Swarm and Evolutionary Computation 63 (2021) 100868

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Survey Paper

Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application

Essam H. Houssein [a,*], Ahmed G. Gad [b], Kashif Hussain [c], Ponnuthurai Nagaratnam Suganthan [d]

[a] Faculty of Computers and Information, Minia University, Minia, Egypt
[b] Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh, Egypt
[c] Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu, China
[d] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Meta-heuristics
Optimization
Particle Swarm Optimization (PSO)
Parameter adaption
Parameter tuning
Neighborhood topology
Fitness landscape analysis
Complex optimization problems
Large-scale optimization
Constrained Optimization Problems (COPs)
Multi-objective Optimization Problems (MOPs)
Multimodal optimization
Surrogate-assisted optimization
Computationally expensive optimization
Ensembles
Bare-bones optimization
Rough optimization
Quantum-behaved optimization
Hyper-heuristics
Parallelized optimization
Real-world applications

ABSTRACT

Over the ages, nature has constantly been a rich source of inspiration for science, with much still to discover about and learn from. Swarm Intelligence (SI), a major branch of artificial intelligence, was rendered to model the collective behavior of social swarms in nature. Ultimately, Particle Swarm Optimization algorithm (PSO) is arguably one of the most popular SI paradigms. Over the past two decades, PSO has been applied successfully, with good return as well, in a wide variety of fields of science and technology with a wider range of complex optimization problems, thereby occupying a prominent position in the optimization field. However, through in-depth studies, a number of problems with the algorithm have been detected and identified; e.g., issues regarding convergence, diversity, and stability. Consequently, since its birth in the mid-1990s, PSO has witnessed a myriad of enhancements, extensions, and variants in various aspects of the algorithm, specifically after the twentieth century, and the related research has therefore now reached an impressive state. In this paper, a rigorous yet systematic review is presented to organize and summarize the information on the PSO algorithm and the developments and trends of its most basic as well as of some of the very notable implementations that have been introduced recently, bearing in mind the coverage of paradigm, theory, hybridization, parallelization, complex optimization, and the diverse applications of the algorithm, making it more accessible. Ease for researchers to determine which PSO variant is currently best suited or to be invented for a given optimization problem or application. This up-to-date review also highlights the current pressing issues and intriguing open challenges haunting PSO, prompting scholars and researchers to conduct further research both on the theory and application of the algorithm in the forthcoming years.

1. Introduction

Swarm algorithms are increasingly used to optimize different types of problems in various fields. Without many technical assumptions, these algorithms are often, with easy implementation, able to effectively solve complex optimization problems. For example, it is not crucial to have an objective function that is differentiable or separable. Due to their flexibility, these algorithms have the potential to tackle various real-world optimization problems increasingly in different disciplines, especially engineering and computer science.

Particle Swarm Optimization (PSO), proposed in [1,2], is a well-known swarm-based stochastic algorithm inspired by nature and originally developed by Russell C. Eberhart, an electrical engineer, and James Kennedy, a social psychologist, based on a simplified model of bird flocking behavior. In PSO, a group of particles (e.g., a flock of birds) performs the search process in the problem space. Each particle then shares, with some random perturbations, its current best position (and its fitness value) with those of one or more particles of the swarm to determine its next move throughout the search space, ultimately based on the historical pathway of each particle as well as the trajectory of the whole swarm. Once all particles have their respective positions updated at the current iteration, the next iteration of the search process starts, pursing regions near the perceived optimum. Eventually, the whole swarm probably approaches the objective function optimum at a convergence speed that highly depends on the PSO variation invented and the values assigned to the control parameters [3].

* Corresponding author.
E-mail addresses: essam.halim@mu.edu.eg (E.H. Houssein), ahmed.gad@fci.kfs.edu.eg (A.G. Gad), k.hussain@uestc.edu.cn (K. Hussain), epnsugan@ntu.edu.sg (P.N. Suganthan).

https://doi.org/10.1016/j.swevo.2021.100868
Received 10 March 2020; Received in revised form 11 February 2021; Accepted 28 February 2021
Available online 8 March 2021

Due to its simplicity, PSO has gained wide appeal among researchers and has thus been widely publicized to provide efficient performance in a variety of application fields. Moreover, its potential to hybridize and specialize, as well as to demonstrate an emergent behavior within high-dimensional search spaces for the discovery of feasible regions [1,2] has made this algorithm a common choice. Nevertheless, apart from the major problem of velocity explosion that the original PSO version encountered, PSO both in its basic [1,2] and inertia models [4] has one limitation: it may not sufficiently explore the research space and thus be prone to premature convergence [5,6] by falling into a local optimum [7,8]. However, for the inertia PSO, depending on the values assigned to the control parameters, the particles may still diverge and not even converge, and when "converged", this may not be to a single point in the search space, but simply to an equilibrium state where the particles stop moving. An effective algorithm should have both good exploration and exploitation properties so that, when it is in proximity, the optimum is located. The two competing objectives are frequently compromised because algorithms which sufficiently explore the space usually struggle to exploit promising areas for determining the optimum, while aggressively exploitative algorithms can easily get trapped into a local optimum. Typically, a few iterations for exploration are needed in PSO before proceeding to the exploitation process [9], possibly decreasing the quality of the solution. The choice of improper control parameters or the strong connection with the the personal and global best locations can be the underlying causes of the premature convergence phenomenon, which may arise by exerting an undue effect and not changing frequently enough during iterations, see, for example, [10,11]. Moreover, it has been revealed that, in a high-dimensional search space where there are many variables to optimize, the issue of premature convergence is encountered by particles, more problematically [12,13]. Consequently, since its advent in 1995, these disadvantages as well as others emerging have been circumvented by researchers through developing various novel Particle Swarm Optimizers (PSOs).

Some of the PSO variants derived benefits from the capabilities of Evolutionary Algorithms (EAs) or other social meta-heuristics, producing hybrid PSO algorithms. The researchers in the PSO community have taken on incorporating popular evolutionary computational paradigms, such as Differential Evolution (DE), and the operators of crossover, mutation, and selection, as in Genetic Algorithms (GAs), into PSO. Such improvements have shown encouraging results and typically helped in avoiding local optima entrapment. However, the particles in PSO often experience divergence and leave the search space when tackling high-dimensional complex problems, exhibiting what is dubbed "roaming" behavior [14], even if the algorithm is able to escape from the local optima. Although PSO was originally implemented to solve numerical boundary-constrained, multi-dimensional optimization problems with a single objective function [1,2], apart from other various types of problems in the optimization domain, it does not necessarily, for example, work properly in high-dimensional models [12,13]. To address such issues, several approaches to PSO implementation have been suggested, along with testing them in different complex optimization scenarios, especially over the past two decades. In this review paper, we intend to summarize and analyze some of the most influential yet versatile developments in various aspects of the PSO algorithm since its original formulation in 1995, giving special emphasis on the post-twentieth century period.

1.1. Motivation

Over the years, PSO has been often the subject of study compared to its Swarm Intelligence (SI) based counterparts [15]. This is perhaps due to the efficiency of the algorithm and its straightforwardness among the scientific community. To cover scientific contributions with respect to PSO, several reviews and meta-analyses have been performed on PSO in various application domains. Banks et al. [16] offered, in two parts, a timely and brief review of different improvements to the original formulation of PSO and the opportunities as well as challenges emanating from the PSO algorithm. Furthermore, the review focused on research regarding adaptations for parallel implementation, algorithm configuration, and dynamic environments. The study in its first part focused on particles' stagnation in dynamic environments. The shortcoming of this part is the insufficiency in explaining the related works. On the other hand, the later part in [17] discussed recent studies in some more complex effective areas of research: constrained optimization, multi-objective optimization, combinatorial optimization problems, and hybridization. Nevertheless, this study has a major defect of not considering the swarm-based algorithms' quality metrics (i.e., stagnation, convergence, and diversity). In connection with PSO applications, Poli et al. [18] presented a summary of the great efforts that have added impetus and new directions to research in particle swarms, along with some important new applications and directions. The strength of this study lies in its presentation of comprehensive challenges and open issues in the PSO algorithm; however, the compatibility of the PSO applications with each approach presented remains ignored. In this regard, Vipul et al. [19] discussed PSO specifically with respect to its practical applications to real-world problems.

The PSO researchers have also highlighted some of the important modifications to PSO. For example, Raghavendra and Ganesh [20] outlined issues concerning Wireless Sensor Networks (WSNs) applications and investigated PSO suitability for such applications. However, the compatibility of high-speed real-time applications with each approach was not considered in this review work. In another work, Alam et al. [21] systematically surveyed the evolution of clustering techniques based on PSO in various application domains, proving the tremendous increase in the popularity of such approaches; however, applications for truly complex problems were absent. When to talk about the PSO variants, Muhammad et al. [22] discussed the different variants of PSO regarding initialization, mutation operators, and Inertia Weight (IW). The study mainly contributes to highlighting different mutation operators and the IW parameter for performance improvement of PSO; however, it did not cover other prominent PSO variants. Wang et al. [23] presented the inception and background of the PSO algorithm and analyzed its current situation of application and research in algorithm structure, topology structure, parameter selection, multi-objective optimization, discrete and parallel PSO algorithm, and its engineering applications. This review is characterized by suggesting distinctive future research directions; however, it does not carry a comprehensive coverage of the PSO literature over the last two decades. Zhang et al. [24] investigated theoretically and practically the PSO algorithm by convergence analysis, parameter tuning, hybridization (with GA, DE, Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Harmonic Search (HS), Tabu Search (TS), Simulated Annealing (SA), and Biogeography-Based Optimization (BBO)), modifications (such as fuzzy PSO, chaotic PSO, bare-bones PSO, quantum-behaved PSO), neighborhood topology (such as fully-connected/all, ring, random, von Neumann, star, etc.), extensions (to binary, discrete, constrained, and multi-objective optimization), and parallel implementation (in cloud computing, multiprocessor, multicore, and GPU forms). To this point, they introduced a survey on the applications of PSO to the following eight fields: biology, chemistry, medicine, engineering, fuel and energy, operation research, communication theory, and automation control systems. One demerit with this study is that the theoretical and empirical analysis has not received sufficient attention. We have, in fact, duly captured this taxonomy and thus have plied with its approach in our survey, albeit more rigorously. Table 1 outlines the recent studies on PSO with respect to some of the PSO related surveys and review works. From the above-mentioned few current literature on PSO, the respective deficiencies propose that a novel, innovative, systematic literature review deems necessary to cover the historical and recent developments on the PSO family of algorithms, pursuing sometimes these literature studies, yet tackling their following weaknesses:

Table 1
PSO related survey studies.

Authors	Main context(s)	Publication year	Years covered
Poli et al. [18]	PSO algorithm, new directions, and applications	2007	1995–2006
Banks et al. [16,17]	PSO field, challenges, and opportunities	2007 & 2008	–
Kulkarni and Venayagamoorthy [20]	PSO suitability for WSN applications	2011	–
Imran et al. [22]	PSO variants	2013	–
Alam et al. [21]	PSO-based data clustering	2014	2002–2012
Zhang et al. [24]	PSO advances and applications	2015	2000–2013
Wang et al. [23]	PSO theory and application	2018	–

- Perspectives on PSO development with respect to swarm size and initialization, velocity initialization and clamping, neighborhood topologies, fitness landscape analysis, and also multimodal optimization, multi-objective optimization, discrete optimization, and PSO parallelization are not considered as a whole in a single previous study.
- Concepts and directions on choosing control parameters are not currently well covered.
- Variants of PSO addressed in existing review studies are not sufficiently subject to analytical assessment regarding swarm diversity, convergence behavior, and stagnation.
- There are still aspects of PSO search behavior (that we can predict) yet to be theoretically analyzed.
- Finally, a systematic treatment of the current literature of PSO is not very present in many of the previous relevant survey studies.

Adding to our motivation, it is common within the field of optimization to state that a particular algorithm, with a particular parameter configuration, has a particular behavior, without accounting for the various optimization problems that the algorithm might be applied to. While it is well known that the parameters of an algorithm must be tuned to problem instances for optimal performance, this qualification is rarely applied when addressing the behavior of algorithms. In fact, the influence of the characteristics of optimization problems on the behavior of algorithms is often ignored and still not well understood. Thus, this study intends to contribute to such an understanding by analyzing the fitness landscape and linking it to PSO.

On another side, the last few years have witnessed many optimization algorithms, especially swarm-based ones [25]. The PSO algorithm, being one of these algorithms, has been addressed and applied in numerous studies since its very inception in 1995, leaving out an overcrowded literary heritage as well as many critical issues related to convergence, stability, and diversity, yet it has achieved remarkable successes in many optimization applications [26–33]. This is also another one argument that this rigorous, systematic review study is a necessity with the aim of empowering the knowledge of researchers and practitioners already working on PSO, as well as providing guidance for those newcomers to the PSO community on how: i) to employ the PSO to solve complex optimization problems, ii) to point out the strengths and weaknesses of the algorithm against diverse optimization scenarios, and iii) to have a strong skill to emphasize the effectiveness of the algorithm compared to counterparts.

1.2. Objectives

The main objective of this study is to contribute to a deeper understanding of the algorithmic structure and the search behavior of PSO, in connection with major parameter adaption strategies and neighborhood topologies, in addition to awareness of the characteristics of different types of optimization problems against PSO variants and hybridization methods. Specifically, this study aims to gain a deep understanding of some of the PSO promising literature over the past two decades, keeping in mind the simplicity and fluidity in presenting and analyzing the PSO high-impact literature. To sum up, our main contributions are:

1. to know more about PSO's working principles and design;
2. to, with evidence, inform PSO researchers and practitioners of many common misconceptions and falsehoods that may hinder the successful use of PSO;
3. to summarize and organize the information on current PSO developments as well as analyze different proposals on the PSO initialization and upgradation methods, control parameter adaptation, swarm adaption, and neighborhood topologies;
4. to gain a better understanding of the influence of problem characteristics on the search behavior of PSO, using fitness landscape analysis;
5. to discuss various optimization scenarios built based on PSO, including large-scale, constrained, multimodal, multi-objective, discrete, surrogate model assisted, and also hybridization and synergy of the PSO with other optimizers;
6. to highlight the knowledge of the existing PSO theories, which will help develop a better intuition as well as greatly improve the effectiveness when working with PSO;
7. to present major applications of PSO and its variants in diverse real-world fields, including health-care, environmental, industrial, commercial, smart city, and miscellaneous; and finally,
8. to put forward a set of interesting open issues and future opportunities for the PSO community.

1.3. Outline

In Section 2, a brief description of the classical PSO and its basic concepts is presented, along with the original formulation of the algorithm. Section 3 demonstrates the various perspectives on development of the PSO paradigm, including different updating strategies, adaption mechanisms for swarm and control parameters, and fitness landscape analysis. Section 4 provides a good enough survey that summarizes some of the PSO applications to large-scale, constrained, multimodal, multi-objective, discrete, and surrogate-assisted optimization problems. Several prominent variants of PSO have been reviewed in Section 5, including parallel and distributed PSO schemes. Section 6 provides an overview of the PSO methods hybridized with other meta-heuristic algorithms as well as the synergy between the algorithm and some of the prominent machine learning techniques. Section 7 highlights some highly potent theoretical perspectives that should definitely guide PSO use. Some prominent yet diverse practices of PSO are summarized in Section 8. Section 9 presents a demonstration of the open challenges and future research lines that are yet to be touched for the PSO family of algorithms. Finally, Section 10 duly concludes this study, along with its limitations.

2. The canonical PSO algorithm

The PSO algorithm pursues optima by traversing a multi-dimensional search space with a swarm of particles. Experiences of each particle and its neighbors' contribute to the formation of the particle's search behavior. In the PSO computational method, a given problem is iteratively optimized by first generating at random a swarm of particles (candidate solutions). Particles coordinate and share their historically personal best positions with the neighborhood (a subset of the swarm or the entire

swarm) while questing throughout the problem's solution space. Thus, the temporary global best solution may be incorporated for possibly updating the solution obtained at each iteration. However, this greatly depends on the structure of the neighborhood topology. For instance, if the ring (*lbest*) topology is used, then information about the global best position is never exchanged or used.

2.1. Mathematical model

Similar to other meta-heuristic algorithms [34,35], a few major structural commonalities are followed in PSO, including swarm and solution initialization, updating strategy, and fitness evaluation. Only the last two steps (updating strategy and fitness evaluation) are repeated in the subsequent iterations until a termination criterion (e.g., reaching maximum function evaluations) is satisfied. Since PSO is a swarm-based meta-heuristic approach, a set of candidate solutions are maintained in the form of particles. During the exploration (diversification) and exploitation (intensification) phases in PSO, these particles or points interact with one another (for instance, the historically best areas or current best particle) and update their positions by adding some random search at each particle, pursuing the optimum location at last. To clarify the PSO search process, the mathematical expressions of PSO steps are detailed in the following.

2.1.1. Initialization process

The social attributes of a flock of birds were stimulated by the researchers to roll out the PSO optimization method in 1995 [1,2] after first simulating the data in the initialization step as follows:

- **Swarm initialization:** The particles are randomly scattered in the initial step of swarm production without a specific criterion (the reader is referred to Section 3.4.1 for different swarm initialization methods). Given subsequent iterations denoted by $t = 0, 1, \dots, T$, PSO begins with a randomly generated swarm of, say N, D-dimensional particles simplified with real-valued position vectors representing the initial candidate solutions by initializing each particle's position $\mathbf{x}_i^0$ (at iteration $t = 0$) to a random position in the search space as

$$\mathbf{x}_i^0 \sim U\left(\mathbf{x}_{\min}, \mathbf{x}_{\max}\right)^D, \tag{1}$$

  where D denotes the search space dimensionality or the size of the problem to be solved, and the individual $\mathbf{x}_i$ has an initial position vector $\mathbf{x}_i^0$ with values usually selected randomly from within a normally distributed range $U\left(\mathbf{x}_{\min}, \mathbf{x}_{\max}\right)^D$, with $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ being the search space bounds. In general, the following notation may be used to represent the i-th particle's position vector at each subsequent iteration $t \geq 0$:

$$\mathbf{x}_i^t = \left(x_{i,1}^t, x_{i,2}^t, \dots, x_{i,D}^t\right). \tag{2}$$

  Decision variables (i.e., elements of a position vector) of the problem are often related to physical measures or components that have natural bounds. Length, weight, and mass are examples of such bounds, where the values of the decision variables would be wanted not to be negative and a certain range (that is limited by $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$) of sufficient values should be defined to independently restrict the value of each decision variable within that range.
- **Velocity initialization:** All particles are frequently moving through the search space with a velocity (step size) that reflects particles' experiential knowledge as well as socially exchanged information about the promising areas visited in the search space, thereby driving the optimization process to the most feasible regions. Similarly to position initialization, each particle's velocity v_i can be initialized as

$$\mathbf{v}_i^t = \left(v_{i,1}^t, v_{i,2}^t, \dots, v_{i,D}^t\right), \tag{3}$$

  where $\mathbf{v}_i^t$ is the velocity vector of a particle i at iteration t. $v_{i,1}^t, v_{i,2}^t, \dots, v_{i,D}^t$ are velocity parameters, each of which is randomly initialized (at iteration $t = 0$) within the user-selected compact set (i.e., $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$) or rather commonly initialized with a random value between an upper bound ($v_{\max}$) and a lower bound ($-v_{\max}$), so as to prevent particles from shaving off the search space [36]. $v_{\max}$ is dubbed as the largest allowable step size in any dimension. It is optional, and, in fact, if the control parameters are correctly set, $v_{\max}$ is not necessary at all. For different velocity initialization methods, the reader is referred to Section 3.1.1.
- **Fitness initialization:** After initializing the swarm and velocity, every particle in the swarm is evaluated for its own initial personal fitness $f\left(\mathbf{p}_{best_i}^0\right)$, along with the initial global (or neighborhood) best fitness $f\left(\mathbf{g}_{best}^0\right)$ using Eqs. (5) and (6), respectively. In the initialized stage, the personal best solution $\mathbf{p}_{best_i}^0$ of a particle i is set to its initial position $\mathbf{x}_i^0$.

2.1.2. Fitness evaluation

The swarm particles are evaluated at the end of each iteration for their own fitness values as well as global optimum fitness. It is assumed that each particle i has a unique fitness value $f\left(\mathbf{x}_i^t\right)$ at each iteration t, which is calculated through an objective function evaluation. PSO memorizes the personal best solution (candidate global best solution) which every particle has ever met until the current iteration t as

$$\mathbf{P}_{best}^t = \left(\mathbf{p}_{best_1}^t, \mathbf{p}_{best_2}^t, \dots, \mathbf{p}_{best_N}^t\right), \tag{4}$$

where $\mathbf{P}_{best}^t$ represents the personal best positions of all particles in the swarm at iteration t. According to the type of optimization problem (minimization or maximization), the personal "best" position $\mathbf{p}_{best_i}^t$ that particle i has visited so far until the current iteration, say t, is calculated as:

$$\mathbf{p}_{best_i}^t = \mathbf{x}_i^l \mid f\left(\mathbf{x}_i^l\right) = \min_{k=0,1,\dots,t} / \max\left(\left\{f\left(\mathbf{x}_i^k\right)\right\}\right), \tag{5}$$

where l is the same index as k-th iteration in which the i-th particle has found the best position so far until the current iteration t. At the end of each iteration, all candidate solutions in $\mathbf{P}_{best}^t$ are sorted and the first ranked solution is selected as the global (or neighborhood) "best" solution or position $\mathbf{g}_{best}^t$ that i-th particle's neighboring particles have visited:

$$\mathbf{g}_{best}^t = \mathbf{p}_{best_m}^t \mid f\left(\mathbf{p}_{best_m}^t\right) = \min_{i=1,2,\dots,N} / \max\left(\left\{f\left(\mathbf{p}_{best_i}^t\right)\right\}\right), \tag{6}$$

where m is the same index as the i-th particle that has the overall best position among the individuals of the swarm.

2.1.3. Velocity and position updating

At each current iteration, say $t + 1$, the i-th particle's velocity $\mathbf{v_i}$ is first regulated by sending its parameters soaring in the positive or negative direction, contingent on the convergence of the current position, attracting the particle towards positions in the search space that are known to be good from past personal experience, as well as from the experience of other particles in the neighborhood of the particle. In fact, the original PSO [1,2] was implemented for two different neighborhood topologies, global best (*gbest*) PSO and local best (*lbest)* PSO.

The ***gbest*PSO** is the basic version of the algorithm introduced by Kennedy and Eberhart [1,2]. Pursuing the optimum solution, each particle moves – at each subsequent iteration $t > 0$ – towards its earlier best personal position $\mathbf{p}_{best}^t$ and the global best position $\mathbf{g}_{best}^t$ in the swarm by adding a velocity vector to the particle's position at the previous iteration, which can be simulated as

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}, \tag{7}$$

where $v_{i,j}^0 = 0$, and $v_{i,j}^{t+1}$ is the velocity vector of particle i – at iteration $t + 1$ – in dimension j, and is given by

$$v_{i,j}^{t+1} = \overbrace{\omega v_{i,j}^t}^{\text{Inertia component}} \overbrace{+ c_1 r_{1_{i,j}}^{t+1}\left(p_{best_i,j}^t - x_{i,j}^t\right)}^{\text{Cognitive component}} + \overbrace{c_2 r_{2_{i,j}}^{t+1}\left(g_{best,j}^t - x_{i,j}^t\right)}^{\text{Social component}}, \tag{8}$$

where $j \in \{1, 2, \ldots, D\}$ denotes the dimensions (or components) of particle i. New parameters, known as the cognition and social acceleration coefficients (or c_1 and c_2), along with the new IW (ω), which were newly added in the inertia PSO version introduced in [4], have shown superior performance over those earlier versions with fixed parameters. The two factors c_1 and c_2, also called "acceleration coefficients", are positive constants commonly used to determine when the cognition speed of i-th particle is accelerated towards $\mathbf{p}_{best_i}$ and $\mathbf{g}_{best}$, respectively. The symbol ω denotes the IW parameter which was originally developed to address the velocity explosion problem in the original 1995 PSO version [1,2]. Furthermore, IW can, in conjunction with c_1 and c_2, be used to balance global exploration and local exploitation. Note that, in the original PSO version [1,2], both c_1 and c_2 are equal to 2, thereby weighing the cognitive and social parts to 1, on average. Moreover, ω is equal to 1 in that version. Continuing to scrutinize Eq. (8), $r_{1_{i,j}}^{t+1}$ and $r_{2_{i,j}}^{t+1}$ are two independent random numbers at the current iteration $t+1$, which are uniformly distributed within the range [0,1], and sampled independently for every particle i at each dimension j, in order to keep the swarm diversity adequate. $p_{best_{i,j}}^{t}$ is the particle's personal best position or *pbest*, which is the best position encountered by particle i (at iteration t) in dimension j, and $g_{best,j}^{t}$ is the global best position or *gbest* is the best position encountered by any particle in the swarm (at iteration t) in dimension j, or equivalently, the fittest *pbest* over all particles in the swarm. Lastly, it should be memorized that at the end of each iteration, $\mathbf{p}_{best}$ and $\mathbf{g}_{best}$ are regularly evaluated using Eqs. (5) and (6), respectively, so that the global best solution $\mathbf{g}_{best}$ is obtained by the end of the algorithm.

