying the performance of unified particle swarm optimization on the single machine total weighted tardiness problem. In: Sattar A, Kang BH (eds) Lecture notes in artificial intelligence (LNAI), vol 4304. Springer, Berlin/New York, pp 760–769
104. Parsopoulos KE, Vrahatis MN (2007) Parameter selection and adaptation in unified particle swarm optimization. Math Comput Model 46(1–2):198–213
105. Parsopoulos KE, Vrahatis MN (2008) Multi-objective particles swarm optimization approaches. In Bui LT, Alam S (eds) Multi-objective optimization in computational intelligence: theory and practice. Premier reference source, chapter 2. Information Science Reference (IGI Global), Hershey, pp 20–42
106. Parsopoulos KE, Vrahatis MN (2010) Particle swarm optimization and intelligence: advances and applications. Inf Sci Publ (IGI Glob)
107. Petalas YG, Parsopoulos KE, Vrahatis MN (2007) Entropy-based memetic particle swarm optimization for computing periodic orbits of nonlinear mappings. In: IEEE 2007 congress on evolutionary computation (IEEE CEC 2007), Singapore. IEEE Press, pp 2040–2047
108. Petalas YG, Parsopoulos KE, Vrahatis MN (2007) Memetic particle swarm optimization. Ann Oper Res 156(1):99–127
109. Petalas YG, Parsopoulos KE, Vrahatis MN (2009) Improving fuzzy cognitive maps learning through memetic particle swarm optimization. Soft Comput 13(1):77–94
110. Piperagkas GS, Georgoulas G, Parsopoulos KE, Stylios CD, Likas CA (2012) Integrating particle swarm optimization with reinforcement learning in noisy problems. In: Genetic and evolutionary computation conference 2012 (GECCO 2012), Philadelphia. ACM, pp 65–72
111. Piperagkas GS, Konstantaras I, Skouri K, Parsopoulos KE (2012) Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. Comput Oper Res 39(7):1555–1565
112. Poli R (2008) Analysis of the publications on the applications of particle swarm optimisation. J Artif Evol Appl 2008(3):1–10
113. Poli R (2008) Dynamic and stability of the sampling distribution of particle swarm optimisers via moment analysis. J Artif Evol Appl 2008(3):10010
114. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1(1):33–57
115. Poli R, Langdon WB (2007) Markov chain models of bare-bones particle swarm optimizers. ACM, New York, pp 142–149
116. Pookpunt S, Ongsakul W (2013) Optimal placement of wind turbines within wind farm using binary particle swarm optimization with time-varying acceleration coefficients. Renew Energy 55:266–276
117. Potter MA, De Jong K (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29
118. Qu BY, Liang JJ, Suganthan PN (2012) Niching particle swarm optimization with local search for multi-modal optimization. Inf Sci 197:131–143
119. Rada-Vilela J, Johnston M, Zhang M (2014) Population statistics for particle swarm optimization: resampling methods in noisy optimization problems. Swarm Evol Comput 17:37–59

120. Rahnamayan RS, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79
121. Reyes-Sierra M, Coello Coello CA (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2(3):287–308
122. Rezaee Jordehi A, Jasni J (2013) Parameter selection in particle swarm optimisation: a survey. J Exp Theor Artif Intell 25(4):527–542
123. Rini DP, Shamsuddin SM, Yuhaniz SS (2014) Particle swarm optimization: technique, system and challenges. Int J Comput Appl 14(1):19–27
124. Schmitt M, Wanka R (2015) Particle swarm optimization almost surely finds local optima. Theor Comput Sci Part A 561:57–72
125. Schoeman IL, Engelbrecht AP (2010) A novel particle swarm niching technique based on extensive vector operations. Nat Comput 9(3):683–701
126. Schwefel H-P (1995) Evolution and optimum seeking. Wiley, New York
127. Sedighizadeh D, Masehian E (2009) Particle swarm optimization methods, taxonomy and applications. Int J Comput Theory Eng 1(5):486–502
128. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings IEEE conference on evolutionary computation, Anchorage. IEEE Service Center, pp 69–73
129. Skokos Ch, Parsopoulos KE, Patsis PA, Vrahatis MN (2005) Particle swarm optimization: an efficient method for tracing periodic orbits in 3D galactic potentials. Mon Not R Astron Soc 359:251–260
130. Souravlias D, Parsopoulos KE (2016) Particle swarm optimization with neighborhood-based budget allocation. Int J Mach Learn Cybern 7(3):451–477. Springer
131. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
132. Suganthan PN (1999) Particle swarm optimizer with neighborhood operator. In: Proceedings of the IEEE congress on evolutionary computation, Washington, DC, pp 1958–1961
133. Sun J, Feng B, Xu W (2004) Particle swarm optimization with particles having quantum behavior. In: Proceedings of the IEEE congress on evolutionary computation 2004 (IEEE CEC'04), Portland (OR), pp 325–331
134. Sun J, Lai C-H, Wu X-J (2011) Particle swarm optimisation: classical and quantum perspectives. CRC Press, Boca Raton
135. Sun J, Xu W, Feng B (2004) A global search strategy for quantum-behaved particle swarm optimization. In: Proceedings of the 2004 IEEE conference on cybernetics and intelligent systems, Singapore, pp 111–116
136. Sun S, Li J (2014) A two-swarm cooperative particle swarms optimization. Swarm Evol Comput 15:1–18
137. Sutton AM, Whitley D, Lunacek M, Howe A (2006) PSO and multi-funnel landscapes: how cooperation might limit exploration. In: Proceedings of the 8th annual conference on genetic and evolutionary computation (GECCO'06), Seattle, pp 75–82
138. Tasgetiren F, Chen A, Gencyilmaz G, Gattoufi S (2009) Smallest position value approach. Stud Comput Intell 175:121–138
139. Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G (2006) Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res 44(22):4737–4754
140. Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177(3):1930–1947
141. Thangaraj R, Pant M, Abraham A, Bouvry P (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl Math Comput 217(12):5208–5226
142. Törn A, Žilinskas A (1989) Global optimization. Springer, Berlin
143. Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85:317–325
144. Tsai H-C (2010) Predicting strengths of concrete-type specimens using hybrid multilayer perceptrons with center-unified particle swarm optimization. Expert Syst Appl 37(2): 1104–1112