The ***lbest*PSO** is similar to the *gbest* PSO. The difference is that, for the *lbest* PSO, the social component of Eq. (8) is defined as

$$c_2 r_{2_{i,j}}^{t+1}\left(n_{best,j}^{t} - x_{i,j}\right), \tag{9}$$

where $n_{best,j}^{t}$ is the neighborhood best position of particle i in dimension j which is the fittest *pbest*, not of the entire swarm as in *gbest* PSO, but of a subset of the swarm that forms the neighborhood of particle i. For the *lbest* PSO, the neighborhood of each particle i consists of two other particles: particle $i-1$, and particle $i+1$. The neighborhood topology of the *lbest* PSO is called a ring topology.

Note that the *gbest* PSO can be reformulated such that its social component is also given by Eq. (9). The neighborhood of each particle in the *gbest* PSO is then simply the entire swarm; such a neighborhood topology is called a fully-connected/all topology.

Notably, since its introduction in 1998, the parameter ω has been included in most of the newly introduced PSO variants. This is the reason behind referring to the algorithm with this improvement as the standard, canonical, or inertia PSO (ω-PSO). Algorithm 1 simulates the mechanism of the standard PSO (ω-PSO) from the late 1990's [37].

Algorithm 1: Standard PSO algorithm (ω-PSO)

```
Create and initialize an N D-dimensional swarm;
repeat
    foreach particle i = 1, 2, ..., N do
        if f(x_i) < f(p_best_i) then
            p_best_i = x_i;
        end
        if f(p_best_i) < f(g_best) then
            g_best = p_best_i;
        end
    end
    foreach particle i = 1, 2, ..., N do
        Update particle's velocity using Eq. (8);
        Update particle's position using Eq. (7);
    end
until maximum iteration is reached or stopping condition is true;
```

2.2. Iteration strategies

Personal and best positions of particles are updated at each iteration using such criterion referred to as iteration strategy. The PSO algorithm works mainly based upon one of two iteration strategies, namely, Synchronous Iteration Strategy (SIS) and Asynchronous Iteration Strategy (AIS). Algorithms 2 and 3 summarize the synchronous and asynchronous updates in PSO, respectively. In general, AIS is faster and less costly and therefore provides better results than SIS. However, it was shown recently that, compared to AIS, SIS yields better results, specifically with unimodal functions, and equal or better results with multimodal functions [38]. For most separable functions, SIS is not preferred. While in non-separable functions, SIS performs slightly well with *gbest* PSO, and AIS performs better with *lbest* PSO. However, there is no significant difference for most of the functions. Note that *gbest* PSO and *lbest* PSO are discussed within the context of Section 3.5.

Algorithm 2: Synchronous ω-PSO

```
Create and initialize the swarm;
repeat
    foreach particle i = 1, 2, ..., N do
        Evaluate f(p_best_i);
        Update p_best_i using Eq. (5);
        Update g_best using Eq. (6);
    end
    foreach particle i = 1, 2, ..., N do
        Update velocity using Eq. (8);
        Update position using Eq. (7);
    end
until maximum iteration is reached or stopping condition is true;
```

Algorithm 3: Asynchronous ω-PSO

```
Create and initialize the swarm;
repeat
    foreach particle i = 1, 2, ..., N do
        Update the velocity using Eq. (8);
        Update the position using Eq. (7);
        Evaluate f(p_best_i);
        Update p_best_i using Eq. (5);
        Update g_best using Eq. (6);
    end
until maximum iteration is reached or stopping condition is true;
```

3. Developments on the PSO paradigm

Rome was not built in a day. Several papers on the PSO algorithm, including the original and canonical versions, have been cited more than ten thousand times [1,2,37], since its very inception in 1995. Some of the great developments of the algorithm have been categorized in this section depending on the PSO aspects explored as follows.

3.1. Methods of velocity handling

3.1.1. Velocity initialization methods

There have been multiple opinions of velocity initialization [39]. On the basis of the flocking analogy theory, which states that physical objects do not have any momentum in their initial state, setting $v_{i,j}^{0} = 0$ for each particle i along all dimensions is one of the methods,

but this method was claimed to be critique and is to limit the exploration ability, which would substantially curtail the search space initial coverage. This is simply not the case because the initial particle positions are uniformly distributed, thus ensuring good initial coverage by the swarm of particles throughout the search space [39]. Another method is to have $\mathbf{v}_i^0 \sim U(\mathbf{x}_{\min}, \mathbf{x}_{\max})^D$, where it is more likely to obtain better solutions faster, depending on the fact that the exploration abilities of the swarm are significantly improved by initial random velocities; however, in this method, more particles may extend beyond the search space bounds in case of large initial step sizes (i.e., $\mathbf{v}_i^0 \sim U(\mathbf{x}_{\min}, \mathbf{x}_{\max})^D \rightarrow \mathbf{x}_i^1 \sim U(2\mathbf{x}_{\min}, 2\mathbf{x}_{\max})^D$), and many best positions may subsequently violate the boundary constraint [39]. As a response to that problem, initialization with small random values was proposed to add to the swarm diversity [39]. However, a question arises, which small random values are most appropriate? Surely, the nature and characteristics of the optimization problem is the most suitable answer to this problem. To conclude, it was observed that zero and small random initializations have similar behaviors. However, random initialization is slower in either improving the best solution or increasing the number of converged dimensions, due to the larger diversity and the emergence of particles with the best position roaming for longer [39]. Moreover, random initialization was noted to perform poorly for some benchmark functions and does not have a significant difference in final accuracy for most of the problems [39].

Gunasundari et al. [40] presented a Velocity-bounded Boolean PSO (VbBoPSO) based on Binary PSO (BPSO), wherein particles are initialized with random binary positions and binary velocities, and the velocity length is checked for the range of values to explore more regions and for better convergence, inspired by [41]. In VbBoPSO, $v_{\max}$ is the number of allowed ones in the new velocity vector v_i^{t+1} and length of the velocity vector is decreased using the negative selection method. To avoid the particles moving faster, each new velocity vector is tested to see if the number of 1s exceeds $v_{\max}$, then one bit with a value of 1 in the velocity vector is randomly selected and set to zero, continuing until the number of 1s becomes beyond $v_{\max}$. A variant of VbBoPSO, an Improved Velocity-bounded Boolean PSO (IVbBoPSO) to solve the feature selection problem, was proposed to solve the rapidly emerging stagnation in subsequent iterations. 28 benchmark functions were used to test the proposed algorithms in comparison with BPSO, BoPSO, BoPSO with velocity mutation (BoPSO-vm), and V-shaped transfer function based BPSO (VPSO).

3.1.2. Velocity upgradation methods

Bhambu et al. [42] developed a new approach named as Self-Balanced PSO (SBPSO), consisting of two phases: PSO search process and ABC inspired fitness-based search process. In PSO search process, a modified velocity update was used to update all solutions. In this process, the new positions were identified based on the previous global best solutions by replacing only the "cognitive component" in Eq. (8) by $c_1 r^{t+1}_{1_{i,j}} \left(g^t_{best,j} - x^t_{k,j}\right)\left(e^{\frac{-\alpha t}{T}}\right)$. The new "cognitive component", now called the global best random component, is calculated at j-th dimension by taking the difference between $g_{best,j}$ found so far and a randomly selected solution, k-th solution, thereby enforcing the k-th solution move towards $g_{best,j}$. The factor $e^{\frac{-\alpha t}{T}}$ is used to exponentially reduce further weight on this component, with α being a constant used to determine the weight of this factor. The impact of the global best solution $\mathbf{g}_{best}$ is reduced by this factor through iterations and hence the swarm diversification is improved while controlling the convergence speed. The same process of velocity update is repeated with ABC inspired fitness-based search process, where the exploitation abilities of the individuals are controlled and well-driven towards $\mathbf{g}_{best}$ and hence the convergence ability is improved.

Sen et al. [43] proposed an adaptive deterministic modified particle velocity-based PSO algorithm (MPV-PSO). In MPV-PSO, the stochasticity inherited from the conventional ω-PSO is eliminated by excluding the random numbers in the velocity equation. Moreover, tuning the weight factor and the cognitive and social acceleration coefficients is not needed any more thanks to introducing adaptive values for them based on the particle previous position. These adaptive values also help significantly in solving problems like getting trapped into local minima as well as oscillations about the global best position during steady-state operation.

Considering with the neighboring area of the global best position and the personal best position, it would extend the search space of their fixed position. It is helpful for the whole swarm to fly to a better position. This can be accomplished by introducing a Gaussian Disturbance to the global best position in PSO (GDPSO), as proposed in [44]. The one-dimension probability of the Gaussian distribution is given as

$$N(\mu, \sigma^2) = \frac{1}{\sqrt{\pi\sigma^2}} e^{-\left(\frac{x-\mu^2}{2\sigma^2}\right)},$$

where μ is the expectation of the distribution which is to determine its location, σ is the standard deviation which determines the scale of distribution. The proposed Gaussian distribution makes 68% of the points scattered in the range $[\mu - \sigma, \mu + \sigma]$. The reason for using such a disturbance operator is to increase the probability of escaping from a local optimum. In GDPSO, the Gaussian disturbance operator acts on the global best position of velocity updating Eq. (8) by multiplying $g^t_{best,j}$ by $\left(1 + N(\mu, \sigma^2)\right)$. In comparison with different standard deviation σ values of Gaussian disturbance on some unimodal and multimodal functions, it is recommended that σ is set to 0.3. Compared with the standard ω-PSO, the results show that GDPSO has faster convergence on some unimodal functions and better exploration ability on multimodal functions. In [45], an algorithm, called AdPSO, was proposed to enable the velocity of swarm particles to be reinitialized as soon as stagnation is sensed. AdPSO improves the exploration of the swarm by re-energizing particles that have very low velocities, by incorporating a new threshold parameter. It is the onset of stagnation when the velocity is below the threshold, and consequently the velocity is reinitialized. During the PSO procedure, reinitialization is allowed only at specific points (every a epochs) and only after executing E initial epochs, to ensure the swarm stability. Thus, a particle is examined for stagnation only at epochs $E + a$, $E + 2a$, $E + 3a$, and so on, to see whether its velocity needs to be reinitialized. The initial velocity vector may have a high component, which drives the particles very rapidly to a specific region, and due to a potential strong velocity component, the particle may be unable to move from this point. Therefore, the velocity vector has also been bounded to allow particle transit only through a number of steps along any given dimension, discouraging the very large movements of particles in the pattern space [45].

3.1.3. Velocity clamping methods

Velocity clamping was introduced by Eberhart and Kennedy [2] to tackle the problem of velocity explosion by keeping the particles within the bounds of the search space and forcing them to take a good step size to comb the entire search domain. With the absence of velocity clamping, the optimization procedure will become prone to explode and the particles will change their positions very rapidly [46]. With a large value of $v_{\max}$, the particles may move unpredictably and the chances are present that they may even jump over the optimal solution. Conversely, if the value of $v_{\max}$ is considerably small, the movement of particles may be restricted and the swarm may not be efficient to examine the promising regions of the lookup space. Moreover, haphazard selection of velocity values may lead to divergent behavior due to excessively large particle movements. Hence, initializing $v_{\max}$ is important and is typically chosen from 0.1 to 1.0 times the lookup range of the potential solution space (e.g., $v_{\max} = \frac{\mathbf{x}_{\max} - \mathbf{x}_{\min}}{2}$ [16]). To restrict the movement of the swarm particles within the lookup area, each velocity component at j-th dimension is being restricted to the range $[-v_{\max}, v_{\max}]$ indepen-

dently for each particle i at current iteration t as follows:

$$v_{i,j}^{t} = \begin{cases} v_{\max} & \text{if } v_{i,j}^{t} > v_{\max}, \\ -v_{\max} & \text{if } v_{i,j}^{t} < -v_{\max}. \end{cases}$$

The significance of velocity clamping in PSO has been reevaluated in recently introduced studies, providing evidence that the popular setting of $v_{\max} = 0.2$ [4] of the search space range may not be the best choice for many problems and that the velocity clamping should be omitted in some cases. In [47], the maximal velocity was set in the range 0.1-1.0 times the search space range with a 0.1 step. In addition, a variant with no velocity clamping was tested. Based on statistical tests using the CEC'13 benchmark set, in low dimensionality ($D = 10$), the settings of $v_{\max} = 0.3$ and 0.6 performed the best while not using velocity clamping at all or $v_{\max} = 0.1$ performed not well and so did the algorithm. However, in higher dimensionality ($D = 30$), the performances of PSO with $v_{\max} > 0.1$ and PSO without $v_{\max}$ are comparable. Moreover, as depicted in [47], the overall shape of the convergence history implies that the $v_{\max}$ setting does not directly affect the diversity of the swarm.

Schutte et al. [48] suggested a Linearly Decreasing $\mathbf{v}_{\max}$ method (LDVM). First, at each iteration, the vector $\mathbf{v}_{\max}$ is recalculated per dimension as a random fraction ($\gamma \in [0, 1]$) of the search domain. In LDVM, after a predefined number of iterations h if there is no improvement in the swarm global best and personal best fitness values, the $\mathbf{v}_{\max}$ and ω are kept decreasing linearly while the number of iterations is increasing, as

$$\text{if } f\left(\mathbf{g}_{best}^{t}\right) \geq f\left(\mathbf{g}_{best}^{t-h}\right), \text{ then } \mathbf{v}_{\max}^{t} = \alpha \mathbf{v}_{\max}^{t} \text{ and } \omega^{t} = \beta \omega^{t-1},$$

where $\alpha, \beta \in (0, 1)$ are prescribed by the user. Later in, motivated by the LDVM, Sakamoto et al. [49] implemented a new replacement method, Rational Decrement $\mathbf{v}_{\max}$ method (RDVM), for mesh routers in Wireless Mesh Networks (WMN). In RDVM, the $\mathbf{v}_{\max}$ vector is kept decreasing as the algorithm proceeds as

$$\mathbf{v}_{\max}^{t} = \sqrt{W^2 + H^2} \times \frac{T - t}{t},$$

where W and H are respectively the width and the height of the considered area to distribute the mesh routers.

Irrespective of the above-mentioned velocity clamping strategies, it has been recently revealed that there is no clear rule of thumb as to whether clamping is generally beneficial and that the optimal clamping strategy depends strongly on the values of the the IW and acceleration coefficients [14]. However, it has also been demonstrated that unconstrained optimization with velocity clamping applied may perform better than other configurations [14].

3.2. Position update mechanisms

To cope with the potential loss of swarm diversity, a new position update rule based on the normal distribution was proposed by Kiran [50]. Fully discarding velocity update rule in Eq. (8), a new position update formula was defined as $x_{i,j}^{t+1} = \mu + \sigma \times Z$, in order to obtain a new position for a particle through iterations, where μ (mean), σ (standard deviation), and Z are calculated as follows:

$$\mu = \left(x_{i,j}^{t} + p_{best_i,j}^{t} + g_{best,j}^{t}\right)/3,$$

$$\sigma = \sqrt{\frac{1}{3} \times \left[\left(x_{i,j}^{t} - \mu\right)^2 + \left(p_{best_i,j}^{t} - \mu\right)^2 + \left(g_{best,j}^{t} - \mu\right)^2\right]},$$

$$Z = (-2 \ln k_1)^{1/2} \times \cos(2\pi k_2),$$

where $k_1, k_2 \sim U(0, 1)$ are defined independently. On solving three constrained engineering optimization problems and 13 nonlinear global optimization benchmark functions, the proposed approach in [50] outperformed ω-PSO and its variants in terms of solution quality and robustness.

Constrained Optimization Problems (COPs) (discussed in Section 4.2) cannot be tackled using the classical ω-PSO which does not comprise constraint handling techniques. Furthermore, certain types of optimization problems can be efficiently solved by most existing PSO variants, with potential premature convergence caused by the directional information used to guide the search process and the limited search operators. To overcome the aforementioned drawbacks, an improved PSO variant, called Constrained Multi-swarm PSO Without Velocity (CMPSOWV) was proposed in [51] based on a drastic modification of the ω-PSO through discarding the velocity component of particles and using instead a linear combination of the personal best position of the particle and the global best position to make a position update. Particularly, CMPSOWV first incorporates a constraint handling technique before optimizing the objective function to attract the swarm towards the feasible regions of the search space. Aiming at robusting the algorithm to solve different types of COPs, multiple search operators were offered for each CMPSOWV particle through introducing two evolution phases, current swarm evolution and memory swarm evolution. Let $\mathbf{P} = [\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N]$ be the swarm of CMPSOWV with a swarm size of N particles randomly initialized. The swarm of CMPSOWV is divided using the multi-swarm approach into K sub-swarms. Each k-th sub-swarm size is denoted as N^k, where $k = 1, 2, \dots, K$ and $N^k = \lfloor N/K \rfloor$. Each of the k-th sub-swarm represented as $\mathbf{P}^{sub,k} = [\mathbf{x}_1^k, \dots, \mathbf{x}_i^k, \dots, \mathbf{x}_{N^k}^k]$ is produced using a randomly generated reference point denoted as U^k. The i-th particle with p_{best_i} whose Euclidean distance is the nearest to U^k is first identified and the remaining N^{k-1} particles with $\mathbf{p}_{best}$ whose Euclidean distances are the nearest to p_{best_i} are then combined to form $\mathbf{P}^{sub,k}$. All $\mathbf{P}^{sub,k}$ members are then discarded from the main population/swarm $\mathbf{P}$. Similar procedures are repeated until K sub-swarms are produced from the main swarm $\mathbf{P}$. After obtaining all sub-swarms, the position $\mathbf{x}_i^k$ of each particle i in every sub-swarm k is updated in the first phase, named as current swarm evolution, as

$$\mathbf{x}_i^k = \begin{cases} c_1\left(\mathbf{r}_{1_i}^k \otimes \mathbf{p}_{best_i}^k\right) + c_2\left(\mathbf{r}_{2_i}^k \otimes \mathbf{g}_{best}^k\right) + c_3\left(\mathbf{r}_{3_i}^k \otimes \left(\mathbf{p}_{best_a}^k - \mathbf{p}_{best_b}^k\right)\right) & \text{if } k = 1, \\ c_1\left(\mathbf{r}_{1_i}^k \otimes \mathbf{p}_{best_i}^k\right) + c_2\left(\mathbf{r}_{2_i}^k \otimes \mathbf{g}_{best}^k\right) + c_3\left(\mathbf{r}_{3_i}^k \otimes \left(\mathbf{g}_{best}^{k-1} - \mathbf{p}_{best_c}^k\right)\right) & \text{otherwise}, \end{cases}$$

where $\mathbf{p}_{best_i}^k$ and $\mathbf{g}_{best}^k$ are respectively the i-th particle personal best position and the global best particle in the current sub-swarm k. $\mathbf{p}_{best_a}^k$, $\mathbf{p}_{best_b}^k$, and $\mathbf{p}_{best_c}^k$ are the personal best positions of three randomly selected particles, where $a, b, c \in [1, N^k]$, $a \neq b \neq i$, and $c \neq i$. c_1, c_2, and c_3 are the acceleration coefficients, and $\mathbf{r}_{1_i}^k$, $\mathbf{r}_{2_i}^k$, and $\mathbf{r}_{3_i}^k$ are random vectors sampled – at each iteration – from a uniform distribution within [0,1], for each particle i in the current sub-swarm k. The third component (c_3 part), newly added in that roposal, is introduced to facilitate the exchange of information between all particles in the k-th sub-swarm and the global best particle in the $(k-1)$-th sub-swarm, in order to prevent the the k-th sub-swarm from premature convergence. In the CMPSOWV second phase known as memory swarm evolution, all sub-swarms are recombined to form the main swarm $\mathbf{P}$. The new personal best position $\mathbf{p}_{best_i}^{new}$ of each particle i is calculated as

$$\mathbf{p}_{best_i}^{new} = \begin{cases} \mathbf{p}_{best_f} + \left(\mathbf{r}_{a_i} \otimes \left(\mathbf{p}_{best_g} - \mathbf{p}_{best_h}\right)\right) & \text{if } \psi > 0.5 \text{ and } f\left(\mathbf{p}_{best_e}\right) > f\left(\mathbf{p}_{best_i}\right), \\ \mathbf{p}_{best_i} & \text{if } \psi \leq 0.5 \text{ and } f\left(\mathbf{p}_{best_e}\right) > f\left(\mathbf{p}_{best_i}\right), \\ \mathbf{p}_{best_i} + \left(\mathbf{r}_{b_i} \otimes \left(\mathbf{p}_{best_e} - \mathbf{p}_{best_i}\right)\right) & \\ + \left(\mathbf{r}_{c_i} \otimes \left(\mathbf{p}_{best_l} - \mathbf{p}_{best_m}\right)\right) & \text{if } f\left(\mathbf{p}_{best_e}\right) < f\left(\mathbf{p}_{best_i}\right). \end{cases}$$

Here, $\mathbf{p}_{best_e}$, $\mathbf{p}_{best_f}$, $\mathbf{p}_{best_g}$, $\mathbf{p}_{best_h}$, $\mathbf{p}_{best_l}$, and $\mathbf{p}_{best_m}$ are the personal best positions of six particles randomly selected from a swarm so as to maintain the swarm diversity, where $e, f, g, h, l, m \in [1, N]$, $e \neq i$, $f \neq g \neq h \neq i$, and $l \neq m \neq i$. ψ is a random number and $\mathbf{r}_{a_i}$, $\mathbf{r}_{b_i}$, and $\mathbf{r}_{c_i}$ are random vectors generated separately per dimension of every particle, at each fitness evaluation step, such that $\psi \in [-1, 1]$ and $\mathbf{r}_{a_i}, \mathbf{r}_{b_i}, \mathbf{r}_{c_i} \in [0, 1]$. As the last phase of CMPSOWV, a probabilistic mutation operator with a mutation probability $P^{\text{MUT}} = 1/D$, is involved to provide an additional momentum for the global best particle to get away from the local optima. At the end of each phase, Deb's rule [52], an intuitive approach

to compare and determine the best feasibility between two solutions without additional parameters (e.g., penalty factor), is applied at every iteration to determine if the $\mathbf{p}_{best_i}^{new}$ of the i-th particle can be used to replace both its earlier $\mathbf{p}_{best_i}$ and the global best particle $\mathbf{g}_{best}$. CMPSOWV overall optimization performances were compared with selected constrained optimization algorithms for solving the CEC'06 and CEC'17 benchmark functions as well as realistic engineering design problems. Among the compared methods, the proposed CMPSOWV has revealed the best search accuracy in solving the majority of problems, based on extensive simulation results.

3.3. Control parameter adaption

In the PSO first application [4,53], the particle velocity was not controlled and was having a tendency to explore large regions of the search space, thereby improving the PSO exploratory performance, rather than the exploitative performance. Therefore, it is necessary to control the behavior of particles in the swarm by developing new variants of the method. To this end, the performance of PSO has been empirically tested, which approved the distinct sensitivity of the algorithm to the values assigned to its control parameters [54]. So, where are these control parameters used? Simply, they are located within the velocity updating formula in Eq. (8) as follows:

- **Previous velocity** $\omega v_{i,j}^t$, also called "inertia component" or flight direction, provides the particle with the appropriate momentum to rove across the search space without changing its direction drastically, based on the hidtory of previous flight directions. This part contains the IW (or ω) control parameter.
- **Cognitive component** $c_1 r_{1_{i,j}}^{t+1}\left(p_{best_i,j}^t - x_{i,j}^t\right)$, denotes the memory of the previous personal best position, having a role to quantify performance relative to past performances – *nostalgia*. This part contains the first acceleration coefficient c_1 control parameter.
- **Social component** $c_2 r_{2_{i,j}}^{t+1}\left(g_{best,j}^t - x_{i,j}^t\right)$, quantifies the current performance related to the global best solutions found so far – *envy*. It involves the second acceleration coefficient c_2 control parameter.

The following subsections reflect PSO performance with respect to some of its recommended parametrizations, with this affecting the convergence performance.

3.3.1. Inertia weight

For the remainder of this section, $\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum allowable IWs, respectively. According to the theoretical and empirical analysis of PSO in [3],

$$-0.16199 < \omega < 0.78540 \tag{10}$$

is necessary to imply the convergent behavior of the algorithms and this rule, from now on, will be used as a basis for the convergence analysis in this subsection and the successive one. The convergence speed of the PSO algorithm is improved by using large ω value – favor exploration, while the convergence precision is improved by using low ω value – favor exploitation. In other words, when $\omega \geq 1$, the swarm's knowledge is suppressed, giving higher attention to the particles' self-knowledge, as well as the swarm may diverge because velocities will highly increase over time and the particles will thus fail to go through more promising regions. In turn, when $0 < \omega < 1$ (smaller ω value), the particles decelerate, and the algorithm acts like a "jumping out" function and is prevented from convergence to a local optimum. However, the algorithm's properties will get worse eventually when there are too many jumps, which makes it perform like a stochastic search. Note that the inertia could not independently balance exploration and exploitation in PSO but jointly with other control parameters, such as c_1 and c_2 [4].

In order to get a satisfactory IW while the algorithm is executed, different strategies were proposed to dynamically change the IW and can be generally divided into two main categories: linear and nonlinear. In the linear strategy, IW can decrease linearly as the number of iterations increases, that ensures early convergence acceleration through initial larger ω values and less chance, however, of falling into the local optimum due to the smaller ω values that originate towards the end of the iterations. Compared to the linear strategy, the nonlinear strategy can quickly spread all particles across the search space by introducing bigger ω values in the initial stage to reduce the slow speed to efficiently determine the approximate range of global optimal values. Moreover, introducing smaller ω values in the initial stage decreases the quick speed and the global optima can be determined by the particles with almost constant speed. Clearly, better performance can be obtained by nonlinear strategy than linear strategy. However, in the nonlinear strategy, there is no any guideline for the ω value, as it is just gradually tentatively decreasing.

The IW model of PSO of Shi and Eberhart [37] adopted the range [0.9,1.2] to assign a positive constant to ω throughout the entirety of the search. However, a function of time t (t being the iteration number), or even a random number [55] can also be used to define ω. On the other hand, a strong relationship between ω, c_1, and c_2 was stated by Van den Bergh and Engelbrecht [56], which can be modelled as:

$$\omega > \frac{1}{2}(c_1 + c_2) - 1.$$

Assuming an equilibrium state (i.e., $c_1 = c_2 = 1$), $\omega > 0$ will be the result, which, according to the convergence rule in Eq. (10), it should exhibit convergent behavior at some time later when ω gradually approaches 0.78540. In the same context, Li et al. [57] used a constant value of 0.6 for the IW based on the ability demonstrated by that value to lead to convergent trajectories [3]. Note that this employed value of ω should trivially satisfy the convergence rule in Eq. (10) and will thus lead to a convergent behavior.

Unfortunately, due to insufficient prior knowledge of the search process, adjusting the IW dynamically through developing a mathematical model is difficult or impossible [58]. Therefore, typically, the exploration and exploitation of the search space is controlled by adjusting the acceleration coeffeceints c_1 and c_2, besides changing ω from $\omega_{\max} = 0.9$ to $\omega_{\min} = 0.4$ using a linearly decreasing function of time [59], such as

$$\omega^t = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{T} \times t.$$

However, this Linearly Decreasing IW technique (LDIW) has one shortcoming: The PSO can stuck into local optima with ease, especially when there are multiple optimum solutions to the optimization (objective) function [7]. To curb this shortcoming, a new strategy was proposed by Farooq et al. [60], wherein during the first half of the iterations, the IW is linearly lessened from 0.9 to 0.4, and the same strategy is reinitialized for the remaining half of the iterations. Thus, the value of IW is high at two points while executing the algorithm (i.e., when starting the algorithm and when reaching half of the iterations). Due to the elevated value of IW at the beginning of the two halves, the particles will work to quest the lookup area. Similarly, at two end points while executing the algorithm, the IW value is approaching its minimum, first when the midpoint of iterations is approached and the second when the maximum iterations are done. Due to these minimum values, the particles will strive to locally exploit the search space to obtain a better solution. Therefore, this approach will exhibit convergent behavior. Chatterjee and Siarry [61] suggested that a nonlinearly decreasing function of time can be formulated to change the IW as

$$\omega^t = \left(\frac{T - t}{T}\right)^n (\omega_{\min} - \omega_{\max}) + \omega_{\max},$$

where the user/researcher chooses the nonlinear modulation index n. Setting $n \in [0.9, 1.3]$ is usually satisfactory, according to the authors. Assuming $n = 1$, according to the convergence rule in Eq. (10), the algorithm will exhibit convergent behavior when $\frac{t}{T} > 0.81353$ (i.e., after 81% of the iterations are run).

PSO with Selective Multiple IWs (SMIWPSO) was implemented by Gupta et al. [62] using the four best-performing linear IW techniques,

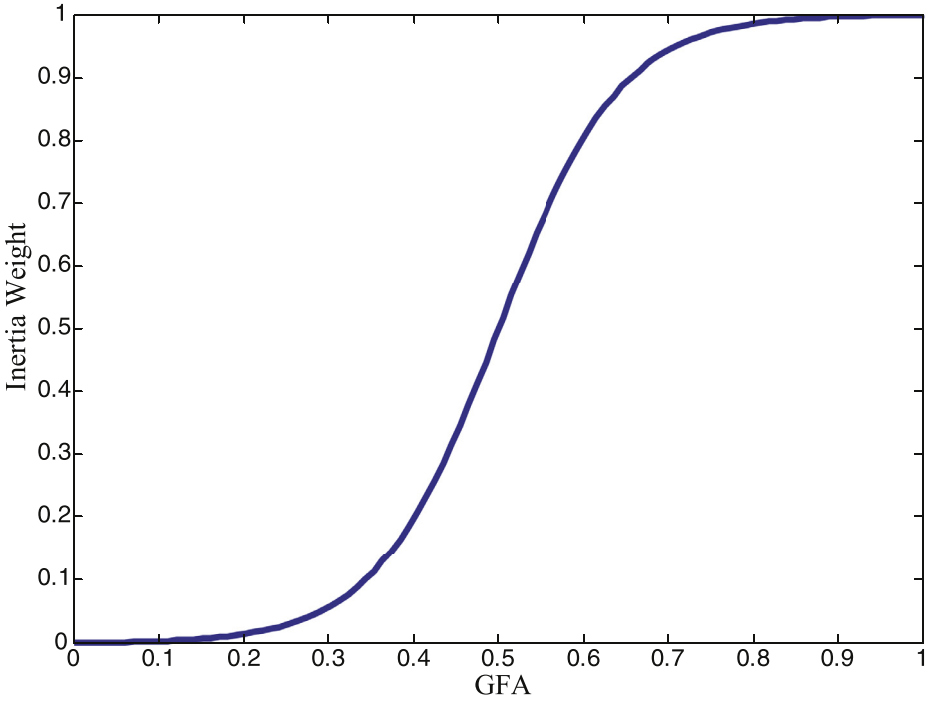


Fig. 1. Nonlinear IW based on APS [63].

including constant, random, chaotic, and linearly decreasing. In SMIWPSO, the four techniques have the same chance to be selected as the current IW to update the velocity of particles, subject to a controlling parameter $P \in [0, 1]$. As the complete results of the experiments have shown over three different criteria, the SMIWPSO performed better than ω-PSO in terms of the Average Mean Error (AME), Average Function Evaluation (AFE), and Success Rate (SR). The Dynamic Nonlinearly Changed IW of PSO (APS-DNIW) based on the Average Particle Spacing (APS), proposed by Liang and Kang [63], adapts the IW based on the degree of dispersion of individual particles to each other in the swarm at the current iteration t which can be defined as

$$S^t = \frac{1}{NH}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{D}\left(x_{i,j}^t - \overline{x_{i,j}^t}\right)^2},$$

where H denotes the diagonal maximum length of the search space, and N and D denote the swarm size and the search space dimension, respectively. $x_{i,j}^t$ and $\overline{x_{i,j}^t}$ respectively denote the value and the coordinate average value of i-th particle. The S^t expresses the average particle spacing in a swarm, such that the smaller the average spacing, the worse species diversity and the more concentrated swarm. To adjust the nonlinear IW as swarm diversity changes, a new nonlinear dynamic strategy based on the APS is employed as

$$\omega^t = 1/\left(1 + e^{-10(S^t - 0.5)}\right).$$

Group fitness variance (GFA [64]) was used in [63] to evaluate the premature convergence phenomena only from the aspects of the function value to estimate the overall distribution of particles. As shown in Fig. 1, under the larger APS, the dynamic nonlinear IW remains near its maximum ($\omega = 1$). Then, as the APS decreases from big to small, the linear transition of IW is approximated. Finally, when APS has small values, the IW is maintained near the minimum. Through these nonlinear changes, the well-balance between global and local search abilities is very well managed, which can help, with fast convergence, obtain the optimal solution, and the APS-DNIW would thus exhibit convergent behavior.

The Self-Regulating IW with PSO algorithm (SRPSO) by Tanweer et al. [65] was managed to regulate the IW such that the inertia is increased for the best particle while decreased for others. The justification was that a high level of confidence should be granted to the best particle in its current direction by accelerating it quicker, while a linearly decreasing IW strategy should be adopted for the remainder of particles. The self-regulating IW is employed according to

$$\omega_i^t = \begin{cases} \omega_i^t + \eta\Delta\omega & \text{for the best particle,} \\ \omega_i^t - \Delta\omega & \text{for other particles,} \end{cases}$$

with

$$\Delta\omega = \frac{\omega_{\text{start}} - \omega_{\text{end}}}{T},$$

where $\eta = 1$ is a constant for controlling the rate of acceleration. $\omega_{\text{start}} = 1.05$ and $\omega_{\text{end}} = 0.5$ are the start and end values of the IW, respectively. Indeed, such parameters do not lead to convergent behavior. Therefore, using the parameters $\omega_{\text{start}} = 0.9$ and $\omega_{\text{end}} = 0.4$ while sustaining the η parameter unchanged would simplify $\Delta\omega = 0.0001$ and should consequently exhibit convergent behavior. Equally, each particle in the swarm is likely to be the best particle, and thereby the IW is increased accordingly. Thus, given a swarm of N particles, the IW of each particle will decrease by $0.0001(N-2)$ every N iterations, or, on average, $\frac{0.0001*(N-2)}{N}$ for each iteration t. Assuming $N = 30$ leads to the necessary condition of $1227.86 < t < 11378.50$ for convergent behavior. Thus, the SRPSO algorithm should start to exhibit convergence at iteration 1228 (i.e., after 24.6% of the search has completed). Note that, when the number of iterations is stated longer than 11378 and the SRPSO algorithm does not reach complete stagnation (the stagnation problem is described in Section 5.6), the divergent behavior will be exhibited once again as the IW will be then less than −0.16199. The Exponential nonlinear IW PSO algorithm (EWPSO) by Borowska incorporates an exponential function of the minimal and the maximal fitness of the particles to select the IW based on the best and the worst particles at each iteration according to

$$eiw^t = \frac{\left(f_{\max}^t - f_{\min}^t\right)e^{-fr^t}}{1000\,fb^t \cdot f_{\max}^t},$$

$$w^{t+1} = w^t - eiw^t,$$

where $f_{\max}^t$ and $f_{\min}^t$ are the values of maximal and minimal fitness in the previous iteration t, respectively. The factors $fr \in [0, 1]$ and $fb \sim U(0, 1)$

determines how much the IW is influenced by the maximal fitness. No guidance was provided for the selection of the fr parameter value, and thus, the mid-point of the allowable range ($fr = 0.5$) is taken. Hence, applying the convergence rule in Eq. (10), this strategy effectively samples the IW at each iteration according to

$$\omega^t \sim U\left(0.002\left(\frac{f^t_{\max} - f^t_{\min}}{f^t_{\max}}\right), 0.0074\left(\frac{f^t_{\max} - f^t_{\min}}{f^t_{\max}}\right)\right).$$

Indeed, the literature is replete with other methods for dynamically changing the IW over iterations (e.g., nonlinear, natural logarithm IW [66], stability-based adaptive IW strategy [67], fuzzy adaptive IW [68], and many more).

3.3.2. Constriction factor

The Constriction Factor PSO (CFPSO) was suggested by Clerc in 1999 [53] to help in solving optimization problems faster by making an exploration-exploitation trade-off to ensure the convergence of the algorithm, affecting the particles' trajectories around candidate solutions [53]. This constriction factor is given by:

$$\chi = \frac{2}{\left|2 - C - \sqrt{C^2 - 4C}\right|},$$

where $C = c_1 + c_2$ and $C > 4$. Thus, Eq. (8) can be rewritten as

$$v^{t+1}_{i,j} = \chi\left[v^t_{i,j} + c_1 r^{t+1}_{1_{i,j}}\left(p^t_{best_i,j} - x^t_{i,j}\right) + c_2 r^{t+1}_{2_{i,j}}\left(g^t_{best,j} - x^t_{i,j}\right)\right].$$

When using the constriction factor with PSO (typically $C = 4.1$, and thus $\chi \approx 0.7298$), a convergent behavior is exhibited as the velocity tends to 0.

Eberhart et al. [59] compared the constriction factor with the IW, and the authors concluded that the constriction factor method could provide better quality solutions, although there is a mathematical equivalence between the two methods, ensuring that when $\omega = \chi$, the CFPSO and ω-PSO are equivalent. The authors also combined the constriction factor with $v_{\max}$ which is defined by the dynamic range of $\mathbf{x}_{\max}$ in each dimension. However, the parameters values were set on the basis of reasonable inference, not rigorous theoretical analysis.

Recently, the effects of parameters on the constriction factor were evaluated for finding more proper parameter values. To this end, Dias et al. [69] analyzed the acceleration coefficients c_1 and c_2, using a set of nine values, in the hope of finding their best combination of values that produces a better PSO convergence behavior without resorting to velocity clamping, using the new constriction factor which is given by

$$\chi = \begin{cases} \frac{2k}{C - 2 - \sqrt{C^2 - 4C}} & \text{if } C > 4, \\ k & \text{otherwise}, \end{cases}$$

where $C = c_1 + c_2$ and $k \in [0, 1]$ are set as such, so as to ensure convergent behavior. According to the analysis results, $c_1 = c_2 = 2.05$ were chosen, thereby providing a balance between exploration and exploitation of the search space and further confirming the value $C = 4.1$ which was recommended by Clerk [53]. Thus, this constriction factor method will exhibit convergent behavior. In turn, Maharana and Dash [70] proposed PSO with a Ramp Rate Constriction Factor (RRCPSO) which is given by

$$\chi = \frac{4}{\left|4 - \psi - \sqrt{\psi^3 - \psi^2 - 3\psi - 2.2}\right|},$$

where $\psi \in [2.1, 3.1]$. As ψ increases, χ decreases, giving a higher chance of slower convergence because of the diminished upgradation in the velocity of particles. Thus, RRCPSO may exhibit convergent behavior.

3.3.3. Cognitive and social acceleration coefficients

Controllable stochastic that influences the velocity of the swarm is evolved when, respectively, multiplying the cognitive and social acceleration coefficients c_1 and c_2 by random vectors $\mathbf{r_1}$ and $\mathbf{r_2}$. c_1 and c_2, simply put, weigh how much a particle should move towards its personal cognitive attractor ($\mathbf{p}_{best}$) and the social attractor ($\mathbf{g}_{best}$) found so far in the neighborhood, respectively. Particles are innately cooperative based on the information exchanged among them, implying that unbiased acceleration coefficients may fully equalize the contributions of the cognitive and social components of acceleration to the convergence behavior, thereby leading to a potential falling into the local optima trap. It is obvious that creating a trade-off between social and cognitive components based on c_1 and c_2 may be useful, especially in the case of multimodal problems with multiple promising regions. Moreover, these two components can be balanced manually during the search process, but more parameters to tune will be required yet it will be problem specific.

The acceleration coefficients c_1 and c_2 pair may have many case specific implementations outlined in Table 2, along with their respective implications, each emerging potential PSO variants like social-only PSO [71] and cognitive-only PSO [71]. In general, the values of c_1 and c_2 are commonly kept positive and constant. An empirically obtained optimum pair seems to be 2.05 [72] for both c_1 and c_2, and incorrect initialization or significant departures may lead to divergent behavior. Clerc's fuzzy acceleration [73] incorporated swarm diversity within the subsequent iterations to adaptively refine coefficient values, thus reporting improvements in the convergence performance. Shirazi et al. [74] tested four different settings of the accelerators c_1 and c_2 with dynamically changing, such that c_1 increasing and c_2 decreasing, $c_1 = c_2$ increasing, $c_1 = c_2$ decreasing, and vice versa. Based on the optimized results, c_1 decreasing and c_2 increasing was chosen, so as to optimize 11 benchmark functions. In six out of the eleven benchmark functions, the acceleration coefficients are dynamically changing such that c_1 is decreasing to a linearly decreasing function from 2 to 0.5, and c_2 is increasing to a linearly increasing function from 0.5 to 2.

To make the idea more concrete, Wu et al. [75] assumed that the process of particle evolution should reflect a gradual reduction in individual cognition while gradually increasing the global cognition. On the other hand, for the linear reduction of c_1 in PSO using time-varying acceleration coefficients [76], c_1 was redefined as

$$c^t_{1,j} = x_{\min,j} \sin\left(\frac{\pi}{2}\tau_1 \frac{\left(\frac{t}{2} - T\right)}{\frac{t}{2}}\right) + x_{\max,j},$$

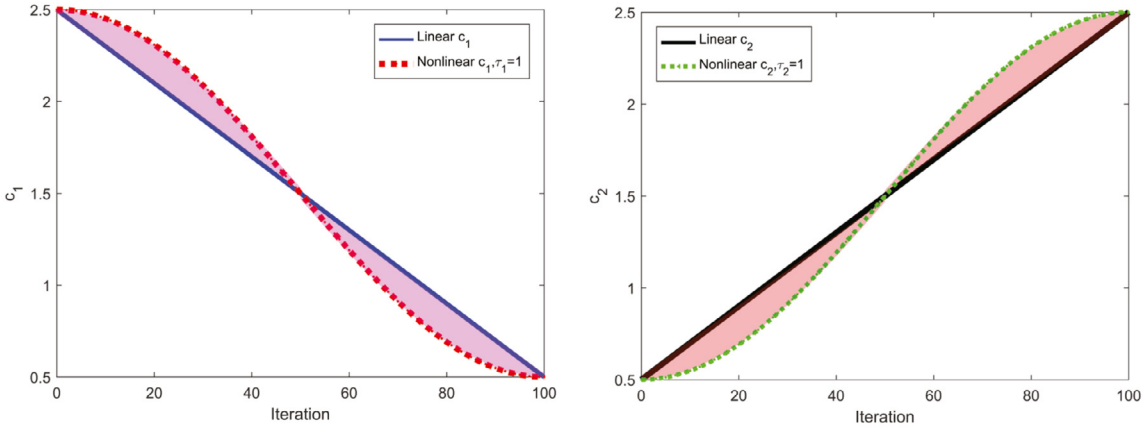
where $x_{\min,j} = 1$ and $x_{\max,j} = 1.5$ are prescribed. When $\tau_1 = 1$, the degree of nonlinearity of c_1 is shown in Fig. 2(a) with the red curve, whereas the emphasis on particle velocity update in various stages is quantified by the red coverage area. The degree of emphasis varies depending on the different value of τ_1; however, the range 0.75 and 1.15 has been proven reasonable for the value τ_1. During the early evolution, the improved nonlinear c_1 decreases slowly, compared with the linear reduction. In the late evolution process, the particles' cognitive ability should be transformed from individual to global to avert falling into a local optimum. Using the global optimal experience, the particles scurry to the global feasible regions. The blue line in Fig. 2(a) represents the trend of c_1, while the improving trend of c_1 is represented by the red curve. The cognitive acceleration is controlled by the c_2 curve, where c_2 would increase in the process of evolution. The nonlinear green curve in Fig. 2(b) is produced by

$$c^t_{2,j} = x_{\min,j} \sin\left(\frac{\pi}{2}\tau_2 \frac{\left(T - \frac{t}{2}\right)}{\frac{t}{2}}\right) + x_{\max,j}.$$

In Fig. 2(b), when $\tau_2 = 1$, the degree of nonlinearity is represented by the pink coverage area. From the above analysis, it is preferable for the particles to comb their personally found optimal regions in the early evolution process. Due to the improved c_2, the environment can be explored more fully such that the particles can gain sufficient experience by deferring the increasing in c_2, aiming to exploit the early individual

Table 2
Potential c_1 and c_2 combinations, along with their implications.

Possibilities of c_1 and c_2	Implication(s)
$c_1 = c_2 = 0$	- Position update will only depend on the inertia component. - Susceptibility to stagnating at a non-optimal position. - The entire swarm is attracted to this globally best position and is susceptible to stagnation at this location.
$c_1 > 0$ and $c_2 = 0$ (cognitive-only PSO)	- Each particle conducts its own search locally. - Particles are independent hill-climbers (i.e., particles rely only on their own experience). - Convergent behavior is not exhibited.
$c_1 = 0$ and $c_2 > 0$ (social-only PSO)	- Swarm is simply one stochastic hill-climber (i.e., particles will rely solely on the knowledge of the best particle in the whole swarm). - More exploitative behavior and fast convergence are exhibited.
$c_1 = c_2 > 0$	- Particles are attracted towards the average of $\mathbf{p}_{best}$ and $\mathbf{g}_{best}$.
$c_1 > c_2$	- Promote exploration (cognitive acceleration).
$c_2 > c_1$	- Promote exploitation (social acceleration).



(a) Linearly decreasing acceleration coefficient c_1.

(b) Linearly increasing acceleration coefficient c_2.

Fig. 2. Comparison of acceleration factors [74].

cognition. As the particles approach the global region(s), they may drastically be turned by type as global experience has a great importance in the relatively late stage of evolution. In the later stage of evolution, the global experience is increased more rapidly due to the improved green curve of c_2 in Fig. 2(b).

3.3.4. Choice of control parameters and their values

Choosing which control parameters to be tuned to solve a given problem is one big challenge when using a given optimizer. Harrison et al. [77] quantified the importance of each of the three ω-PSO basic control parameters (i.e., ω, c_1, and c_2), according to their respective variances of fitness, by employing fANOVA (function ANalysis Of VAriance). The results indicated that the greatest sensitivity to the assigned values goes to the IW, which is thus the most important parameter to be tuned when solving low-dimensional problems by using an optimized PSO, as this specific setting gives a greater likelihood of reaching the target optimum position quickly. However, in general, the problem at hand itself is the guidance to guess reasonable values of the control parameters, by testing the proposed parameters on a set of common benchmark functions. For instance, Isiet and Gadala [54] have recently conducted an extensive sensitivity analysis to determine the optimal parameter combination. When tackling a constrained optimization problem (discussed in Section 4.2), the results revealed that the IW and the acceleration coefficients were the most influential parameters in PSO. Generally, what are the approaches to find the best values for control parameters? Some scholars slacken and prefer to just use the values published in the literature, others find it better to use fine-tuned standard static values. However, these approaches were found to be unsatisfactory, especially since the ideal parameters may be time-dependent. Therefore, several self-adaptive and dynamic approaches have been developed, as discussed in the previous subsections.

3.3.5. Can we dispense with some control parameters?

Arumugam and Rao [78] modified the ω-PSO with no additional complex computational efforts, forming another PSO variant (M-BPSO). These modifications were made to ameliorate the drawbacks of ω-PSO associated with premature convergence and weak ability of local search and to lessen the control parameters while preserving efficient performance. In M-BPSO, a dynamically decreased particle velocity limits were used instead of the IW parameter to balance the local and global search activities, which indicates that the effectiveness of PSO algorithms is not closely associated with their standard control parameters such IW and can perform well without any of these parameters. Experiments on M-BPSO revealed that the random vectors ($\mathbf{r_1}$ and $\mathbf{r_2}$) may be compensated for other exploration catalyzers (e.g., position clamping) and that the acceleration coefficients may also be discarded in the velocity updating Eq. (8), thus weighing the global optimal solutions [78]. Moreover, the computational effectiveness and efficiency of different variants of M-BPSO were tested based on extensive numerical simulations, and the strength of the algorithm was found to heavily lie on how quickly that algorithm can locate a near optimal solution by combing the search space, taking into account to refine the results by exploiting the neighborhood experience using the dynamically decreased velocity limits (DDVL [78]).

3.4. Swarm adaption

3.4.1. Swarm initialization methods

In fact, uniform or Gaussian distributions are usually preferred by the PSO researchers to randomly generate the particle coordinates at the initialization stage, which unfortunately disallows the particles to cover the search space completely. A number of researchers have tried to use other swarm initialization methods for improving the diversity in PSO. In [79], the impact of alternative initialization functions on the optimization performance was studied based on logarithmic, normal, and log-normal distributions. It was clear that different initialization strategies can, in some cases, largely improve the convergence speed in the case of benchmark functions. Farooq et al. [60] proposed a different method by using Generalized Opposition-based Learning (GOBL) to initialize the swarm particles with already fittest particles. Moreover, to equalize the proportions of exploration and exploitation capabilities during the search process, a linearly decreasing IW pattern has been proposed. The projected changes increase the overall performance of PSO, especially on noisy optimization problems.

Chaotic PSO (CPSO) was investigated in [80] using several chaotic maps, including Chebyshev, Circle, Iterative, Logistic, and Sinusoidal. Three variants of binary CPSO were considered: ChPSO1 with chaos in both initialization and the PSO body, ChPSO2 with chaos only in particles, and ChPSO3 with random initialization of particles and chaos is inserted in the PSO body. The initialization executed on ChPSO1 and ChPSO2 validated the positive impact of chaos on PSO [80]. In [81], two variants, CPSO-I and CPSO-II, were proposed to find the best path planning based on Bèzier curve. In CPSO-I, r_1 is a chaotic number between 0 and 1, while in CPSO-II, r_2 is a chaotic number between 0 and 1. Different chaotic maps were used to test the results of the two proposed algorithms. For both algorithms, the results of the experiments showed that the Singer and Sine maps were very suitable.