145. Van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimiser. In: Proceedings of the 2002 IEEE international conference on systems, man and cybernetics, vol 3, pp 94–99
146. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
147. Van den Bergh F, Engelbrecht AP (2006) A study of particle swarm optimization particle trajectories. Inf Sci 176:937–971
148. Voglis C, Parsopoulos KE, Lagaris IE (2012) Particle swarm optimization with deliberate loss of information. Soft Comput 16(8):1373–1392
149. Voglis C, Parsopoulos KE, Papageorgiou DG, Lagaris IE, Vrahatis MN (2012) MEMPSODE: a global optimization software based on hybridization of population-based algorithms and local searches. Comput Phys Commun 183(5):1139–1154
150. Wang H, Moon I, Yang S, Wang D (2012) A memetic particle swarm optimization algorithm for multimodal optimization problems. Inf Sci 197:38–52
151. Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M (2011) Enhancing particle swarm optimization using generalized opposition-based learning. Inf Sci 181(20):4699–4714
152. Wang H, Zhao X, Wang K, Xia K, Tu X (2014) Cooperative velocity updating model based particle swarm optimization. Appl Intell 40(2):322–342
153. Wang Y-J, Zhang J-S (2008) A new constructing auxiliary function method for global optimization. Math Comput Modell 47(11–12):1396–1410
154. Wu H, Geng J, Jin R, Qiu J, Liu W, Chen J, Liu S (2009) An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas. IEEE Trans Antennas Propag 57(10 PART 2):3018–3028
155. Xianfeng Y, Li LS (2014) Dynamic adjustment strategies of inertia weight in particle swarm optimization algorithm. Int J Control Autom 7(5):353–364
156. Xu W, Duan BY, Li P, Hu N, Qiu Y (2014) Multiobjective particle swarm optimization of boresight error and transmission loss for airborne radomes. IEEE Trans Antennas Propag 62(11):5880–5885
157. Xue B, Zhang M, Browne WN (2014) Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl Soft Comput J 18: 261–276
158. Yang J, Zhang H, Ling Y, Pan C, Sun W (2014) Task allocation for wireless sensor network using modified binary particle swarm optimization. IEEE Sens J 14(3):882–892
159. Yang J-M, Chen Y-P, Horng J-T, Kao C-Y (1997) Applying family competition to evolution strategies for constrained optimization. Lecture notes in mathematics, vol 1213. Springer, Berlin/New York, pp 201–211
160. Yen GG, Leong WF (2009) Dynamic multiple swarms in multiobjective particle swarm optimization. IEEE Trans Syst Man Cybern Part A Syst Hum 39(4):890–911
161. Yu X, Zhang X (2014) Enhanced comprehensive learning particle swarm optimization. Appl Math Comput 242:265–276
162. Zambrano-Bigiarini M, Clerc M, Rojas R (2013) Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements. In: 2013 IEEE congress on evolutionary computation, Cancún, pp 2337–2344
163. Zhang Q, Wang Z, Tao F, Sarker BR, Cheng L (2014) Design of optimal attack-angle for RLV reentry based on quantum particle swarm optimization. Adv Mech Eng 6:352983
164. Zhang Y, Gong D, Hu Y, Zhang W (2015) Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing 148:150–157
165. Zhang Y, Gong D-W, Ding Z (2012) A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch. Inf Sci 192:213–227
166. Zhang Y, Gong D-W, Sun X-Y, Geng N (2014) Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis. Soft Comput 18(7):1337–1352
167. Zhao F, Li G, Yang C, Abraham A, Liu H (2014) A human-computer cooperative particle swarm optimization based immune algorithm for layout design. Neurocomputing 132: 68–78

168. Zhao J, Lv L, Fan T, Wang H, Li C, Fu P (2014) Particle swarm optimization using elite opposition-based learning and application in wireless sensor network. Sens Lett 12(2): 404–408
169. Zheng Y-J, Ling H-F, Xue J-Y, Chen S-Y (2014) Population classification in fire evacuation: a multiobjective particle swarm optimization approach. IEEE Trans Evol Comput 18(1):70–81