Recently, swarm initializations based on 22 different probability distributions were applied and their influence on the performance of five different algorithms has been studied systematically [82]. It has been found that the proposed initialization mechanism does not significantly affect some algorithms, such as DE (i.e., DE is more robust, while this initialization method has more significant sensitiveness for others like, for example, PSO). However, the nature of the optimization problem itself is involved in the process of examining such sensitivity. Empirical results showed that the most appropriate initialization methods for ω-PSO were random, beta distribution, and Latin Hypercube Sampling (LHS). This might explain why the uniform random initialization, most preferred in the literature, performed highly in many PSO variants.

3.4.2. Swarm size adaption

In most applications, many authors restrict the swarm size to 20-50 particles, effortlessly following the initial suggestion from 1995, even though it is a basic control parameter and one of the most difficult parameters to tune in most meta-heuristics [25]. In a recent study [83], the performance of eight PSO variants has been tested over 3 up to 1000 particles composing swarm sizes. Sixty 10- to 100-dimensional scalable benchmarks and twenty-two 1- to 216-dimensional real-world problems were used to conduct tests. Results did differ for the specified PSO variants; however, the overall best performance was obtained with 70-500 particles composing swarms for the majority of the considered PSO algorithms, that indicates the extremely small size of the classical choice. For more difficult problems and practical applications, larger swarms frequently improve the efficiency of the method. For unimodal problems, some PSO variants perform best with hundreds of particles; however, slightly lower swarm sizes are recommended in many special cases. To verify these implications, in [82], it has been demonstrated that the ω-PSO usually requires a larger swarm size, while the DE can give better results with smaller swarm size and larger number of iterations. However, the above findings are preliminary, and different initialization methods are problem-dependent and may offer different performance for different algorithms [82]. On the other hand, the correlation between the number of iterations, swarm size, and the objective function value has been studied in [84]. To guarantee excellent Radio Frequency Identification (RFID) performance, a 300–1000 number of iterations with a swarm size of 100–300 gave a larger objective function value, more than 13000 [84]. Besides, the lowest objective function value below 9000 was given when 200 iterations and 800 particles were used. In conclusion, it has been settled that the best setting is 200 number of iterations, and a swarm size of 800 for the RFID problem.

3.5. Neighborhood topologies

The structure of the swarm in a swarm-based meta-heuristic algorithm is very effective, as emphasized by Kennedy [85], and witnessed in the literature [86,87]. The purpose of the neighborhood topology or sociometry is to enhance swarm diversity when dealing with multimodal problems [88], the mechanism of information exchange in the swarm guides the search trajectories of particles [89]. The most common swarm structure is the *panmictic*, in which mating selection is random and all individuals can interact with each other during the exploration process. Hence, it encourages rapid information flow and consequently may lose swarm diversity, which may lead to premature convergence. While, this issue was also addressed in the literature by building a swarm structure based on neighborhood or topological properties [86,87]. In structured swarms, information exchange depends on fitness as well as topological relationships, and it is more common between individuals that are close. Thus, the spreading of good solutions throughout the swarm is slowed down, thereby discouraging stagnation and premature convergence. Generally, the swarm structuring approaches can be categorized into two basic groups: cellular/fine-grained approaches [90] and distributed/coarse-grained approaches [91]. The neighborhood topologies of these approaches are depicted in Fig. 3.

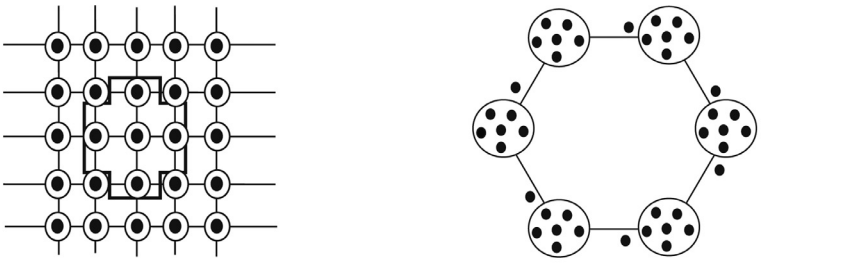
The first approach works in a diffusion manner, where the swarm is distributed in a grid fashion, where each individual in the swarm has relative coordinates and is allowed to interact only within the neighborhood. Thus, because of the slow information exchange among neighborhoods, the problem of premature convergence is mitigated. On the other hand, in the distributed/coarse-grained approach, the swarm is divided into several smaller groups, so they evolve independently on separate islands, and swarm individuals are allowed to migrate between the islands. It allows the divided sub-swarms to explore the search regions more rigorously, in order to maintain the swarm diversity [92].

It is common in PSO that the swarm is randomly initialized, and each individual performs the search, driven by the most successful individual(s) in the swarm. The neighborhood topology determines the breadth of influence both on the individuals and the algorithm performance [89,93]. A PSO algorithm with panmictic structure may prematurely converge by losing swarm diversity. To address this, one may instead use a structured neighborhood topology [36]. Nevertheless, because of the numerous benefits, various neighborhood topologies have been proposed for PSO algorithm. However, in this regard, a comprehensive literature review on the developed PSO neighborhood topologies is essential. Therefore, we attempt to provide a comprehensive survey of the neighborhood topologies proposed for PSO, thereby encouraging researchers to introduce these topologies to other swarm-based algorithms to achieve the highest possible performance.

The major neighborhood topologies proposed for the PSO algorithm are organized in Fig. 4. The two main structured swarm approaches, cellular and distributed models, are reported in this review as follows.

3.5.1. Cellular topology approach

In this approach, a candidate swarm is divided into several tiny subswarms typically composed of only one individual, by arranging the individuals in the toroidal mesh, so that the interactions occur only within the small neighborhood around. Here, the neighborhoods are supposed to be partially overlapped to allow slow information diffusion



(a) Cellular/Fine-grained approach. (b) Distributed/Coarse-grained approach.

Fig. 3. Cellular approach versus distributed approach.

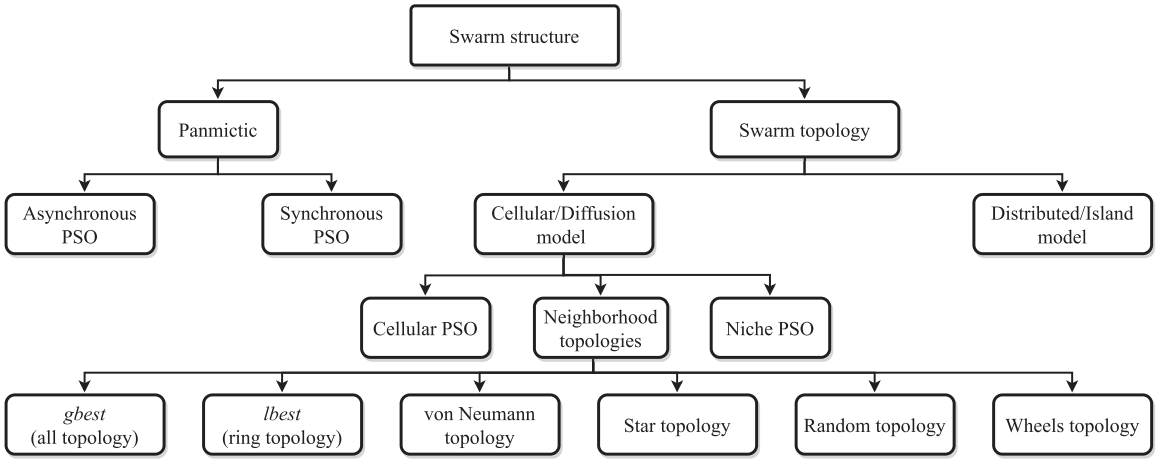


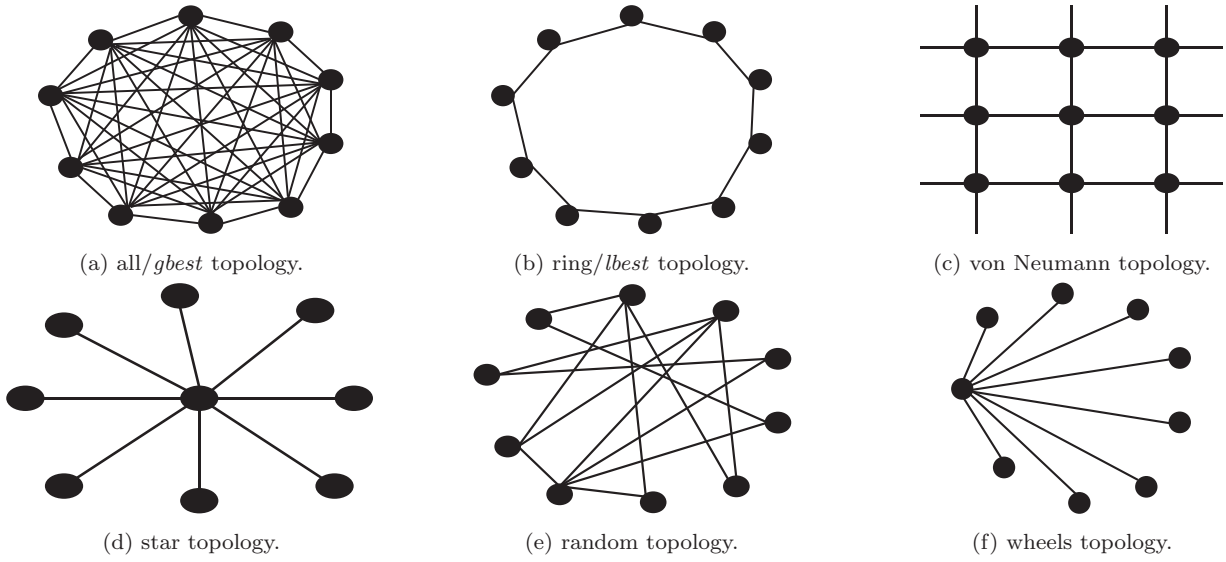
Fig. 4. Neighborhood topologies studied for PSO.

to maintain the swarm diversity [90]. Because of potential inbreeding within a neighborhood, the characteristics of the cellular approach may cause speciation and incline the swarm to form niches [94]. However, the swarm tends to become homogeneous over time [93], making convergence much slower so that the chances of premature convergence to a local optimum are mitigated. Therefore, it is important to understand their influence on the performance keeping in view both the *lbest* and *gbest* topologies in PSO [89,95,96]. Moreover, these neighborhood topologies are often viewed as the generalization of the diffusion approach. Some of the popular cellular topologies proposed for PSO are discussed as follows.

Cellular PSO: To address the diversity loss problem in PSO, Hashemi et al. [97] proposed two-dimensional cellular automata using cellular PSO. The performance of the proposed algorithm was validated on the moving peaks benchmark problem, and it was found that the proposed PSO outperformed the compared mQSO, a PSO model for dynamic environment. Later in, many other studies also proposed other cellular versions of PSO. Shi et al. [90] proposed a cellular PSO algorithm by integrating the lattice cellular model into PSO. In this work, three different lattice cellular structures, namely, cubic, trigonal, hexagonal, were employed as the neighborhood topologies where the individuals could only exchange information within their neighborhood. The experimental analysis revealed its efficacy. Moreover, a similar approach was combined with a constraint handling technique and proposed to solve a nonlinear constraint optimization problem in [98].

In the canonical PSO [2], the *gbest* topology is used where every particle is attracted to the best solution found so far. Here, all particles can communicate with each other and the topology can be represented by a fully-connected social network graph. Eberhart et al. [99], on the other hand, proposed that each particle is also attracted by the local best solution (*lbest*), within the local neighborhood topology. Here, only one single neighborhood particle (*gbest* particle) exists in the entire swarm. Conversely, for the *lbest* PSO and von Neumann PSO, there is a distinct neighborhood with its own *nbest* particle for each particle in the entire swarm, and each particle can thus be the *nbest* of multiple neighborhoods (the *nbest* technique is discussed in Section 4.3.2). Kennedy [89] investigated different neighborhood topologies, such as circles, wheels, stars, and randomly assigned edges while proposing a small-world social network with PSO. The author found that the sociometry of a swarm can significantly affect its efficiency in finding the optimum. Moreover, it was suggested that the swarms with fewer connections might perform better on solving multimodal problems, while a highly interconnected swarm might show good results while solving unimodal problems. Based on these findings, Kennedy and Mendes [85] used heterogeneous swarm structures, with some tightly connected sub-swarms while others are relatively isolated. The study observed the effect of *gbest, lbest*, star, small, pyramid, and von Neumann neighborhood topologies by varying the number of neighbors. The neighborhood topologies of fully-connected (all)/*gbest*, ring/*lbest*, wheels, random, von Neumann, and star are illustrated in Fig. 5.

Another topology called dynamic topology was first proposed by Suganthan [100]. Here, the author selected a small number of particles at the early iterations based on Euclidean distance. Later, the the neighborhood size is created dynamically over consecutive iterations until the fully-connected particles are formed towards the end of iterations. Thus, the search commences with an Euclidean distance based (instead of swarm index based) *lbest* topology and gradually changes to *gbest* topology by the end of the search. Similarly, Wang et al. [101] proposed a dynamic tournament topology strategy, wherein several better solutions chosen from the entire swarm were utilized to guide each particle. The proposed approach encourages the sharing of information across a wider network. Although stochastic, the selection of better particles still favored particles with better solutions. It is worth mentioning that the proposed approach is especially pronounced on solving complex problems. In another work, Xia et al. [102] introduced a new Multi-Swarm PSO approach (MSPSO) combined with the Dynamic sub-swarm Number Strategy (DNS), Sub-swarm Regrouping Strategy (SRS), and Purposeful Detecting Strategy (PDS). In this technique, in the early search, the DNS divides the entire swarm into many sub-swarms and periodically increases the size of each sub-swarm in the evolutionary process.



(a) all/*gbest* topology. (b) ring/*lbest* topology. (c) von Neumann topology.

(d) star topology. (e) random topology. (f) wheels topology.

Fig. 5. Neighborhood topologies in particle swarms.

This helped in maintaining the exploration-exploitation balance. Second, with a special number of sub-swarms in each DNS iteration, the SRS used the stagnancy information of the global best position to regroup these sub-swarms for boosting the exploitation with the help of sharing the search information among different sub-swarms. Lastly, the PDS detects whether the PSO was dropped into a potential local optimum based on some historical information of the search process for jumping out of that local region.

Niche PSO: A niching PSO called NichePSO was proposed by Brits et al. [103], where the swarm is divided into sub-swarms based on the fitness of the individual particles, while the sub-swarms utilize the guaranteed convergence [104] to locate multiple optimal solutions in multimodal optimization problems. The experimental results showed that NichePSO successfully located all maxima [103]. Huang et al. [105] also used the niching technique for emergent multimodal optimization of PSO algorithms to extend conventional unimodal optimization to challenge multimodal optimization of composite structures. The authors claimed that the optimal designs of composite structures are by nature typical multimodal optimization problems in the real-world.

In [106], a new niching PSO with equilibrium factor named as E-SPSO was proposed. Different from the existing niching PSOs, in this work, the numbers of particles in different niches were kept in balance in E-SPSO. The velocity of each particle was influenced by not only the personal best particle and the global best particles, but also by an Equilibrium Factor (EF). By using the equilibrium factor to update the velocities of particles, the particles were allocated uniformly among the niches. In this way, the computation resources were assigned to the niches in a more balanced manner, so the algorithm could gain more swarm diversity for finding high-quality solutions in all niches.

3.5.2. Distributed topology

In this topology, the candidate swarm of PSO is subdivided into smaller islands/sub-swarms called “demes”. Here, each island/sub-swarm is isolated from the others, as well as evolves independently and simultaneously. In this approach, the information exchange between particles happens in the form of selected individuals moving between sub-swarms. During the migration process, one or more individuals migrate to one or more islands, given a certain migration policy. The islands may exchange migrants with other islands at a fixed predefined time for all islands or at independent times. This topology enhances exploration when the sub-swarms are non-interacting; otherwise, it enforces exploitation on the occasion of migration between the islands. In this way, independent evolution and migration schemes delay the stagnation of evolution and provide exploration-exploitation trade-off for swarm-based algorithms.

To talk about some seminal works with regards to distributed topology, the following are discussed. Earlier in 2005, a Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO [107]) was introduced, wherein the swarms are dynamic and the swarms’ size is small. In this approach, various regrouping schedules are used to regroup the swarms frequently so that information is exchanged among the swarms. Later in, two extended local search DMS-PSO methods, [108,109], were presented for large-scale global optimization and handling constrained real-parameter optimization problems, respectively. In [108], quasi-Newton method was incorporated, while a sequential quadratic programming method was combined in [109], in order to improve the local searching ability of DMS-PSO. A novel MSPSO with Dynamic Learning Strategy (DLS) was proposed by Ye [110]. The research was proposed to promote information exchange among sub-swarms, using a particle classification mechanism that advocates that the particles in each sub-swarm are classified as ordinary particles. In MSPSO-DLS, the ordinary particles focus on exploitation under the guidance of the local best position in its sub-swarm, while the communication particles with dynamic ability focus on exploration in a new search region under the guidance of a united local best position, to promote information exchange among sub-swarms. Ayari and Bouamama [111] presented a new dynamic distributed PSO algorithm (D^2PSO) for trajectory path planning of multiple robots, in order to find collision-free optimal path for each robot in the environment. The proposed approach calculated two local optima detectors, LOD_{pBest} and LOD_{gBest}. Particles with unimproved personal best and global best for a predefined number of successive iterations were replaced with restructured ones. Stagnation and local optima problems were resolved by adding diversity to the swarm, without losing the fast convergence characteristic of PSO.

Akhmedova et al. [112] proposed a soft island model topology where the initially defined swarm is virtually separated into several sub-swarms such that the connection between individuals from different sub-swarms is probabilistic. In the same context, a Discrete Parallel PSO variant (DPSO) was proposed by Ikegami and Mori in [113]. The proposed DPSO algorithm adopted the island model of parallel computation, wherein an adaptive PSO algorithm is integrated with the sigmoid function to handle combinatorial optimization problems. A recent survey of parallel and distributed PSO algorithms can be found in [114].

As such, the different PSO architectures branch out into: static neighborhood, wherein, while PSO is executed, the neighborhood does not change; and dynamic neighborhood, in which the neighborhood

changes based on e.g., the distance among particles in the search space or the number of iterations.

3.6. Fitness landscape analysis

Meta-heuristics have proven successful in solving many real-world optimization problems which are usually very complex. However, it remains a great challenge for researchers to choose the best approach. One approach to this dilemma is to use fitness landscape analysis [115]. The complexity of fitness landscape itself may be the reason behind the the very few techniques in practice based on fitness landscape analysis. In order to make fitness landscape analysis more accessible, this section provides an overview of the basic concepts and characteristics of the fitness landscape, highlighting the ways of adapting techniques to be appropriate or more feasible. Moreover, a set of factors that can influence problem difficulty are addressed, with particular regard on how to shift the focus towards measuring characteristics in lieu of predicting problem hardness. To make the idea more realistic, the ω-PSO search behavior based on Fitness Landscape Characteristics (FLCs) is to be discussed in Section 3.6.3.

3.6.1. Fitness landscapes

Optimization is the task of selecting the best element from a set, based on some criteria. An optimization problem can be defined by an objective function $f : S \rightarrow \mathbb{R}$, which maps candidate solutions from a search space S to real values $\mathbb{R}$, representing the fitness[1] of the chosen solutions. Optimization problems with only a single global optimum are called unimodal. Optimization problems with additional non-global (local) optima are called multimodal. For objective functions defined in one or two dimensions, one can visualize a fitness landscape (a concept attributable to Wright [116]): the vectors of candidate solutions correspond to coordinates on the terrain, and the fitness of those solutions are represented by the elevation of the terrain at those positions. For maximization problems, the optima are situated at the peaks of mountains in the landscape. Similarly, for minimization problems, the optima are situated at the depths of valleys in the landscape.

When regarding an optimization problem (whose complexity is too large, the objective function is not in mathematical form, or noise or uncertainty is exhibited by the objective function) as a fitness landscape, interesting techniques for analyzing the problem become available. For example, consider a hill climbing optimization technique. A hill climbing algorithm randomly selects an initial solution from the search space, and then iteratively replaces it with a fitter solution, until no fitter solution can be found. Hill climbing is considered as a local optimization technique, and as such, is susceptible to fail on multimodal optimization problems. A multimodal maximization problem can be visualised as a landscape with a global maximum at the peak of a single tall mountain, and local maxima at the peaks of smaller hills. It then becomes clear why the hill climbing technique is likely to fail: The hill climber simply rises to the peak of whichever hill is closest to its initial starting point; that hill is unlikely to be the tallest mountain in the landscape. In this example, the notion of a neighborhood is further implied: for a particular initial solution, the hill climber will only select a fitter solution from its immediate neighborhood, which is the set of solutions that are nearby enough. In addition to the set of candidate solutions and the objective function, the definition of the neighborhood of a solution is critical to the shape of the fitness landscape. For example, suppose that a hill climber solving an objective function defined in two dimensions is limited so it can only move to new solutions in a single direction. Despite the dimensionality of the objective function, the hill climber perceives the landscape as a one-dimensional function. For a deeper understanding, an example of PSO in 2D with both single-maximum fitness landscape and multimodal fitness landscape was recently introduced in [117].

[1] The term fitness here refers to the quality of a solution in the general sense, and is not limited to the context of evolution.

3.6.2. Fitness Landscape Characteristics (FLCs)

Similar to landscapes in nature, fitness landscapes can exhibit various characteristics. For example, a landscape can have cliffs, or be steep, or be rough. Such fitness landscapes may exhibit certain characteristics which, if measured, can provide insight into the difficulties that the optimization problems pose for the search techniques solving those problems. Analysis of the characteristics of a fitness landscape can further our understanding of what makes it easier or harder for search techniques to solve optimization problems. This study focuses on the ruggedness, neutrality, gradients, global landscape structure (underlying modality), and deception of fitness landscapes. These basic landscape characteristics, along with a scalar-valued metric for quantifying the presence and severity of each characteristic, are discussed briefly in Table 3, which could help understand the complex optimization problems to be solved before making a decision on the most appropriate algorithm.

3.6.3. Linking FLCs to PSO search behavior

In an attempt to link Fitness Landscape Characteristics (FLCs) to PSO search behavior, Engelbrecht et al. [118] studied the correlations between FLCs and the search behavior of a suite of PSOs which are known to exhibit convergent behavior. The DM metric, which indicates the underlying structure of a landscape, was found to be imprecise at times. The metric depends on a parameter n_b, which determines the number of fitter positions, the dispersion of which should be taken. However, the optimal choice for this parameter is problem-specific. Additionally, for any choice of n_b, the metric may still yield results that are not indicative of the true structure of the landscape. A modified metric, namely, DM_{med}, was proposed to provide more reliable results. The DM_{med} metric, which is obtained as the median of DM values obtained for various n_b-values between 2 and $\frac{n_a}{2}$, was shown to provide more accurate measurements of a landscape's global structure in cases where the unadapted DM metric failed. The link between single FLCs and PSO search behavior was also investigated in [118] by computing the correlation coefficients between each FLC metric and Diversity Rate-of-Change (DRoC) measurements on a suite of benchmark functions. A number of possible relationships between FLC metrics and PSO search behavior were found. PSOs tended to reduce their diversity at a faster rate when solving landscapes that were indicated to be more searchable by PSOs, landscapes that were less deceptive, single-funnelled landscapes, and landscapes with less variation in their gradients. Neutrality and gradients were not strong indicators of PSO behavior. More ruggedness in landscapes was associated with faster convergence, not with slower convergence as one might expect.

From the findings above, it seems that when the FLCs values indicated a landscape to be easier to optimize, the PSOs were able to converge to a promising region at a faster rate. This was the case for landscapes that could be considered easier to solve due to being more searchable, being less deceptive, having simpler underlying structures, and having less variation in gradients. The exceptions were that landscapes did not correlate with faster convergence when those landscapes exhibited less ruggedness, less neutrality, or shallower gradients. These findings may form a basis upon which an algorithm selection framework can be founded by providing a deeper understanding of the link between optimization problems and search algorithm behavior. Thus, fitness landscape analysis enables the audience to predict how a search algorithm will behave on an optimization problem, which may allow optimization researchers to choose an optimal algorithm for any given problem, based on a quick analysis of the characteristics of the problem.

4. PSO in complex optimization scenarios

A significant advance in research has been made in the last two decades to adopt PSO for optimization in complex environments, including high-dimensional optimization, nonlinear constrained (equality and inequality) optimization, multimodal optimization, multi-objective optimization, and discrete hyperspace optimization. Such optimization processes have been tackled with numerous PSO approaches developed over the past two decades. This section addresses some of these approaches.

4.1. Large-scale optimization

In this special category of optimization problems, extremely large values can be rendered from a very high number of constraints, objectives, and/or decision variables. To locate the optimum solution in a high-dimensional (≥ 100 dimensions) space for real-valued optimization problems can be typically very challenging. This is very expected, especially with the exponential increase of dimensions in the lookup area. In higher dimensionalities where distance measures are usually broken down, or even in moderately high-dimensional search spaces, a search

Table 3
Fitness landscape characteristics.

Brief description	Scalar-valued metric(s)	Quantifying the presence and severity
Ruggedness - Ruggedness refers to the level of variation in a fitness landscape with numerous "ups and downs", or sudden changes in fitness. - The two landscapes exhibiting ruggedness in Fig. 6(a) have similar underlying structures, the landscape on the left is smooth, while the landscape on the right is rugged. - Rugged landscapes solve for certain search techniques due to their many local optima [119,120].	Ruggedness of a landscape can be estimated on the micro- and macro-level for micro- and macro-ruggedness (FEM0.01 and FEM0.1, respectively), using the first entropic measure (FEM [121]). The measurements are obtained as follows: - Based on a progressive random walk of n_t steps, given a time series of fitness values $f_1 \dots f_{n_t}$, overlapping three-point groups are sampled from the walk as a time series of a string of symbols $S(\varepsilon) = s_1 \dots s_{n_t-1}, s_i \in \{\bar{1}, 0, 1\}$, according to $S_i = \begin{cases} \bar{1} & \text{if } f_i - f_{i-1} < -\varepsilon, \\ 0 & \text{if } \lvert f_i - f_{i-1} \rvert \leq -\varepsilon, \\ 1 & \text{otherwise,} \end{cases}$ where ε controls the allowable error between finesses that will be considered to be equal. - The entropy of the subset of rugged groups is estimated using an information function $H(\varepsilon)$ of the probability distribution of the overlapping rugged elements within the group. - To obtain a scalar-valued metric of ruggedness, $H(\varepsilon)$ is calculated for various $\varepsilon \in [0, \varepsilon^*]$, where ε^*, also called the information stability, is the smallest value of ε for which the landscape becomes completely neutral. - The largest value of $H(\varepsilon)$ is taken as the metric of ruggedness.	- A group of two adjacent symbols, s_i and s_{i+1}, is considered as a rugged group if $s_i \neq s_{i+1}$ - For larger values of ε, more finesses are considered to be equal, and the landscape becomes more neutral. - The result of $H(\varepsilon)$ is a value in [0,1], where 0 indicates a flat landscape, and 1 indicates maximal ruggedness.
Neutrality - Neutrality* refers to sections of the landscape where neighboring points have nearly equal fitnesses. At such sections, the slope of the landscape is approximately zero. - The two examples of landscapes with neutrality in Fig. 6(b) show that: The landscape on the left has a large connected neutral area on the left and a smaller connected neutral area on the right; and the landscape on the right has many small neutral areas. - Zero-slope areas of the search space in neutrality offer no information to guide the search in a particular direction, which poses a challenge for some search algorithms that may falsely interpret the lack of variation in fitness as an indication of converging to an optimum, when the algorithm might, in fact, just be moving through a neutral area.	First of all, the search space is sampled using progressive random walks of n_t steps, giving a time series of fitness values $(f_1 \dots f_{n_t})$. A string of symbols $S(\varepsilon) = s_1 \dots s_{n_t-2}, s_i \in \{0, \bar{0}\}$ is generated from the fitness time series, according to $S_i = \begin{cases} 0 & \text{if } \max\left(\{f_{i-1}, f_i, f_{i+1}\}\right) - \min\left(\{f_{i-1}, f_i, f_{i+1}\}\right) < \varepsilon, \\ \bar{0} & \text{otherwise,} \end{cases}$ where ε is the neutrality threshold used by van Aardt et al. [122] of 1×10^{-8}. Two neutrality metrics proposed by van Aardt et al. [122] are obtained as follows: - The proportion of a landscape's terrain that is neutral can be estimated by the Proportion of Neutral Structures (PNS) in a random walk as $\text{PNS}(\varepsilon) = \frac{n_n}{\lvert W \rvert}$, where n_n is the number of neutral structures s in $S(\varepsilon)$, where $s = 0$, and $\lvert W \rvert$ is the overall number of structures in $S(\varepsilon)$. - The connectedness of the neutral terrain can be estimated by the length of the Longest subsequence of Neutral Structures (LSN) of such a random walk, given by $\text{LSN}(\varepsilon) = \frac{\max(\{\lvert w_n \rvert\})}{\lvert W \rvert}$, where $\max\left(\lvert w_n \rvert\right)$ is the maximum number of steps in any neutral-only subsequence w_n of $S(\varepsilon)$, such that $s = 0 \, \forall s \in w_n$.	- $s_i = 0$ indicates a neutral 3-point structure in the walk, and $s_i = \bar{0}$ indicates a non-neutral structure. - Higher values for ε result in more structures being considered neutral. - PN(ε) and LSN(ε) are a value in [0,1], where 0 indicates that there is no neutrality in the landscape, and 1 indicates that the landscape is completely neutral. - PN(ε) gives an indication of overall neutrality, while LSN(ε) indicates the size of neutral plains in the landscape.

(continued on next page)

Table 3 (*continued*)

Brief description	Scalar-valued metric(s)	Quantifying the presence and severity
Gradients - Gradients measure the magnitude of neighboring fitness changes in terms of steepness. - The landscapes in Fig. 6(c) have similar shapes, but the basins of the local optima of the landscape on the right are steeper than those of the landscape on the left. - Some search algorithms may specifically struggle to optimize a landscape that has local optima in very deep basins.	At first, the search space is sampled using a progressive Manhattan walk of n_t steps, where each step updates the position by a fixed step size u in a single randomly selected dimension, producing a sequence of the normalized gradient $(g_1, g_2, \dots, g_{n_t-1})$ of each step in the walk, which is generated according to $g_t = \frac{(f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t))/(f_{\max} - f_{\min})}{u/\left(\sum_{j=1}^{n} x_{\max,j} - x_{\min,j}\right)}$, where u is the fixed step size used in the progressive Manhattan walk, $f_{\max}$ and $f_{\min}$ are respectively the maximum and minimum fitness values encountered during the walk, and $x_{\max,j}$ and $x_{\min,j}$ are the search space bounds in dimension j. Now, the severity of gradients in a landscape can be estimated by two metrics [123] derived from the sequence g, as follows: - The average gradient G_{avg}, given by $G_{avg} = \sum_{t=1}^{n_t} \lvert g_t \rvert / (n_t - 1)$. - The standard deviation of the gradients G_{dev}, given by $G_{dev} = \sqrt{\left(\sum_{t=1}^{n_t} (G_{avg} - \lvert g_t \rvert)^2\right)/(n_t - 2)}$.	- G_{avg} and G_{dev} are positive real numbers. - Higher values of G_{avg} indicate the presence of steeper slopes. - Higher values of G_{dev} indicate highly contrasting gradients (i.e., the gradients are vertical or very steep at some areas, or may be close to neutral at other areas of the landscape). - Lower values of G_{dev} indicate that the gradients are more evenly distributed across the landscape, and therefore G_{dev} is a good indicator of the general gradient of the landscape.
Global landscape structure - , also called funnel, is a global basin shape of clustered local optima. - Both landscapes in Fig. 6(d) are similarly multimodal and rugged, with an underlying unimodal structure (on the left) and an underlying multimodal structure (on the right). - For certain algorithms, an underlying unimodal structure can ease landscape optimization, while an underlying multimodal structure can harden the optimization.	First, a uniform random distribution of the search space is used to sample n_a positions, each dimension of which is normalized in [0,1]. Then, a subset $S^*_{n_b}$ of the n_b fittest positions is determined, where $n_b < n_a$. Thus, the underlying modality of a landscape can be estimated by: - Dispersion Metric (DM), introduced by Lunacek and Whitley [124], given by $\mathrm{DM} = \mathrm{disp}\left(S^*_{n_b}\right) - \mathrm{disp}_D$, where $\mathrm{disp}\left(S^*_{n_b}\right)$ is the average pair-wise Euclidean distance between the positions in $S^*_{n_b}$, and D is the dimensionality of the search space, and disp_D is the average pair-wise Euclidean distance of a large sample taken from $U(0,1)^D$.	- DM is a value in $[-\mathrm{disp}_D, \sqrt{D} - \mathrm{disp}_D]$, where negative values indicate a landscape with an underlying unimodal structure, and positive values indicate a landscape with an underlying multimodal structure.
Deception - Deception is the information that leads away from the global optimum, and a landscape's difficulty is influenced by the availability and quality of the information (gradients) that guides the search towards the global optimum. - The gradients lead towards and away from the global optimum in the left and right landscapes of Fig. 6(e), respectively. - A landscape with readily available, high -quality information is easier to solve than the one where such information is scarce and deceptive.	- A sample of n_a points is taken from the landscape, along with their associated fitnesses $F = f_1, f_2, \dots, f_{n_a}$. The fittest point $\mathbf{x}^*$ of the sample is determined. The Euclidean distance from each sampled point to $\mathbf{x}^*$ is determined, giving $\mathrm{Dist}^* = d^*_1, d^*_1, \dots, d^*_{n_a}$. The Fitness Distance Correlation (FDC) metric by Jones and Forrest [125] was adapted to substitute the fittest position $\mathbf{x}^*$ (which should be pre-known for the metric, thus attaining unsuitability for some practical scenarios) for the global optimum: - The metric is the correlation between F and Dist^*, given by $\mathrm{FDC} = \frac{\sum_{i=1}^{n_a} \left(f_i - \overline{f}\right)\left(d^*_i - \overline{d^*}\right)}{\sqrt{\sum_{i=1}^{n_a} \left(f_i - \overline{f}\right)^2} \sqrt{\sum_{i=1}^{n_a} \left(d^*_i - \overline{d^*}\right)^2}}$, where $\overline{f}$ is the mean of F, and $\overline{d^*}$ is the mean of Dist^*.	- FDC is a value in $[-1, 1]$. - Assuming minimization, higher values of FDC indicate landscapes with more high-quality information to guide the search towards the global optimum.

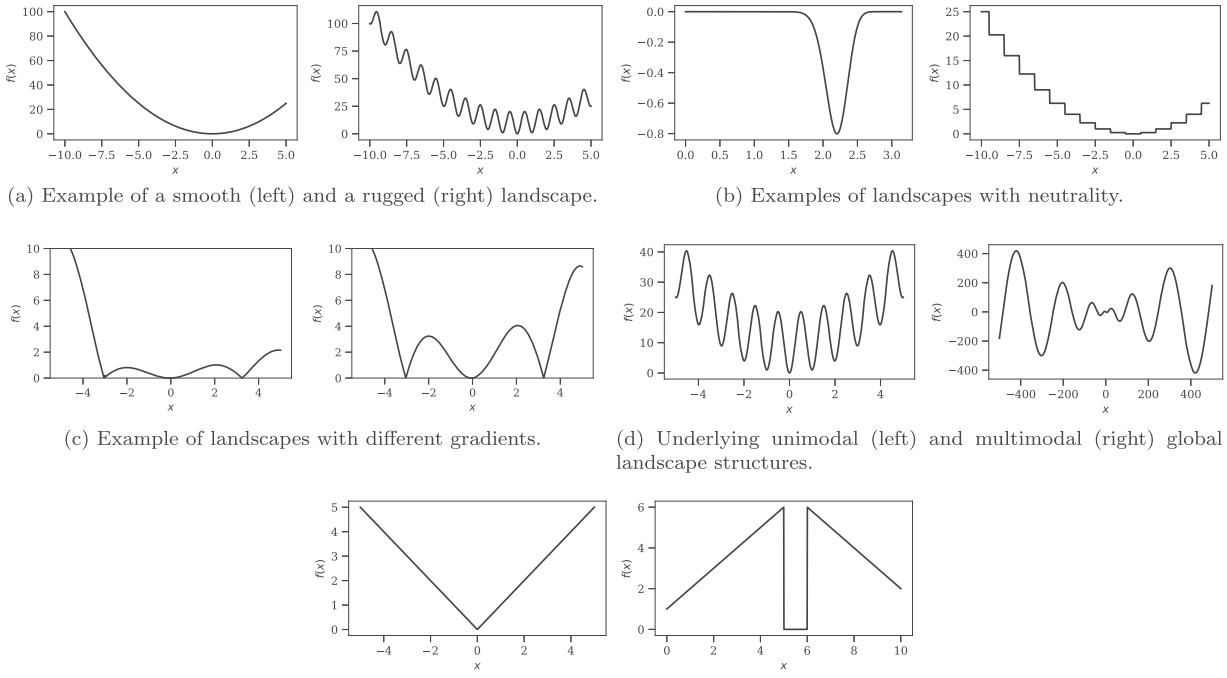
* Neutrality is distinct from smoothness in that a smooth landscape is not rugged, but may still have a nonzero slope to lead search algorithms towards fitter regions of the landscape.

strategy which is suitable for small dimensions might be useless here. Large-scale problems are usually addressed according to the following definition:

$$\min/\max\left(\left\{f(\mathbf{x}) = f\left(x_1, x_2, \dots, x_D\right)\right\}\right), D \geq 100, x \in X \subseteq \mathbb{R}^D,$$

where $X \subseteq \mathbb{R}^D$ denotes the decision space, $\mathbf{x} = \left(x_1, x_1, \dots, x_D\right)$ stands for the decision vector, and $f : X \to \mathbb{R}$ represents a real continuous objective function for the mapping from D-dimensional to one-dimensional space with fitness value $f(\mathbf{x})$.

Over the past twenty years, PSO has witnessed several modifications to work efficiently in high-dimensional spaces. Compared to other population- and swarm-based meta-heuristics, such as EAs, ACO, and ABC, PSO similarly struggles regarding large-scale problems. This means that the performance of PSO clearly becomes deteriorated as the dimensional size of the problem to be tackled increases, because the landscape complexity and characteristic alteration are elevated and the search space is exponentially enlarged. The GA and PSO algorithms have similarities that have encouraged researchers to propose PSO variants with



(a) Example of a smooth (left) and a rugged (right) landscape.

(b) Examples of landscapes with neutrality.

(c) Example of landscapes with different gradients.

(d) Underlying unimodal (left) and multimodal (right) global landscape structures.

(e) Landscapes with gradients leading towards (left) and away from (right) global optima.

Fig. 6. Examples for fitness landscape structures.

the help of operations found in GA. The Cooperative PSO (CPSO), an example of such variants, was proposed [126] and then refined [127] by van den Berg and Engelbrecht for large-scale optimization problems. The concept of cooperation used in GA was incorporated in the CPSO algorithm, wherein all sub-swarms contribute and exchange information, thereby exhibiting cooperation. According to van den Berg and Engelbrecht, the application of this concept to PSO can be achieved by replacing the idea of a single swarm working in all dimensions with a number of swarms in each dimension. Thus, only 1-D vector will be optimized by each sub-swarm. Despite the simplicity of this approach, the original algorithm must undergo some changes, specifically to the objective function evaluation, which still requires the traditional input of d-dimensional array. Therefore, the problem of objective function evaluation was curbed by using a context vector which is built at every iteration, based upon a component from each dimension of the best particle. Then, if a component has a new fitness value better than the prior, this specific component in addition to the fitness value of the best individual are updated.

In the first variant of CPSO [126], the search space is divided into exactly d subspaces, wherein the search components may be correlated. Motivated by this idea, van den Bergh and Engelbrecht [127] later proposed the CPSO-S_k algorithm that splits the search space into k subspaces, where $k \leq d$, thus generating the generalized paradigm of the CPSO algorithm. The CPSO-S_k converges to the optima of the respective subspaces, increasing the algorithm's chance to trap into local optima. However, those authors supposed that CPSO-S_k has faster convergence than the original ω-PSO. On the other hand, comparing ω-PSO with CPSO-S_k, PSO is more unlikely to fall into local optima, because the dimensions are considered as a whole in the later case. Accordingly, van den Bergh and Engelbrecht [127] were stimulated to suggest a hybrid approach using CPSO-S_k and ω-PSO, namely, CPSO-H_k, taking the advantages of both algorithms to produce an improved local escape mechanism from a fast hybrid approach. When compared to ω-PSO, the algorithms CPSO-S_k and CPSO-H_k significantly performed better in terms of the overall performance and the quality of the solutions found, even on elevating the problem dimensionality.

Two Steps Forward, One Step Back: Prior to tackling CPSO-S_k and CPSO-H_k [127], ω-PSO related problem so-called "two steps forward, one step back" was observed by van den Bergh and Engelbrecht [126]. They found that the elements of the d-dimensional vector in the ω-PSO are changed at each iteration, allowing some components to approach the optimum, while others move away from it. Considering a minimization problem, PSO can evaluate a new candidate solution and accept that solution only if its current fitness value is lower than prior. In their paper, a vector of three components was adopted to exemplify this weakness of ω-PSO, wherein one of those components already has the optimal value. However, in the next iteration, its value changed poorly while the other two components as well as the fitness value improved. Thus, the algorithm was taken two steps forward and one step back, by improving two components while sustaining one unchanged. To overcome this problem, van den Bergh and Engelbrecht proposed that the objective function is evaluated once a component changes, while keeping the other $d - 1$ components' values unchanged.

The adaptive variables' weighting and random grouping schemes have shown high effectiveness for handling non-separable high-dimensional problems. Accordingly, Li and Yao [128] incorporated them into PSO, presenting a Cooperative Coevolving PSO algorithm (CCPSO). For non-separable functions ranging from 30 to 1000 dimensions, the proposed CCPSO algorithm revealed high performance when compared with a previously developed coevolving PSO algorithm. Subsequently, Li and Yao [129] managed to solve large-scale 2000-d optimization problems having real-valued variables by scaling new PSO variants, one of which is CCPSO2 algorithm. CCPSO2 builds on an early CCPSO algorithm which relies on a new position updating rule for the new points in the search space by using Cauchy and Gaussian distributions. In addition, CCPSO2 employs an effective variable random grouping technique and a scheme for dynamically determining the variables' coevolving sub-component sizes. When tackling large-scale, multimodal optimization problems, CCPSO2 has shown superiority over a cooperative coevolving DE algorithm, two existing PSO algorithms, and a leading EA sep-CMA-ES (Covariance Matrix Adaptation Evolution Strategies).

Recently, Ge et al. [130] proposed a cooperative hierarchical PSO framework to decompose large-scale problems into subproblems using a two-stage variable interaction reconstruction algorithm. The proposed algorithm has revealed a guaranteed convergence to the global optimal solutions when the problems are correctly decomposed. In [131], the authors proposed a Cooperative Asynchronous Parallel PSO algorithm (CAPPSO) with a new velocity calculation method that utilizes a cooperative model of sub-swarms. The asynchronous communication among the sub-swarms makes CAPPSO faster than a parallel and more accurate than the master-slave PSO (MS-PSO) when tested on big problems.

4.2. Constrained Optimization Problems (COPs)

For constraint handling with PSO, Parsopoulos and Vrahatis [132] proposed a constriction factor method, which includes a penalty function. This was, to the authors' best knowledge, the first trial to optimize Constrained Optimization Problems (COPs) using PSO. A penalty function is used to transform a COP into an unconstrained problem, by penalizing the objective function in case of breaching the conditions on the variables. Therefore, a standard unconstrained optimization algorithm is used to build and optimize a single objective function. A penalty function $F(\mathbf{x})$ is given by

$$F(\mathbf{x}) = f(\mathbf{x}) + h_p(t)H_p(\mathbf{x}), \mathbf{x} \in X \subseteq \mathbb{R}^D,$$

where $f(\mathbf{x})$ is the original objective function, $h_p(t)$ is a dynamically modified penalty value, and $H_p(\mathbf{x})$ is a penalty factor which is calculated as

$$H_p(\mathbf{x}) = \sum_{k=1}^{m} \theta\left(q_k(\mathbf{x})\right)\left(q_k(\mathbf{x})\right)^{\gamma\left(q_k(\mathbf{x})\right)},$$

where m is the number of inequality constraints, $q_k(\mathbf{x}) = \max\left(\left\{0, g_k(\mathbf{x})\right\}\right)$ for $k \in \{1, 2, \ldots, m\}$, $\theta\left(q_k(\mathbf{x})\right)$ is a multi-stage function, and $\gamma\left(q_k(\mathbf{x})\right)$ represents the penalty function power. Despite not considering the equality constraints h_i, we can transform them into two inequality constraints (i.e., $g_k(\mathbf{x}) \leq 0$ and $-g_k(\mathbf{x}) \geq 0$). Although a penalty function can be used to transform COPs into unconstrained problems, COPs require fine-tuning of more parameters (in this case, $h_p(t)$, $\theta\left(q_k(\mathbf{x})\right)$, and $\gamma\left(q_k(\mathbf{x})\right)$), in order to prevent premature convergence.

A more straightforward brute-force method called Feasible Solutions Method (FSM), was proposed by Hu and Eberhart [133] and Hu et al. [134], in order to optimize COPs more effectively. In their proposal, FSM preserves the whole feasible solutions found so far while combing the search space. After meeting a stopping criterion, all the problem's constraints may be then fulfilled by finding an optimal solution. Using the same problems to compare these two methods with FSM, better, average optimal solutions may be obtained when the parameters of the penalty function are fine-tuned. However, determining which constraint handling approach to be used might highly depend on the problem to be solved. He et al. [135] introduced a different constraint handling approach into PSO, known as fly-back mechanism. The idea in [135] is simple: in the search space, a particle's position is reset to its previous (feasible) one, upon flying to a non-feasible region.

On the other hand, a more advanced approach was proposed by Sun et al. [136], wherein a new feasible position $\mathbf{x'}_i^{t+1}$ is computed as soon as a particle falls into a non-feasible region as $\mathbf{x'}_i^{t+1} = \mathbf{x}_i^t + \boldsymbol{\alpha}\mathbf{v}_i^{t+1}$, where $\boldsymbol{\alpha}$ is a diagonal matrix having values in its diagonal set within the range [0,1]. Thus, if $\boldsymbol{\alpha}_{ii} = 1$ for $i \in \{1, 2, \ldots, D\}$, then $\mathbf{x}_i^{t+1}$ is marked as a feasible position. If $\mathbf{x}_i^{t+1}$ inhabits an unfeasible position, the particle i must then be brought back to a feasible position by adjusting $\boldsymbol{\alpha}$, which can be found, as suggested by Sun et al. [136], by

$$\min\left\{\left\{\prod_{k=1}^{m+2D} e^{\max\left(\left\{0, g_k\left(\mathbf{x}_i^t + \boldsymbol{\alpha}\mathbf{v}_i^{t+1}\right)\right\}\right)} \prod_{l=1}^{p} e^{\max\left(\left\{0, \left|h_l\left(\mathbf{x}_i^t + \boldsymbol{\alpha}\mathbf{v}_i^{t+1}\right)\right|\right\}\right)}\right\}\right\},$$

where m (the number of inequality constraints) and $2D$ (the search space dimensionality) are transformed into two inequality constraints. Then, similar to the ω-PSO, the algorithm proceeds until a termination condition is met. The suitability of that algorithm for solving COPs was successfully verified. However, when the optima were on the search space bounds, the algorithm turned out to be a bit dysfunctional.

4.3. Multimodal function optimization

The ω-PSO algorithm was extended to include multimodal function optimization as well (i.e., to include finding local optimal solutions, along with the global best positions for a system of equations or just one equation). Decision makers, in particular, would benefit from such type of optimization, since decisions can be made, having diverse optimal solutions at hand, despite the existence of, for example, physical and cost constraints. However, classical nonlinear programming techniques cannot be used to solve all these problems, as there are multiple local and global optima found. On the other hand, EAs and PSO can be used to find the optimum positions faster than other traditional optimization techniques [137]. However, only one optimum of a function can be found based on the design of ω-PSO, which requires some changes. Assuming a minimization problem, all desired minima can be, in fact, found by applying ω-PSO multiple times on the same function. Nevertheless, not all may be found, and the premature convergence problem can emerge as a result of fast convergence in this type of optimization, because ω-PSO (or EAs) may get trapped into a local optimum. Thus, the swarm diversity should be maintained before reaching the target optimum. Apparently, the *lbest* model can be put up to find multiple solutions, depending on that a candidate solution is provided by each neighborhood. However, one particle can be simultaneously in several neighborhoods, as well as that particle may have the best fitness along all those neighborhoods, thereby causing all these neighborhoods' particles to converge to the same point. Consequently, the algorithm will converge prematurely if that point is a local optimum, as the neighborhood shall be biased towards that position. Therefore, this kind of problem has been tackled with many approaches, and the next subsections implicitly describe the most relevant.

4.3.1. Objective function stretching

Parsopoulos et al. [137] were the first to introduce multimodal function optimization with ω-PSO. STretched PSO (STPSO), the first version of their algorithm, aimed mainly to find a global optimum of a multimodal function, without letting the algorithm fall into the trap of local optima. To this end, a function stretching technique was used to apply a two-stage transformation to the objective function when finding a local optimum. A function stretching $H(\mathbf{x})$ acts like a filter which transforms the original function into a more flatter surface, yet highlights potential local and global optima. As already said, upon finding a local optimum, this transformation is applied, forcing the rest of the swarm to move away from that position. After this transformation, $H(\mathbf{x})$ substitutes $f(\mathbf{x})$, and the algorithm continues to perform until a prescriptive termination criterion is fulfilled.

Parsopoulos and Vrahatis [138] have improved this method to be more effective and efficient in finding all globally optimal solutions. A threshold ε that reflects the desired accuracy is defined, such that for a given particle, when the objective function applied to it has a fitness value less than ε, that particle is isolated from the rest of the swarm and its point undergoes a function stretching, so as to ensure the rest of the swarm are away from that particle's position. Furthermore, the swarm is randomly provided with a new particle that substitutes the pulled-away one. Then, if the isolated particle has a function value higher than the requested accuracy (ε), a niching technique is applied to create a new subswarm, as well as the algorithm is executed in a new instance, although it is conditional to that search region. At the maximum number of iterations, when the number of global minimizers is unknown or reaches

a predefined one, the algorithm stops. Unfortunately, this stretching transformation, arguably also a technique to accelerate convergence, may create local minima missing in the original objective function. A few restarts of the algorithm until a global minimum may resolve this problem [139]. As a consequence, in order to further improve their method, Parsopoulous and Vrahatis [140] introduced a deflection technique that empowers the objective function with knowledge from previously detected minimizers in addition to a better repulsion technique that ensures repelling a particle away from any of the detected local optima.

4.3.2. Nbest technique

In 2002, a new ω-PSO based technique, neighborhood best or *nbest* PSO, was proposed by Brits et al. [139], showing a successful experiment when applied to solve systems of unconstrained equations. One fitness function can express the transformation of a system of k equations as

$$f(\mathbf{x}) = \sum_{i=1}^{k} |f_i(\mathbf{x})|,$$

where each equation is equalized to zero by algebraically rewriting it; however, using this transformation to formulate the problem fails when the search space has multiple solutions. In order to tackle this problem, the objective function was reformulated as the minimum of the objective function between each pair of equations; that is, as in the example given by the authors in [139], when three equations (A, B, and C) are present in a system of equations, the combination with the minimum fitness among those equations is defined as the objective function:

$$f(\mathbf{x}) = \min\left(\left\{f_{AB}(\mathbf{x}), f_{AC}(\mathbf{x}), f_{BC}(\mathbf{x})\right\}\right).$$

Thus, the particles close to one of the solutions further away from the global best particle are rewarded, although not introducing any penalization. A dynamic neighborhood approach that builds on the Euclidean distance between particles, is used in the *nbest* technique (discussed in Section 3.5.1), in order to bias to a single optimum. As is well-known, it is computationally intensive to calculate the Euclidean distance, and depending on it, to choose the neighborhood may lead to divergent behavior. As a consequence, Euclidean neighborhood was abandoned later, wherein the Euclidean distance from every individual to all others is calculated, thereby forming the neighborhood for each particle while keeping the centre of the mass of the position as *nbest*. Meanwhile, PSO continues to perform normally until meeting a termination condition. The findings concluded by Brits et al. [139] show that all the globally best solutions can be found by the *nbest* technique. However, the systems of equations to be optimized do not always compose three equations, especially in real-world applications, and frequently their count is much larger. Thus, this method may cause performance degraded in such cases as the number of combinations may arise fast.

4.4. Multi-objective Optimization Problems (MOPs)

To begin with, only the optimization of one function was considered in most research on PSO. However, having only a single objective to optimize is rare in real-world problems, instead, multiple objectives are optimized simultaneously. At first glance, the algorithm can be independently run for different functions to optimize each of them, but the objectives may conflict with each other, and thus optimal solutions are seldom found. Modelling Multi-objective Optimization Problems (MOPs) is generally attributed to the following definition:

$$\min/\max\left(\left\{f(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\right)^T\right\}\right), \mathbf{x} \in X \subseteq \mathbb{R}^D,$$

where k is the number of objective(s) to be optimized.

In MOPs, a single solution that simultaneously optimizes each objective cannot be found, but instead so-called Pareto optimal solutions $\left(\mathbf{g}^*_{best}\right)$, a set of feasible solutions, are used. In other words, a feasible vector $\mathbf{x}$ would penalize at least one objective value, in order to optimize some other objective values. The so-called Pareto front is formed by this set of feasible solutions. To this point, it is the responsibility of the user/researcher to choose the best solution from the Pareto front to the problem at hand. Thus, Pareto dominance, known as a notion of dominance, was introduced: a vector $\mathbf{u} = (u_1, u_2, \ldots, u_k)$ is said to dominate the vector $\mathbf{z} = (z_1, z_2, \ldots, z_k)$ if $\forall i \in \{1, 2, \ldots, k\}, u_i \leq z_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < z_i$. Over the past two decades, PSO has also been paid attention by MOP optimization researchers. A few significant approaches to adapt PSO for multi-objective optimization are in order.

Hu and Eberhart [141] incorporated the concept of Pareto optimality, proposing a PSO variant for solving MOPs. A dynamic neighborhood PSO was presented; that is, each particle, at every iteration, has a different neighborhood than the prior, based on the distances from its position to the other particles' in the first objective function's fitness space. The fitness value of the second objective function determines the *lbest* particle of each particle within its neighborhood. Upon finding a new solution $\mathbf{p}^{t+1}_{best_i}$ which dominates the current solution $\mathbf{p}^{t}_{best_i}$, the new $\mathbf{p}^{t+1}_{best_i}$ is recognized. Unfortunately, only two objective functions were used in [141], as well as no enough details were provided on the implementation of the algorithm, especially how the distance between particles is computed. As well, the proposed strategy, in essence, only optimizes one objective function, with no guarantee on providing an optimal solution for the two objective functions together.

On the other hand, Coello et al. [142] proposed a Multi-Objective PSO (MOPSO), incorporating the notion of the external repository, in which non-dominated vectors composed of particles' positions are stored to be used later, in order to compute, at each iteration, each particle's velocity (replacing $\mathbf{g}_{best}$ in Eq. (8)). At each iteration, this repository is dynamically chosen by storing the new solution found that should be dominated over the external swarm. In addition, in order to prevent converging to a local optimum prematurely, a constrained optimization method as well as a mutation operator were incorporated to solve multi-objective COPs using PSO and to ensure that the swarm will maintain diversity, respectively. In the proposed constraint handling mechanism, if a particle violates the bounds, it is either inhibited within its respective bounds, or forced to search in the opposite direction by multiplying its velocity by -1. As for the mutation operator, for each particle, it is applied by changing the value of only one randomly chosen dimension, based on the current and total number of iterations, yet considering its bounds. The algorithm then performs like ω-PSO until a termination condition is met. The algorithm outputs a Pareto front which is being built as a grid, at the end of each iteration, using the external repository's values. Better results were revealed by the MOPSO approach over other multi-objective EAs with low computational time. Those were the first research steps on solving MOPs using PSO. Later, Mostaghim [143] improved the MOPSO algorithm.

Recently, Zhang et al. [144] proposed a multi-objective particle swarm optimizer based on a competitive mechanism, wherein the current swarm at each iteration performs pairwise competitions which in turn are used to update the particles. The experiments revealed that the algorithm has well performed in terms of both optimization quality and convergence speed. However, the global and personal best particles are not saved in an external archive, as well as the competition in each

swarm is among the only currently selected elite particles. Adhikari et al. [145] developed a novel Container-based Energy-Efficient Scheduling method (CEES) which handles different types of quickly successive Internet of Things (IoT) and non-IoT based tasks. A Multi-Objective Accelerated PSO technique (MOAPSO) is used in the proposed method for assuring a minimum delay by finding an appropriate container for each task. The proposed algorithm achieves superior performance over the existing ones in terms of low CO_2 emission and temperature, low computation time, and better CPU and memory utilization.

4.5. Discrete hyperspace optimization

Continuous meta-heuristics cannot handle a variety of optimization problems. Therefore, combinatorial optimization [146] was suggested to tackle such problems. DPSO, a discrete version of PSO, has shown interesting results in solving combinatorial optimization problems [147]. In DPSO, after completing each iteration or when the optimization process ends, a clamping criterion is used to round off the discrete variables to their nearest values that may offer significant speedup; however, choosing round-offs unintelligently may render worse fitness values due to the resulting movement of the particle towards a comparably infeasible region. Another widely used binarization approach amongst different transfer functions is the sigmoid function that is applied at the end of each iteration, so as to map the updated velocity into the closed interval [0,1]. The updated velocity indicates that the updated position has the binary value 1, since the sigmoid function is then more likely to output 1 when there is a high enough velocity. To ensure a potential reversal of the sigmoid output value, the maximum velocity value is often clamped to a low value. Afshinmanesh et al. [148] used XOR and OR operations in Boolean algebra to modify flight equations, which inspired a velocity bounding constraint by negative selection mechanisms in immune systems. In turn, in order to render better exploration capabilities to the Binary PSO, Deligkaris et al. [149] used a mutation operator on particle velocities.

Chen et al. [150] characterized the discrete search space of combinatorial optimization problems using a set representation approach, in which a crisp set is used to represent the position (solution) while a set of possibilities are used to represent the velocity. Operators defined on crisp sets and sets of possibilities are used instead of the conventional operators in Eq. (8), thus releasing a structure that is applicable to a discrete search space while remaining similar to the ω-PSO in its behavior. Two well-known discrete optimization problems, the Multidimensional Knapsack Problem (MKP) and Traveling Salesman Problem (TSP), were tested, demonstrating the promising nature of the discrete version. Nema et al. [151] solved the Mixed Discrete Nonlinear Programming problem (MDNLP) by hybridizing PSO with the deterministic Branch and Bound algorithm. The computational effort required in nonlinear programming problems is reduced by linking the PSO's global search capability to the Branch and Bound algorithm's fast convergence rate.

Chowdhury et al. [152] noted the stagnation problem when using PSO, especially in constrained single objective problems and pointed out the pronounced effect of introducing a mixture of continuous and discrete design variables in objective functions. In [152], the authors presented a modification to the ω-PSO, which decides the primary stage as continuous optimization while the subsequent stage as a criterion for nearest vertex approximation to update the discrete variables. Moreover, the proposal of a diversity preservation strategy produced a stochastic update in the case of discrete variables and a dynamic repulsion towards the global best in the case of continuous ones. A set of 98 problems regarding Mixed Integer Nonlinear Programming (MINLP) as well as a set of nine unconstrained problems were tested to successfully validate the the performance of the proposed approach.

More recently, Gong et al. [153] attempted social networks' influence maximization and Aminbakhsh and Sonmez [154] presented a DPSO strategy for solving large-scale Discrete Time-Cost Trade-off Problem (DTCTP), in which empirical experiments provided high-quality, within-seconds solutions for time-cost optimization of large-size projects, thus enabling optimal planning of real-time projects. Xu et al. [155] developed a new chaotic PSO (CS-PSO), which enhanced the classical PSO algorithm thanks to the chaotic method for solving combinatorial optimization problems. In CS-PSO, the initial particles are optimized by utilizing the prior knowledge of the combinatorial optimization problems, particularly in the initialization phase. Furthermore, in the chaos perturbing phase, the positions and velocities of particles are perturbed by introducing a brand-new range of rules for satisfying the adaptability and the capability of the ideal global search with the major aim of effectively avoiding the early convergence problem that haunts the ω-PSO algorithm.

4.6. Surrogate-assisted optimization for expensive problems

Surrogate-assisted optimization was established to handle expensive and complex problems arising from real-world applications. The methodology of such type of optimization depends on the optimal exploitation of the available information with the aim of minimizing the number of expensive evaluations required for a function, thereby reducing the related costs, resources, and time [156]. The use of PSO for surrogate-assisted optimization of expensive problems has been recently underlined by some research efforts. The success of surrogate-assisted swarm algorithms depends on the integration of the surrogates, also known as meta-models, as well as the efficiency of the underlying optimizer for approximating the objective function. PSO with its competitive performance can be a good core optimizer in such cases. Over the past few years, the literature has reported various surrogate-assisted swarm algorithms. In general, based on the number of surrogates used, two categories can represent existing frameworks: single-surrogate-assisted swarm algorithms and multi-surrogate-assisted ones.

Jin et al. [157] made a global surrogate model based on a neural network to help create a covariance matrix adaptation strategy. In addition, they evaluated the management strategies of individual-based and generation-based models. Praveen et al. [158] employed the Radial-Basis Function (RBF) model to develop a new global surrogate-assisted PSO algorithm for prescreening promising solutions, thereby saving the computational resource. Tang et al. [159] proposed a hybrid surrogate model formed by combining an RBF model built by interpolating the residual errors, with a low-order polynomial regression model. Inspired by the idea of active learning, Wang et al. [160] proposed a surrogate-assisted PSO using an ensemble surrogate-based model management method, in order to pursue the promising candidate solutions to be evaluated by an expensive objective function.

Yu et al. [161] proposed a Surrogate-assisted Hierarchical Particle Swarm Optimizer (SHPSO) consisting of a social learning-based PSO (SL-PSO), along with the ω-PSO algorithm for the goal of solving high-dimensional problems. The cooperation between SL-PSO and ω-PSO is proposed such that the search space is exploited and explored comprehensively, and the surrogate model provides information that simultaneously improves both global and local search performances. SHPSO achieved a significant improvement on 30, 50, and 100-dimensional cases. In turn, Sun et al. [162], pointed out that this cooperation can be used in such a different way that the promising solutions evaluated by the real objective function are shared among the individuals, having, at the same time, the two capabilities that the SL-PSO focuses on exploration while ω-PSO concentrates on local search.

In most existing multiple-surrogate-assisted PSO algorithms, the local surrogate model typically targets to capture the local details of the objective function around the current individuals' neighborhood, while the global ones aim to smoothen out the local optima. However, performing a local search around each individual may not help efficiently in finding the global optimum, and finding the search space's optimum currently covered by the swarm might be more helpful.

5. Variants of PSO

5.1. Adaptive PSO

An adaptive PSO approach was first ever suggested by Hu et al. [163], in which various changes in a dynamic system are automatically tracked by introducing re-randomization to respond to dynamic changes. Later in, Xie et al. [164] presented an adaptive PSO on an individual level using a replacement criterion, based on the diversity between the fitness of the current particle and the historical best fitness, to adaptively maintain the swarm social attribution by taking off inactive particles.

Zhan et al. [165] proposed one important approach for improving the convergence and search efficacy of PSO, as well as solving both unimodal and multimodal functions. According to the evaluation of each particle's fitness and swarm diversity (or distribution), four evolutionary states for PSO are defined in the Adaptive PSO (APSO) presented by those authors: exploration, exploitation, convergence, and jumping-out, for which different strategies, such as parameter adaptation, can be applied. To assess the swarm diversity, each particle's mean distance to all other particles is calculated as Euclidean metric as

$$D_i^t = \frac{1}{N-1}\sum_{k=1,i\neq k}^{N}\sqrt{\sum_{j=1}^{D}\left(x_{i,j}^t - x_{k,j}^t\right)^2}.$$

Then, e_f, an evolutionary factor, is given by

$$e_f = \frac{D_g - D_{\min}}{D_{\max} - D_{\min}} \in [0,1],$$

where D_g is the D_i^t's value of the globally best particle, and $D_{\min}$ and $D_{\max}$ are respectively the minimum and maximum distances among the particles. Based on this factor, one of the evolutionary states is then chosen to express the algorithm's current status. For example, the exploration state is indicated by a medium to substantial value of e_f, while the exploitation is indicated by a shrunk value of e_f. In turn, at a minimum value of e_f, the convergence state happens, and when the best particle's mean distance value significantly exceeds the other particles' mean distance value, the "jumping out" of local optima takes place. On some unimodal and multimodal functions, APSO demonstrated its ability to improve the convergence behavior, and most importantly, to enhance the algorithm's accuracy, compared with other well-known approaches.

5.1.1. Self-adaptive PSO

Recently, Isiet and Gadala [166] proposed a new adaptive method known as the Unique Adaptive PSO (UAPSO), to adapt the control parameters to the particles' circumstances at a specific moment. In UAPSO, particles are forced to learn only from their previous feasible solutions. Therefore, a constraint handling technique was proposed to work in accord with the adapting scheme to ensure the adaptiveness of the particles to the environment by only directing to the feasible regions. The personal fitness of each particle as well as the global fitness are used in UAPSO to produce a unique $c_{1_i}^t$, $c_{2_i}^t$ and ω_i^t for every particle i at each iteration t, based on a feedback parameter as a mechanism to self-adapt the particles. In addition to a real-world design problem, eight benchmark COPs were involved in a comparative study to verify the performance of UAPSO. Compared with PSO variants and other state-of-the-art metaheuristics, UAPSO has numerically shown consistency and efficiency, as well as superior ability to avoid premature convergence.

While proposing many such SAPSO techniques, their adaptation mechanisms as well as their searching behavior are not well understood due to the lack of in-depth critical analysis. Therefore, and in order to deepen the understanding of the convergent behavior of the SAPSO algorithms in the literature, Harrison et al. [167] investigated analytically and empirically 18 SAPSO algorithms, as well as empirically examined the stability of the adapted parameters and the likehood that a well-known convergence criterion can be adhered to by the tuned parameter values. According to the experimental results, a grim state of the SAPSO algorithms was depicted; that is, divergent behavior was exhibited by over half of the SAPSO algorithms while many others prematurely converge. Evidence clearly indicates that a sad state hits the field of self-adaptive PSO algorithms, given that many techniques either show rapid convergence, which prohibitively causes low particle's step sizes, or demonstrate divergent behavior with too much invalid particles, and thus infeasible solutions.

5.1.2. Fuzzy-adaptive PSO

In early 2000's, Shi and Eberhart [168] implemented a fuzzy inference system to dynamically adapt the IW of the ω-PSO. Experimental results on three benchmark functions with different dimensions illustrated that the Fuzzy APSO (FAPSO) is a promising optimization method, especially in dynamic environments. Motivated by that adaption method, Bajpai and Singh [169] used the FAPSO algorithm to solve the bidding strategy of a thermal generator for each trading period in a uniform price spot market to supply in each trading interval by following the frequently changing market demand. Nobile et al. [170] proposed a new algorithm called fuzzy self-tuning PSO (FST-PSO), wherein the minimum and maximum velocity of each particle, the social and cognitive factors, and the IW are independently calculated by using FL, thus creating a completely setting-free variant of PSO. The strength and novelty of FST-PSO is derived from the fact that it requires no experience to formulate PSO, since the behavior of each particle is automatically and dynamically adjusted during the optimization process. However, one remaining challenge is to implement FST-PSO for large-scale optimization.

5.2. Fully informed PSO

A Fully Informed Particle Swarm (FIPS) was proposed by Mendes et al. [171], in which the information of all particles in the neighborhood are used to guide the search, instead of using only the information of the *gbest* particle in the neighborhood (i.e., the particles are fully informed). To this end, the authors considered five different topologies, including *gbest*, ring, four clusters, pyramid, and square, and proposed a new velocity update equation, integrating the constriction factor approach of PSO and deprecating the social component, as

$$\mathbf{v}_i^{t+1} = \chi\left(\mathbf{v}_i^t + c\left(\mathbf{p}_{best_i}^t + \mathbf{x}_i^t\right)\right).$$

Typically, $\chi \approx 0.7298$ and $c = 4.1$. The particle's personal best position $p_{best_i}^t$ is given by

$$\mathbf{p}_{best_i}^t = \frac{\sum_{i=1}^{N}\sigma(i)\left(\mathbf{c}_i \otimes \mathbf{p}_{best_i}^t\right)}{\sum_{i=1}^{N}\sigma(i)\mathbf{c}_i},$$

with

$$\mathbf{c}_i \sim U\left(0, \frac{c_{\max}}{N}\right)^D.$$

Through the iterations, a constant value is returned by the function $\sigma(i)$, or as in the experiments by Mendes et al. [171], the distance from the i-th particle's best position found so far to the current particle or the best fitness of the i-th particle. Indeed, the later adjustment for $\sigma(i)$ ([171]) can be seen as a weighted FIPS version, wherein the particles' contributions are assessed according the distance to the target particle or the previous best fitness value. In fact, the authors are right, since, for all considered neighborhood architectures (apart from *gbest*), both FIPS variants performed well, always getting the minimum of the benchmark functions. However, an extra computational cost is required by the weighted versions, which may not be justified, since, in their study, a quite good performance was exhibited by the unweighted version.

Similarly, Mansour et al. [172] also used FIPS as a generalization of standard PSO and applied different neighborhood approaches, including *gbest* topology, distance topology, and ring topology. The study evaluated performance of the FIPS-based clustering algorithm using eight different datasets. According to the results, the study concluded that the

distance topology performed better than the canonical PSO-based clustering method.

5.3. Ensemble PSO

When averaged over a set of objective functions, superior results using a single optimization algorithm cannot be obtained, according to the "No Free Lunch" theorem [173]. Instead, for a given optimization problem, different degrees of effectiveness should be reflected by different algorithms. Here, given an objective function, ensembles of optimizers have been put together by researchers to obtain a set of candidate solutions and choose from the promising ones. Multi-strategy ensemble PSO (MEPSO [174]), one of such existing ensemble approaches, uses a two-stage approach: Gaussian local search and differential mutation, to respectively increase the diversity of the particles as well as improve the convergence capability. Furthermore, the comprehensive learning PSO (CLPSO [175,176]) was proposed, and Heterogeneouss CLPSO (HCLPSO [177]) was developed as well, in order to enhance the exploitation and exploration processes of CLPSO by using all other particles' *pbest* information to update a particle's velocity. These strategies help discourage premature convergence while preserving the diversity. The ω-PSO is simple and easy to develop, but some potential issues, such as the phenomena of "two steps forward, one step back" and "oscillation", may disrupt its performance. Therefore, it has been an urgent task for PSO researchers to design such an efficacious learning strategy that can be utilized, in order to avoid the above-mentioned two phenomena as well as improve the search efficiency by jumping out from the local optimal solution.

A pool of different PSO search behaviors emerged in the Heterogeneous PSO (HPSO [178]) which empirically outperformed the peer homogeneous PSO. Similarly, a pool of PSO strategies were also applied in the ensemble PSO proposed by Lynn and Suganthan [179] to gradually, through a merit-based scheme, choose a suitable one, so as to guide the movement of particles in a particular iteration. In [74], a particle swarm optimizer with an ensemble of IWs was proposed and incorporated into HCLPSO for testing its effectiveness. Different strategies of IWs, including linear, logarithmic, exponential decreasing, Gompertz, chaotic, and oscillating were considered in [74] to demonstrate the suitability of the proposed method by comparing it, on a large set of multi-dimensional benchmark problems, against other strategies.

Recently, in [180], the authors came up with a HCLPSO-based method to create a 3D spatial trajectory tracker, thereby realizing a new saturated approach to control a quadrotor. First, the quadrotor model is divided into an inner position control loop inside a Cascaded Control Structure (CCS) as well as an outer attitude control loop. Second, saturated control is applied to limit the quadrotor's thrust force in the outer attitude control loop. At the end, the quadrotor parameter adjustment difficulty is alleviated by employing an HCLPSO algorithm to optimize the control parameters. In the spatial trajectory tracking control of the quadrotor, the HCLPSO verified its ability to achieve better optimization performance than the PSO, CLPSO, GA, and DE. Nevertheless, it is a good idea to combine the robust control method with the saturated control. Moreover, novel Chaotic HCLPSO variants were introduced [181], where HCLPSO is combined with ten different chaos maps to identify the parameters of several static and dynamic photovoltaic models based on different experimental datasets.

5.4. Bare-bones PSO

Kennedy [182] observed that, in a particle swarm where personal and global best positions are kept constant and $c_1 = c_2$, multiple position updates would yield a distribution of positions with a mean halfway between the personal and global best positions. Moreover, the particles in PSO have shown to converge on a theoretical attraction point $\frac{1}{2}(\mathbf{p}_{best} + \mathbf{g}_{best})$, which is the position halfway between their personal and global best positions, given $c_1 = c_2$ [183]. Subsequently, Kennedy [182] proposed the Bare-Bones PSO (BBPSO). The BBPSO does not use velocities to update particle positions. Instead, at each iteration, each particle's new position is sampled directly from a Gaussian distribution around the particle's theoretical attraction point, where the deviation of the distribution depends on the absolute value of the distance between the particle's $\mathbf{p}_{best}$ and $\mathbf{g}_{best}$ positions. For each particle's component j, the position vector is updated according to

$$x_{i,j}^{t+1} \sim N\left(\frac{1}{2}\left(p_{best_i,j}^{t} + g_{best,j}^{t}\right), \sigma_{i,j}^{t}\right),$$

where the deviation $\left(\sigma_{i,j}^{t}\right)$ of the sample is given by $\left|p_{best_i,j}^{t} - g_{best,j}^{t}\right|$.

Kennedy [182] also proposed a modified version of the BBPSO. In the Modified Bare-Bones PSO (MBBPSO), the updated positions of particles are recombined with their personal best positions. Position component updates are given by

$$x_{i,j}^{t+1} = \begin{cases} p_{best_i,j}^{t} & \text{if } U(0,1) < 0.5, \\ N\left(\frac{1}{2}\left(p_{best_i,j}^{t} + g_{best,j}^{t}\right), \sigma_{i,j}^{t}\right) & \text{otherwise.} \end{cases}$$

In [182], it was concluded that particles in the MBBPSO are more cognitively-oriented than in the BBPSO, resulting in a more explorative search behavior.

Recently, a novel BBPSO was presented by [184]. The idea is that, along appropriate directions, the prospect of generating new particles is maximized. Due to the lack of a priori knowledge of these alignment directions, the proposed method adaptively learns suitable alignments by employing a learning mechanism using cellular learning automata. For each particle in the presented method, multiple alignment strategies are developed. Moreover, these strategies are adjusted so that, during the searching process, the particles are guided towards the most promising directions based on cellular learning automata. Durán-Rosal et al. [185] proposed also novel approaches for time-series segmentation. The proposed methods include adapting the PSO algorithm to tackle this problem by introducing new advanced variants of PSO (e.g., BBPSO and its exploitation-based version (BePSO)). Furthermore, a new algorithm called Dynamic exploitation BBPSO (DBBePSO) was derived to emphasize the importance of updating the cognitive and social components of PSO through iterations based on values extracted from a Gaussian distribution. In this approach, a local search step is involved to further improve the algorithms' performance using two popular traditional segmentation algorithms (top-down and bottom-up). However, Weibull distribution should be considered, and posterior tasks (clustering or classification) should be applied to observe the noise reduction between the original and the approximated time series.

5.5. Rough PSO

The theory of rough sets is a paradigm of uncertainty and ambiguity. The philosophy of this approach is based on the fact that every object in the universe can have its own information. More specifically, the rough set theory is a mathematical tool utilized to extract knowledge from information obtained from uncertain data. In feature selection, rough set analysis is used because it requires only a basic data to be supplied without the need for any supplementary information. Moreover, the rough set is very suitable for exploring the qualitative and quantitative properties of data. Das et al. [186] used the hybrid roughPSO technique to group the pixels of a satellite image in its intensity space, treating this image segmentation problem as a clustering problem, in which a rough set is used to model each cluster and PSO is employed for tuning the threshold and the relative importance of the rough sets' upper and lower approximations. The proposed algorithm, despite the presence of many noisy spots, was effective in detecting the correct segments in the underlying image. In [187], roughPSO algorithm was used within databases that can take numeric attributes, giving satisfactory results in its first application in data mining.

Recently, Fan et al. [188] proposed another roughPSO algorithm for solving the problem of premature convergence as well as getting better

accuracy and precision in classification tasks. The roughPSO utilizes the rough set upper and lower approximation sets to get the membership values for each parameter. Then, these values are used to refine the position and velocity of each particle. In another work, [189] proposed a technique of feature selection based on the ω-PSO algorithm and rough set (PSORS-FS) to select features for detecting permission-based Android malwares.

5.6. Guaranteed convergence PSO

Van den Bergh and Engelbrecht [190] have shown that the ω-PSO does not guarantee convergence to a local optimum, noticing a phenomenon that affected all *gbest* topology-based variants of the ω-PSO developed until then; that is, if the personal position of a particle and the global best position are the same (i.e., $\mathbf{x}_i^t = \mathbf{p}_{best_i}^t = \mathbf{g}_{best}^t$), then the velocity (position update) of that particle in Eq. (8) will only depend on its inertia component. This means that the previous velocity and ω must be non-zero so the particle can budge. Otherwise, all particles will eventually stagnate, causing the premature convergence to a position that is not guaranteed to be a local or global optimum, but only the best solution the entire swarm has found so far.

In order to overcome this problem, the Guaranteed Convergence PSO (GCPSO) was proposed by Van den Bergh and Engelbrecht [104]. At each iteration t, the fittest particle $\mathbf{x}_\tau$ is determined such that $f\left(\mathbf{x}_\tau^t\right) = f\left(\mathbf{g}_{best}^t\right)$, and an alternative velocity equation is suggested to update the position $\left(\mathbf{x}_\tau\right)$ of particle τ:

$$\mathbf{v}_\tau^{t+1} = \omega \mathbf{v}_\tau^t + \left(\mathbf{g}_{best}^t - \mathbf{x}_\tau^t\right) + \rho^t\left(1 - 2\mathbf{r}_{2_\tau}^t\right),$$

where $\mathbf{r}_{2_\tau}^t$ is a random vector sampled from $U(0,1)^D$. The term $-\mathbf{x}_\tau^t$ resets the τ-th particle personal best position to the global best one. ρ^t is a function introduced to define the diameter of a bounding box around *pbest*, which is to be randomly pursued. Other particles' positions are updated using the basic velocity formula in Eq. (8). The ρ value is determined as follows: ρ is set to 1 in the initialization step. Thereafter, ρ is updated for each subsequent iteration $t+1$, according to

$$\rho^{t+1} = \begin{cases} 2\rho^t & \text{if } CS > s_c, \\ \frac{1}{2}\rho^t & \text{if } CF > f_c, \end{cases}$$

where CS denotes the consecutive successes generated as a result of a fitter *pbest*, CF denotes the consecutive failures generated as a result of a non-fitter *pbest*, and the constants s_c and f_c are thresholds for successes and failures, respectively. CF is reset to zero whenever CS is incremented, and vice versa.

Although GCPSO algorithm has been used in several PSO variants [104,191], compared to the ω-PSO, it produces a highly poor solution when working on multimodal functions, due to the faster convergence towards a local optimum by the best particle. Peer et al. [192] have investigated that for the *lbest* (non-star/ring) model. Nevertheless, this scenario is out of question and most authors subsequently do not regard this approach to update the position of the best particle.

5.7. Quantum-behaved PSO

Forty years ago, a quantum computer was proposed, and in the late 1980s, the quantum computer formal definition was given. Due to the potential of the quantum computer in various special problems, this field has witnessed many great efforts in connection with SI algorithms. In 2004, [193] firstly introduced the Quantum-behaved PSO (QPSO) for improving the convergence rate of the ω-PSO. In QPSO, particles strive in quantum space for the global optimum via searching throughout the full solution space. Fang et al. [194] presented a quantum-inspired PSO with a cellular structured swarm, called cQPSO, in which the particles are distributed in a two-dimensional grid and restricted to communicate with their own neighbors locally. The experimental and statistical analysis results showed superior performance of the cQPSO with local best compared to other swarm-based algorithms. In [195], an optimal path planning was proposed for free-floating two-wheel pendulum robot system based on its self-balancing. First, the corner trajectory of this pendulum robot is parametrized by QPSO, and the native attitude and the control precision of the robot's terminal attitude and position are used to formulate the objective function. The addressed problem of optimal path planning is classified as a nonlinear optimization problem solved using the QPSO algorithm for the aim of optimal path planning. Nevertheless, the posture trajectory control energy can be further saved, and the time required for target docking and base pose re-stabilization can be shortened, too.

5.8. Parallelized implementation of PSO

Besides getting trapped into local optima, another problem with PSO is, with the increased dimensionality of the problem to be solved, the performance of the algorithm becomes progressively worse [114]. To alleviate this problem, it was suggested to distribute the processing among multiple processing units of a computer system, creating sub-swarms which significantly contribute to speeding up the algorithm's execution. Thinking about the independency of each sub-swarm, the parallel computing paradigm fits well into PSO. This section will describe some of the major methods concerning Parallelized PSO (PPSO).

For PPSO approaches, the tasks of each parallel sub-swarm can be processed using a Graphical Processing Unit (GPU) or even a multicore Central Processing Unit (CPU), potentially providing some mechanism for a synchronous or asynchronous exchange of information among them. When information is exchanged synchronously, the sub-swarms rely on each other (i.e., the particles in a sub-swarm do not move to the next iteration until the particles in other sub-swarms end), giving the same result as the serial execution, with parallel processing, however. On the other hand, asynchronous exchange enables the particles in a sub-swarm to move to the next position, based on the information provided by other parallel sub-swarms at the end of an iteration (especially the *gbest* information), irrespective of the current execution state (running or finished) of those sub-swarms. The exchange of information can be controlled using different architectures, some of which are: master-slave, in which one processing unit controls the execution of other units; fine-grained, where the sub-swarms of the big swarm are arranged in a 2D grid, allowing communication only between each sub-swarm's neighbors; and coarse-grained, in which the sub-swarms of the big swarm are independent of each other, but they, however, exchange particles among themselves, periodically [114].

Several instances of PSO were implemented based upon parallel computing platforms, to which literature has pointed as follows. In an early work where the first PPSO was implemented, Gies and Rahmat-Samii [196] used a system with parallel 10 nodes, which reported an eight-fold performance gain over a sequential node. Motivated by these results, Baskar and Suganthan [197] introduced a novel concurrent approach, called CONPSO, to improve the performance of fitness-distance-ratio PSO (FDR-PSO [198]). As stated by Schutte et al. [86], it is easy to produce synchronous implementations of PPSO. Nevertheless, a poor parallelization efficiency is usually yielded by such implementations, due to the potential idleness of some processing units.

While constraint handling was included in PPSO by Han et al. [199], a synchronous multi-swarm strategy for PPSO was proposed by Gülcü and Kodaz [200]. In this multi-swarm approach, a PSO variant is independently run by dividing the swarm into sub-swarms: one master-swarm and several slave-swarms. However, since the master-swarm is responsible for communication by migrating particles, the slave-swarms cannot directly communicate with each other. Cao et al. [201] upgraded this work to optimize multi-objective problems as well.

Along with all these developments, many methods using GPU rather than CPU were suggested, especially with the emergence of the CUDA development kit from NVIDIA. The use of GPU is far more complicated than using a CPU with regard to developing parallel algorithms. How-

ever, according to several studies (see; e.g., [202]), a GPU implementation of the PPSO has reported significant results with respect to execution time (with a speedup factor of 40 in some cases), compared with its corresponding implementation on a CPU. Li et al. [203] proposed a fine-grained PPSO for maintaining swarm diversity, keeping the utmost parallelism, and inhibiting premature solutions. Furthermore, Hung and Wang [204] proposed GPU-accelerated PSO (GPSO) for high-dimensional problems with a large number of particles by implementing a GPSO thread pool model on a GPU. Liao et al. [205] suggested a distributive PSO for solving a constrained luminance control problem. Finally, it should be noted that the distributed memory clusters [204] or a grid of multiple threads [206] can be utilized to develop new PSO distributed and load balancing versions.

6. Connections to other computational intelligence tools

Combining PSO with other intelligence tools comes in two forms: hybridization with (Sections 6.1 and 6.2) or application to (Section 6.3) others for prospective improvement.

6.1. PSO with Evolutionary Algorithms (EAs)

When combining the ω-PSO algorithm with other optimizers (e.g., GA operators, such as mutation, crossover, and selection), or other swarm-based algorithms), a PSO variant is produced and referred to as hybrid. The major aim of hybridization is to specifically increase the search quality of the swarm's individuals as well as to, in general, enhance both the algorithm's effectiveness and efficiency. The ω-PSO algorithm and some of its variants are known by their tendency to a potential local optima trap, which impedes exploring the whole search space. Therefore, the PSO algorithm is combined with EAs, in order to overcome this stumbling block against escaping from local optima as well as quell the inherent deficiencies of other algorithms it is hybridized with.

6.1.1. Evolutionary Computation (EC) operators

Angeline [207] produced what is considered the first hybrid PSO algorithm in 1998 by, similar to most traditional EAs, incorporating a selection mechanism into PSO. This mechanism works by scoring a point for the least fit among each particle's current fitness and other particles' current fitness values so that this score is used later to sort the swarm accordingly. After that, the best half's positions and velocities are used to replace the worst half's current positions and velocities, while maintaining the personal best position. Thus, the low-scored particles, thanks to the selection process, are reset, within the search space, to locations that have yielded better results. On most of the tested functions, introducing this truncation selection mechanism to PSO improved significantly the algorithm performance. In turn, Yang et al. [208] used also the roulette wheel selection operator, wherein it is more likely that the swarm's best particles are selected. Since then, many researchers, on the other hand, managed to combine PSO with different mutation strategies (e.g., Gaussian and Cauchy mutations [191,209], and crossover operators [210]), reporting highly significant performance. What motivated these upgrades was the fact that PSO cannot easily find optimal or near-optimal solutions for many COPs, such as multi-objective optimization and multimodel optimization.

Mutation is a genetic operator which changes the value of a particle's next position or the value of g_{best} from its current state, with a certain probability, in the hope of finding a better solution, while the swarm diversity is maintained. This operation prevents premature convergence to a local optimum as well as provides strong capabilities of exploration and exploitation to the swarm. In turn, crossover (reproduction/recombination/breeding) is a two-stage process that involves: i) combining two particles which are randomly chosen, at a given breeding probability, from the particles selected for breeding, and ii) performing a crossover operation, in which two new particles are generated based on their parents' characteristics, replacing the parents with those new particles. In this context, an arithmetic crossover operator was suggested by Løvbjerg et al. [210], to compute the position of each new child particle.

Although Evolutionary Computation (EC) operators usually slow down the algorithm's efficiency, better results may be produced by those operators, especially when put against multimodal functions and multi-objective optimization. Miranda and Fonseca [211], in 2002, merged the EC's concepts with PSO, producing a new approach, denoted Evolutionary PSO (EPSO). In their algorithm, the operations of mutation (on the cognitive, social, and IW parameters), replication (where the replacement of each particle takes place r times; usually $r = 1$), selection and crossover (before evaluation) are applied to enable the survival and propagation of the fittest particle as well as to generate diversity. For clustering purposes, Premlatha and Natarajan [212] considered embedded GA operators to create a discrete version of PSO, DPSO. Once the particles stagnate, the GA operator initiates the reproduction process. "DPSO with mutation-crossover" was the name of this hybrid version of PSO. More recently in 2013, Wang et al. [213] proposed the Diversity-enhanced PSO with Neighborhood Search (DNSPSO), incorporating both a crossover operator that plays to, with a given probability, recombine the particle's current position with each previous dimension where the particle has not roamed, and a new neighborhood search strategy that combines the *lbest* and *gbest* models.

6.1.2. PSO with Genetic Algorithms (GAs)

Similar to PSO, Genetic Algorithm (GA) has a swarm of potential candidate solutions, which leads to PSO combined with GA. In GA, there are mutated chromosomes in each element of the swarm, based on a certain probability, to improve the solution by maintaining a certain level of swarm diversity. Each iteration is referred to as a generation, and the natural selection process is reflected by the algorithm, in which the fittest individuals are chosen for producing the next better-than-before generation. The ω-PSO is known for its inability to effectively avoid falling into the trap of local optima while searching the lookup area. Thus, this weakness can be reduced by using the GA algorithm, along with its operators. However, GA presents a slower convergence behavior than PSO [208]. Motivated by these merits and demerits, the researchers managed to introduce new hybrid optimization algorithms by combining PSO with GA.

Hybridizing GA with PSO can be accomplished using such popular approaches as the sequential or parallel use of the two approaches or involving GA operators within the PSO framework. Robinson et al. [214] were the first to introduce a hybridized approach of PSO and GA, in order to optimize a profiled corrugated horn antenna. In their approach, PSO is either first used and then GA (PSO-GA), or GA followed by PSO (GA-PSO), using the results of one algorithm as input for the other. If one of the algorithms does show no further improvement with respective to its currently obtained solution, this algorithm toggles between PSO and GA. In [215], once a predefined number of iterations is reached, the first algorithm is terminated. In this approach, the particle pool in the second algorithm is populated with the best particles from the first one and random generations are used to fill the empty positions, which maintains the diversity of the swarm, similar to the case at the end of the first phase. On the other hand, exchanging the fittest particles between GA and PSO while running in parallel was also put forth in [215]. PSO combined with GA was also used in some other applications (e.g., recurrent network design [216]), in which crossover and mutation operations, as in GA, are used to create individuals in a new generation, running at the same time an instance of PSO. However, GA and PSO, both, unlike the above approach, share the same swarm. After the fitness values are computed at the end of each generation, the bottom 50% of the elements are discarded while the top half is marked for maturing. The PSO algorithm is concerned with handling the maturing technique that is utilized for further improving the best-performing elements, instead of direct reproduction of the next generation. A tour-

nament selection is then used to choose parents so that the next offspring can be produced by applying crossover and mutation.

Abdel-Kader [217] proposed a hybrid algorithm for *k*-means clustering and named it GAI-PSO. An initial kernel of solutions was first found, based on the exploration ability of the GA algorithm, so that *k*-means subsequently – in a local search – can use the cluster centroids contained in that kernel. For handling COPs, Garg [218] used a PSO with GA to serve for improving and updating the decision vectors, respectively. A mathematical model of the heliostat field was developed and optimized by Li et al. [219], based on a hybrid PSO-GA for the sake to forecast the highest Daily Energy Collection (DEC). According to the results obtained, DEC during the winter solstice, autumnal equinox, summer solstice, and spring equinox increased by $0.9 \times 10^5 MJ$, $1.2 \times 10^5 MJ$, $1.8 \times 10^5 MJ$, and $1.1 \times 10^5 MJ$, respectively. In [220], data mining problems were solved by developing a framework of GA and PSO, which stems from the PSO's potential to find out an optimal solution via collaborative interactions and the GA's potential to find a global solution.

6.1.3. PSO with Differential Evolution (DE)

Differential Evolution (DE) belongs to the class of EAs, and like PSO and other meta-heuristics, it optimizes a given problem by improving a candidate solution (called agent) among a swarm of potential candidates, through iterations. In addition, this method was designed to solve real-valued optimization problems, with no differentiable functions involved. It has the potential to perform a local search within the search space, while maintaining an adequate level of swarm diversity, with no guarantee, however, that an optimal solution is ever obtained. However, it cannot memorize the previous process, making the combination of DE and PSO a promising development. A set of real numbers are used to represent each agent which walks through the hyperplane until satisfying a stopping criterion. Using three different agents, mutation and crossover are applied by DE for generating a new parameter vector. When putting on an objective function, if that new vector is better, it replaces the current vector [221].

An early hybrid approach combining PSO with DE, namely, SDEA, was first proposed by Hendtlass [222]. In this simple approach, PSO runs conventionally, while DE is run intermittently to move the particles to more feasible regions. A set of four benchmark functions (i.e., *d*-dimensional 3 pothole function, timbo2 function, the six-hump camel back function, and the goldstein-price function), were adopted in the experiments, showing significant improvements in performance. However, more fitness evaluations were required in the new algorithm, and problems with computationally heavy objective functions can thus benefit from the component swarm-based algorithm. Two years later, the DEPSO algorithm was proposed by Zang and Xie [223]. This time, DE and PSO run alternately, based on the current iteration number. If the number of the current iteration is even, then DE runs; otherwise, PSO is executed (or the other way around). In addition, a bell-shaped crossover and mutation were introduced in DEPSO to increase the diversity in the swarm, by applying different operations at random instead of the simultaneous application of both changes (as DE originally does). On benchmark problems, DEPSO revealed high efficiency when compared to PSO or DE alone.

Other PSO-DE hybrid approaches have also been proposed. For example, Talbi and Batouche [224] developed the DEPSO algorithm with the objective to find the optimal transformation approach for a multimodal rigid-body image registration, which, by maximization of mutual information, superimposed two images. Hao et al. [225] used a partly PSO approach, partly DE approach, as approaches of selective update for the positions of particles and used a suite of benchmark problems to test these approaches, proving both their effectiveness and high performance. On the other hand, Epitropakis et al. [226], in a hybrid approach, merged each particle's personal experience with the swarm's social and cognitive experience.

Recently, DE was improved by developing a novel self-adaptive mutated DEPSO [227], which can significantly make use of the fast convergence capability of PSO strategy and the stronger global exploration capability of an improved DE mutation strategy. As a result, the diversity of the swarm is first managed well throughout the evolution process so that a higher convergence speed can be then obtained. The overall performance of these algorithms is better than using DE and PSO alone. However, the problem of premature convergence in PSO still casts a shadow.

6.1.4. PSO with Simulated Annealing (SA)

Simulated Annealing (SA), another meta-heuristic optimization algorithm, mimics the thermodynamic process of annealing, wherein the microstructure of a metallic material is altered through slow and controlled cooling, to improve its main properties of ductility, hardness, and strength. When a state of minimum energy is reached by the material, that process ends. SA, like other meta-heuristics, makes no assumption on the convexity, differentiability, or continuity of the problem's constraints and cost functions. However, unlike PSO, poor solutions are accepted, at a certain probability, by SA, in order to improve the search process by maintaining the diversity [228]. Due to the popular inability of PSO to escape from optimal local solutions, PSO-SA hybrid variants were proposed with the understanding that SA is known for escaping from those solutions by making uphill movements, avoiding in this way premature convergence as well. However, guaranteed convergence to the global minimum is not always ensured by that algorithm. Additionally, these hybrid algorithms may also have computational effectiveness and efficiency compromised.

Wang and Li [229], Zhao et al. [230], and Yang et al. [228] were the first to make studies on the hybridization of PSO and SA. Wang and Li [229] showed that, when SA is independently run on each particle after evaluating its fitness and the movement is changed accordingly, the rate of convergence may raise as well as the algorithm will be able to escape from local optima. This algorithm is referred to as SA-PSO. In turn, in the HPSO algorithm proposed by Zhao et al. [230], the PSO is first executed during the hybrid search process to provide SA later with the initial solution. Yang et al. [228] presented the PSOSA algorithm, wherein the PSO and SA algorithms run simultaneously; that is, each particle's personal best position undergoes a Gaussian mutation operation. If the new fitness value is worse than the prior, then the solution can remain acceptable based on a given probability; otherwise, it is replaced by that new value. This method performance was noted to be efficient on a set of common benchmark functions from the literature.

Sadati et al. [231] proposed a similar algorithm called PSO-B-SA, by which the Under-Voltage Load Shedding (UVLS) problem is tackled based on the sensitivity of the static voltage stability margin at the collapse point or the maximum loading point. IEEE 14 and 18 bus test systems were used to implement PSO-B-SA in the UVLS scheme, considering each load's both economic and technical aspects. Minimum runs are required to reach optimum solutions by PSO-B-SA, compared to the other competitors, thereby verifying its suitability for power system applications which require, within a finite time bound, an approximate solution. Both hybrid algorithms outperformed in terms of search behavior, computation speed, and overall performance, when compared to PSO or SA. Shieh et al. [232] also used the PSO algorithm with SA to determine the validity of a currently obtained worse-than-prior solution, using the Metropolis criterion [233]. Niknam et al. [234] managed to solve the Dynamic Optical Power Flow problem (DOPF) using the proposed PSO-SA approach, taking into consideration ramp-rate constraints, valve-point effects, and prohibited zones. The PSO-SA showed effectiveness on the IEEE 30-bus test system with respect to visiting all possible solutions to a DOPF problem with high nonlinearity and nonconvexity. Wang and Sun also solved the *k*-means clustering problem using a hybrid SA-PSO [235].

Recently, a hybrid PSO-SA was contributed by Javidrad and Nazari [236]; that is, through the iterations, PSO may repeatedly show no improvement regarding the global best particle. SA then intervenes to update that global best particle over and over, until a reject decision is

Table 4
Summary of PSO algorithms hybridized with EAs.

Methods hybridized	Author(s)	Area of application
PSO with EC operators	Angeline [207]	Truncation selection mechanism
	Løvbjerg et al. [210]	Unimodal and multimodal optimization
	Miranda and Fonseca [210]	Power system problems
	Premlatha and Natarajan [212]	Document clustering
	Wang et al. [213]	Rotated multimodal and shifted high-dimensional global optimization
	Jana et al. [209]	Reconstruction of gene regulatory network
PSO with GAs	Robinson et al. [214]	Optimizing a profiled corrugated horn antenna
	Shi et al. [215]	Numerical global optimization
	Juang [216]	Recurrent network design
	Abdel-Kader [217]	Data clustering
	Garg [218]	Constrained optimization
	Li et al. [219]	Optimization of a heliostat field layout
	Moslehi and Haeri [220]	Mining of quantitative association rules
PSO with DE	Hendtlass [222]	Unconstrained global optimization
	Zhang and Xie [222]	Unconstrained global optimization
	Talbi and Batouche [224]	Multimodal image registration
	Hao et al. [225]	Unconstrained global optimization
	Wang et al. [227]	Arrival flights scheduling
PSO with SA	Wang and Li [229]	Global optimization
	Zhao et al. [230]	Partner selection in virtual enterprise
	Yang et al. [228]	Global Optimization
	Sadati et al. [231]	Under-voltage load-shedding
	Shieh et al. [232]	Global optimization
	Niknam et al. [234]	Dynamic optical power flow
	Javidrad and Nazari [236]	Global optimization

Table 5
Summary of PSO algorithms hybridized with other social meta-heuristic algorithms.

Methods hybridized	Author(s)	Area of application
PSO with ABC	KıRan and GüNdüZ [237]	Continuous optimization
	Sedighizadeh and Mazaheripour [238]	Multi-objective vehicle routing with precedence constraints
	Shakti et al. [239]	Capacity optimization of grid-connected solar/fuel cells
PSO with ACO	Shelokar et al. [240]	Improved continuous optimization
	Kaveh and Talatahari [238]	Truss structures
	Xiong and Wang [241]	Hybrid clustering
	Liu et al. [242]	Container truck route optimization
PSO with FA	Arunachalam et al. [243]	Combined economic and emission dispatch
	Xia et al. [244]	Global Optimization
	Al-Thanoon et al. [245]	Classification with application in chemometrics
PSO with CS	Enireddy and Kumar [246]	Compressed image classification
	Dash et al. [247]	Linear phase multiband stop filters
	Jacob and Pradeep [248]	Multi-objective task scheduling in cloud environment
PSO with BA	Manoj et al. [249]	Medical image registration
	Zarei et al. [250]	Optimal water allocation among consumers

made by the Metropolis criterion [233]. PSO then initiates the search process again with the new global best obtained from that new information. This sharing process continues until a satisfactory convergence behavior is met. A brief listing of some important hybrid PSO algorithms using EAs are summarized in Table 4.

6.2. PSO with other social meta-heuristic algorithms

The PSO, or its variants, has been hybridized with other social optimization algorithms commonly used in diverse engineering applications. In the literature, too many PSO-based hybrid algorithms exist and use such common techniques as ABC, ACO, Firefly Algorithm (FA), Cuckoo Search (CS), and Bat Algorithm (BA). A few prominent examples of important hybrid PSO algorithms using other social meta-heuristic algorithms are provided in Table 5.

The last two decades have witnessed an exceptional impetus in the investigation horizons of PSO. However, it is worth noting here that only the hybrid PSO-based algorithms most practical or relevant to future research are enlightened in this section.

6.3. Synergy between PSO and machine learning techniques

6.3.1. Artificial Neural Networks (ANNs) with PSO

Artificial Neural Networks (ANNs) are competent for learning via training using a series of selected input and output vectors, reformed as a set of data to be trained to build a standard model for testing the remaining data [251]. In the two papers where PSO has been originally introduced [1,2], Eberhart and Kennedy made the first experiment on training ANN weights using PSO. They claimed to have successfully used PSO to train a feedforward multilayer perceptron ANN to classify the Fisher's Iris dataset and to solve the problem of exclusive OR (XOR), showing the same results as the backpropagation algorithm, and sometimes better. Note that the momentum term in a gradient descent ANN algorithm is similar to the IW in PSO [127].

Eberhart and Hu [252] adapted PSO to train a feedforward multilayer perceptron ANN using sigmoid activation functions for the classification of tremor types in Parkinson's disease. Although this dataset is small, the ANN has been successfully trained using PSO with high performance and low error. Janson et al. [253], in turn, showed that the product unit ANNs (where each node's output is calculated as a weighted product) can also be trained using the PSO, producing the lowest errors compared to other optimization algorithms, such as GA. For

voice-controlled robot systems, Chatterjee et al. [254] showed that the weights of a Takagi-Sugeno neuro fuzzy network can be trained using the PSO algorithm. Juang [216] combined PSO, GA, and the elite strategy concept to apply PSO to recurrent neural/fuzzy network training for obtaining the best network design.

A modified version of the PSO, called MDPSO, was applied by Ince et al. [255] to a feedforward multilayer perceptron ANN, to find the optimal weights and architecture for the classification of electrocardiogram signals. In order for short-term forecasting of load and wind power to happen, the mutation was also integrated into the PSO algorithm by Quan et al.[256] for training a feedforward ANN.

In the aforementioned approaches, ANNs are trained using the PSO algorithm and its variants. Training an artificial network using the backpropagation algorithm is neglected, since, when compared with PSO, it requires differentiable and gradient information, is highly prone to get trapped in local minima, and experiences slow convergence [257]. Furthermore, typically, it seems that fewer epochs are needed by PSO to get good results, unlike the backpropagation algorithm.

6.3.2. Support Vector Machine (SVM) with PSO

Support Vector Machine (SVM) is a classifier that comes under supervised machine learning algorithms. In this algorithm, the data is analyzed, and particular visible or hidden patterns are explored for the purpose of performing the two jobs of classification and regression. Earlier in the literature, Tang et al. [258] employed the ω-PSO algorithm, along with Least Squares SVM (LSSVM), to perform gene selection from DNA microarray data, demonstrating good and stable performance. Ghamisi and Benediktsson [259] introduced a feature selection methodology by hybridizing GA and PSO. This method was tested on the Indian Pines hyperspectral dataset as well as four road detection purposes. This method selected the most informative features within an acceptable processing time automatically and did not require the user to set the number of desired features beforehand. Recently, Hoang et al. [260] proposed a novel differential PSO-based SVM (DPSO-SVM) classifier to monitor the conditions of a surge arrester. In the proposed method, the parameters of SVM classifiers are optimized by investigating the DPSO-SVM technique, in order to give better results.

7. Theoretical analysis of PSO

Theoretical analysis of PSOs is crucial to understand their search behavior, to determine on which problems the algorithm with a prescribed configuration will perform successfully or will fail, and to find out the permissible range of control parameters. Fortunately, the framework of PSO lends itself easily to analytical treatments, without having to make assumptions. Despite the emergent and chaotic properties of PSO, we still can predict some aspects of its behavior. In this section, a few theoretical aspects of PSO as a function optimizer are provided, hoping to help guide PSO use.

7.1. The need for per-dimension stochasticity

In PSO, the vectors $\mathbf{r}_1, \mathbf{r}_2 \sim U(0,1)^D$ are the source of stochasticity. However, some researchers have opted instead to use "scalars" r_1 and r_2. Indeed, this is simply a poor idea that, with a little use of linear algebra, will get clear. To keep it simple, consider a scenario where to initialize the velocities to $\mathbf{0}$, and the initialized swarm reflects the personal best information $\mathbf{p}^0_{best_i}$ and global best information $\mathbf{g}^0_{best}$. A similar situation is discussed in a non-simplified manner here [261]. Considering a swarm size N, and a search space dimensionality D, if scalars r_1 and r_2 are used, after the first iteration, all new positions generated will be simply a "linear" combination of $\left(\mathbf{p}_{best_i} - \mathbf{x}_i\right)$ and $\left(\mathbf{g}_{best} - \mathbf{x}_i\right)$, and therefore, all initial positions are generated as well as all new positions will be derived, from $\text{span}(\mathcal{I}) \subseteq \mathbb{R}^n$ where $n = \min(\{N, D\})$ and $\mathcal{I} =< \mathbf{x}^0_0, \mathbf{x}^0_1, \dots, \mathbf{x}^0_N >$, thereby restricting the search forever within that span! Now we have two situations with two different issues: i) $N < D$, where the search is within a "subspace" of the whole search space $\mathbb{R}^N$, thereby having a part of the search space unreachable; and ii) $N \geq D$, in which N is the maximum subspace size of $\text{span}(\mathcal{I})$, with, however, no guarantee of such largeness, since in the initial set $\mathcal{I}$, we could get unlucky with the orthogonality degree, and thus we still only search a subspace. Even if $\text{span}(\mathcal{I}) = \mathbb{R}^D$ could be guaranteed, degrees of freedom are still possible to be lost, which also occurs with the original set from which new positions can be derived. Meanwhile, the scalars r_1 and r_2 cannot be used to recover such a loss of orthogonality. Using the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ simply helps in avoiding all the above issues, where each component is sampled independently.

7.2. Stability of particles

From a theory perspective, particle convergence yet is often misunderstood, although it is probably the most analyzed aspect of PSO behavior. Likely, very overloaded terminology is the cause of the confusion. Specifically, in a stochastic context, the word "convergence" is ambiguous. In the early works, for example, Trelea [183], on particle convergence of the ω-PSO, the stochastic components were considered constants. As a result, the criterion of [183] was provided to ensure that the convergence of a swarm $\mathbf{s}_t \in \mathbb{R}^n$ (sequence of particles) is exhibited only if $\lim_{t\to\infty} \mathbf{s}_t = \mathbf{s}$ for an existing $\mathbf{s} \in \mathbb{R}^n$. However, understanding of the actual PSO behavior requires taking the stochasticity into consideration, which raises the question what is convergence in a stochastic context? The simplest type of stochastic convergence is "in expectation" which is defined – based on $\mathbf{s}_t$'s expectation $\left(E\left[\mathbf{s}_t\right]\right)$ – as order-1 stability, in which $\lim_{t\to\infty} E\left[\mathbf{s}_t\right] = \mathbf{s}_E$, for an existing $\mathbf{s}_E \in \mathbb{R}^n$.

While the convergence in expectation is informative, the issue is still open as has been noted by Poli [262]; that is, the variance may be increased even when the expectation of a stochastic sequence is stable. To illustrate, consider a random sequence $\lambda_t \sim U(-t, t)$. Now, the expectation of λ_t is zero for every t, which implies that the sequence λ_t is order-1 stable. However, the variance of the sequence λ_t is increasing gradually, thus ensuring the non-particular stability of λ_t. It is for this reason both order-1 and order-2 stability are needed and can be defined – based on $\mathbf{s}_t$'s variance $\left(V\left[\mathbf{s}_t\right]\right)$ – as $\lim_{t\to\infty} V\left[\mathbf{s}_t\right] = \mathbf{s}_V$, for an existing $\mathbf{s}_V \in \mathbb{R}^n$. When $\mathbf{s}_V$ must equal zero, this is termed order-2* stability which cannot be guaranteed for PSO [263]. In the literature, a sequence of particles is said to be convergent if it has both order-1 and order-2 stability. However, order-1 and order-2 stability is different from traditional convergence, because order-1 and order-2 stable particles can, just with a fixed expectation and variance, still move. This can actually be marked as positive since, with the non-zero fixed point of the order-2 moment, the swarm can continue to search.

Order-1 and order-2 stability can be guaranteed by some criteria on control parameters. Some of the existing possibilities in the literature are exhibited in Fig. 7. In fact, the curved line segment, AGB, is the correct region as Poli and Bromhead [262] and Jiang [261] originally independently derived as

$$0 < c_1 + c_2 < \frac{24\left(1-\omega^2\right)}{7-5\omega}, |\omega| < 1.$$

Note that, if a particle violates this criterion, it is stigmatized as unstable. Cleghorn and Engelbrecht [3] have empirically verified the above criteria without the presence of assumptions. In addition, the same authors, in another work [264], have recently cloned those criteria in what might be described as the minimal necessary modeling assumptions. Thus, stability really matters as it tells where to find viable parameter configurations. Specifically, particle instability resulting from improper parameter configurations was shown to almost often degrade PSO performance, when compared to random search [265].

PSO variants are typically preferred by researchers or practitioners, while most theory only addresses the inertia PSO and constriction PSO. However, the stability criteria can be easily derived for all these variants, using the theorem in [264], which can be rewritten as

$$\mathbf{x}_i^{t+1} = \alpha \mathbf{x}_i^t + \beta \mathbf{x}_i^{t-1} + \boldsymbol{\gamma}_t,$$

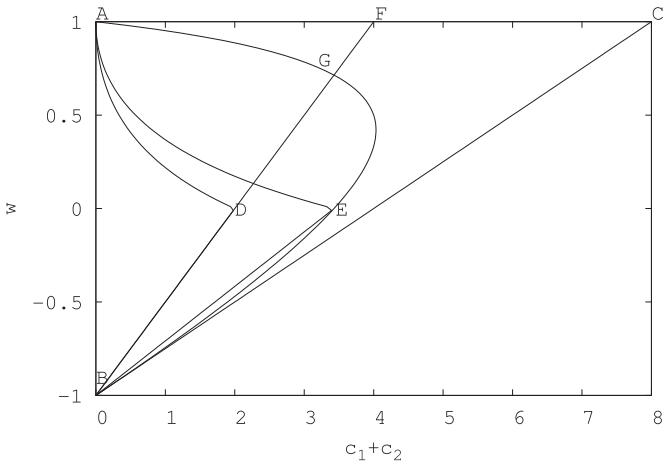


Fig. 7. Possibilities of control parameters to guarantee stability.

where $\boldsymbol{\gamma}_t$ is a sequence of random variables, and α and β are well-defined, independent random variables. Although this equation is simple, it caters for a variety of PSOs (e.g., FDR-PSO [198], unified PSO [266], and FIPS [267]), even with the use of any well-defined arbitrary distributions.

7.3. Roaming behavior of particles

Particle roaming in PSO is a well-known issue. If a particle is moving outside the feasible search regions, it is then said to be "roaming". Helwig and Wanka [268] have proved that the particles in the initial iteration will, with overwhelming probability, leave the search space when the velocities are initialized to $\mathbf{0}$ or uniformly initialized within $\left(\mathbf{x}_{\min}, \mathbf{x}_{\max}\right)^D$. The problem of roaming is not so severe in low-dimensional search spaces, as particles will inevitably return to the search space under the "let them fly" approach [46]. However, this problem is highly significant in high-dimensional search spaces and can be handled in two common approaches: i) *particle variance restriction*, in which we can predict the component-wise variance of the particle positions as

$$V\left[x_{i,j}^{t+1}\right] = \frac{c(5\omega+1)}{c(5\omega-7)-12\left(\omega^2-1\right)}\left(g_{best,j}^{t} - p_{best_i,j}^{t}\right)^2,\ c = c_1 = c_2,$$

such that the likelihood of a boundary violation decreases in case that the variance is restricted, and ii) *boundary constraint handling* many of the current boundary constraint handling approaches often interact poorly with the explosive dynamics of PSO (e.g., movement direction warping, bound bias (where the swarm majority is stuck on the bound), and constant swarm reinitialization). Nevertheless, a per-dimension hyperbolic boundary constraint handling is currently the best approach to alleviate the dimensionality curse. Further inclusive relevant approaches can be found here [14].

7.4. Particle movement patterns

The particle movement manners have been undertaken in some early research papers. However, they were derived in a deterministic context [58,183], which does give useful information – considering the particle trajectories "in expectation" – not enough, however. Nevertheless, the stochastic behavior of a particle can be characterized based on two aspects: *the range of motion* or the variance coefficient V_c, which is given as

$$V_c = \frac{c(5\omega+1)}{c(5\omega-7)-12\left(\omega^2-1\right)},$$

and *the base frequency* $F \in [0, 0.5]$, which refers to the largest amplitude of the Fourier series of the particle's positions throughout the search [269], in which particles' smooth trajectories are typically exhibited with a small value for F (that tends to 0) whereas large values for F (that tends to 0.5) implies that particles are prone to large steps between positions with more oscillations at each iteration. The control coefficients (i.e., $c = c_1 = c_2$ and ω) can be then easily derived for a given F and V_c. In fact, low- and high-dimensional search spaces give very different ideal movement patterns, as shown in Fig. 8(a), in which the optimal frequency-variance combinations are exhibited for 10- and 1000-dimensional CEC'10 large-scale problem suites, respectively in Figs. 8(a) and 8(b) (note that lighter/higher scores are better).

8. Diverse applications of PSO

Over the last two decades, parallel to the algorithmic research, the PSO researchers have also been spiking over the application-specific research pertaining to PSO. Each PSO application area is likely to render some problems that should be addressed to come up with prosperous solutions that will then make PSO more efficiently and influentially applicable in real environments. Among thousands of application papers in diverse areas, studies focusing on issues of major concern regarding the application of PSO are reviewed in this manuscript by individually addressing different contexts in each application area. To this end, we present a variety of PSO applications, in which special subjunctives have been addressed and discussed. Instead of going into detailed discussions, especially with the presence of a large volume of works published, only the major practices of PSO are highlighted in Tables 6 and 7.

9. Potential future research lines

While PSO has been effectively applied in diverse domains, some challenges and future research directions have possessed. In the past few years, PSO has gained increasing interest from researchers and has been used in various fields of applications, now reaching an impressive state. However, critical challenges and issues are continuously emerging for the algorithm and are yet to be catered for. In this regard, more efforts are required from scholars and researchers. Moreover, further inspirations and more effective techniques are required to develop novel PSO approaches. Based on the literature review, we unfold in Section 9.1 some pressing issues and related topics in the area of PSO that merit further work by the PSO community. In addition, several PSO methods and applications can be sophisticated in some areas as future research directions in this regard in what follows in Section 9.2.

9.1. Open issues

1. **Premature convergence:** In ω-PSO and its variants, when the search approaches the local optimal solution through random initial conditions, the optimal solution for a single particle (the individual particle optimum) and the optimal solution for all particles (the group optimum) might converge towards the local optimal solution and a deceptive optimization result will be obtained. Thus, it is not possible to guarantee the capability of exploring the global optimal solution, and the convergence ability will be ineffective. Over the last few years, the stagnation (premature convergence) issue was and remains a major challenge facing the PSO community [10,11]. A roadmap for future research in this regard can be drawn up based on some topics, including particles' stability analysis, redistributing mechanisms, and random sampling of control parameters.
2. **Convergence speed controller:** Despite the fast convergence of PSO, it may be trapped into a local optimum due to premature convergence. The convergence speed controller [306] is one among many solutions to that problem. Two adaptive approaches can be adopted to adjust the convergence speed. First, when the particle prematurely converges, the convergence speed is slowed down. Second, when the particle cannot update its best solution at the present time, the convergence speed of PSO is accelerated.

Table 6
Picked applications of PSO to diverse optimization problems.

Author(s)	Main context	Case study	Strengths	Shortcomings	New finding(s)
Environmental applications					
Camci et al. [26]	Quality inspection of rice farms by employing quadcopters	Rice farms in Longsheng, China	- High performance on handling noise and measurement uncertainties in the system	- Ineffective quadcopter control in natural landscapes	- Control structure
Ehteram et al. [270]	Improving the Muskingum flood routing method	Wilson, Karahan, and Viessman and Lewis floods in the USA and UK	- Low computational time - Approaching the best solution at a significant rate	- Conducting Muskingum models with little parameters	- Hybrid BA and PSO algorithm
Cao et al. [271]	Remote sensing to monitor water quality in inland waters based on HJ-1A HSI imagery	Inland waters in Weishan Lake	- Fast convergence - High stability - High prediction accuracy	- Not tested on different lakes - Using hyperspectral sensors with low spectral resolution	- Modified BPSO with catastrophe strategy
Kour and Arora [272]	Segmentation and classification of plants based on their leaf images using hybrid PSO-SVM	Agriculture environment	- High accuracy - Low computational time	- Not including plants that have medicinal and scientific significance	- Automatic vision-based method
Rahgoshay et al. [273]	Predicting daily suspended sediment load	Royan and Veynakeh earth dams in Semnan, Iran	- Reducing the computational time and RMSE - Increasing the accuracy	- Structure does not have ambiguous parameters	- Support vector method
Kumar et al. [274]	Short-term temperature prediction using ambient sensors	Environmental monitoring	- Improving the accuracy - Improving the generalization performance	- Lack of application in a realistic environment	- Hybrid model
Smart city applications					
Ramya et al. [27]	Retrieval of deprived riot video data	Smart city	- Identifying the previous criminal records in a particular region of the smart city - Can be used with other signal processing and image processing applications	- Low accuracy in processing large database efficiently	- Algorithm
Sato et al. [275]	Total optimization of energy networks in a smart city	Smart city	- Improving the solution quality	- Not verifying the robustness - Low scalability - Not evaluating monetary cost	- Hybrid algorithm
Hu et al. [276]	Scheduling urban traffic light	An area centering Xudong, Wuhan, China	- Controlling vehicle dynamics - Eliminating traffic congestion	- Not evaluating monetary cost - Not considering scalability - Not considering the case of self-organized cities -	- Hybrid algorithm
Ma et al. [277]	Appliances scheduling in a residential unit	Residential unit	- Good convergence performance of the algorithm - High profit	- Uncertainty of renewable generation	- An appliance scheduling model
Jordehi [278]	Scheduling shiftable appliances in smart homes	Smart home	- Decreasing consumers' daily electricity bill charge without affecting their comfort	- Not considering scalability	- Hybrid algorithm
Li et al. [279]	Forecasting Day-ahead traffic flow	Highways	- High accuracy - High stability	- The dataset used was built based upon a small section of a highway - High time-consuming	- Algorithm - Hybrid model
Zhang et al. [280]	GIS-based placement of charging stations for electric vehicles	Changping, Beijing, China	- Low monetary cost - High coverage	- Considering only two scenarios	- Algorithm - Optimal model
Abid et al. [281]	Managing energy in smart homes	Residential area of ten homes	- Low power consumption - Low cost	- Not evaluating accuracy - Not comparing the presented method with existing ones	- Algorithm - Energy management strategy

Table 7
Picked applications of PSO to diverse optimization problems (continued from Table 6).

Author(s)	Main context	Strengths	Shortcomings	New finding(s)
Health-care applications				
Srisukkham et al. [28]	Intelligent Leukaemia diagnosis	- Accelerated chaotic search - Escaping from the local optimum trap	- Not considering the particles' velocity besides only updating their positions	- Two modified BPSO algorithms
Raj and Ray [282]	ECG signal analysis	- High overall accuracy - High sensitivity - Positive predictivity	- High computational time - Not analyzing other classes of arrhythmia signals - Not performing real-time analysis	- Hybrid algorithm
Li et al. [283]	Medical image segmentation	- Improving the performance of dynamic context cooperative QPSO - Obtaining the most exploiting contextual information	- Limited type of image segmentation problems	- Algorithm
Jain et al. [284]	Cancer diagnosis and classification using DNA microarray technology	- Ensure faster and more reliable gene selection for classification process - Providing a solution to the local optimum problem of traditional BPSO	- Not considering the computational complexity	- Two-phase hybrid model
Zeng et al. [285]	Diagnosis of Alzheimer's disease	- Outperformance against several SVM algorithms and two other well-known learning methods	- Only one dataset, namely, ADNI, was used for testing	- Framework
Industrial applications				
Rahman and Zobaa [29]	Optimizing PMUs placement	- Reducing the number of PMUs needed for IEEE 300-bus system - Low computational cost	- Not considering scalability	- Algorithm - Objective function
Lopes et al. [286]	Distributing electrical loads throughout the day in an industrial context	- Better quality solutions - Minimizing the the cost of industry production	- Testing on small datasets - Not assessing computational time	- Hybrid algorithm
Qi et al. [287]	Predicting the unconfined compressive strength of cemented paste backfill	- High accuracy - Low response time - Low cost	- Ignoring some substantial variables - Not foreseeing the long-term strength	- Algorithm
Song et al. [288]	Positioning a 3D wind turbine with multiple hub heights on flat terrain	- Low cost - High power output	- Lack of assessment on complicated and realistic wind farm optimization problems	- Hybrid algorithm
Ghorbani et al. [289]	Optimal sizing of an off-grid house with photovoltaic panels, wind turbines, and battery	- Minimizing the total cost - Increasing reliability	- Not presenting overall accuracy evaluation	- Hybrid PV-WT generating unit - Hybrid algorithm
Liu et al. [290]	Multi-objective optimization design of the airborne electro-optical platform	- Reducing mechanical resonance - Improving stability - Reducing mass	- Considering multi-objective optimization problems with only three targets	- Hybrid algorithm
Alnaqi et al. [291]	Prediction of energetic performance of a building integrated photovoltaic/thermal system	- High reliability - High performance	- Not considering scalability	- Hybrid algorithm - Neural network model
Maiyar and Thakkar [292]	Food grain transportation problem	- Economic and environmental results - Reducing food grain wastages	- Not considering perishable food grain products	- Algorithm - Decision support tool
Commercial applications				
Pradeepkumar and Ravi [30]	Forecasting volatility from financial time series	- Yielding statistically significant results	- Not evaluating total computational time	- Neural network model
Yi et al. [293]	Cost prediction of transmission line project	- Improved accuracy - Strong practical significance	- Not evaluating scalability	- Intelligent cost prediction model
Shen and Han [294]	Profit calculation module of financial accounting information system	- Getting real-time financial processing results	- Not considering computational cost	- Accounting information system reconfiguration
Jiao et al. [295]	Optimal electric business centre location	- High transportation convenience - Low cost - Social benefits	- Not incorporating a multi-objective model	- Hybrid algorithm - A locating model

(*continued on next page*)

Table 7 (*continued*)

Author(s)	Main context	Strengths	Shortcomings	New finding(s)
Miscellaneous				
Sheikholeslami and Navimipour [296]	Service allocation in cloud computing	- High resource utilization - Fast resource allocation - High revenue for users and service providers alike	- Not evaluating customer satisfaction - Not implemented in an actual system - Not considering different weighting parameters	- Algorithm
Suresh and Lal [297]	Segmentation of satellite images based on multilevel thresholding	- High stability - Robust and fast algorithm	- High computational time - Low performance metrics values	- Algorithm
Alswaitti et al. [298]	Data clustering	- High classification accuracy - High cluster compactness	- Lack of real-world assessment	- Algorithm
Nouiri et al. [299]	Job shop scheduling problem	- Effectiveness in directing real production - Effective decentralizing decisions	- Not all entities participate in solving the problem - High energy consumption	- Architecture
Thabit and Mohades [300]	Path planning of multi-robots	- Short, safe, and smooth paths	- Not implementing real-world scenarios	- Algorithm
Sun et al. [301]	Locating attacks' position in WSNs	- Low attack localization time - Low energy consumption	- Not locating the source nodes of the attack	- Algorithm
Zhong et al. [302]	Travelling salesman problem	- High balance between intensification and diversification	- Not compared with numerous optimization techniques	- Algorithm
Lin et al. [303]	Set-union knapsack problem	- High-quality solutions	- High computational cost $\mathcal{O}((I_{\max} ND + N^2D + 2N) \times T)$, where $I_{\max}$ is a TS process iteration in one pass	- Hybrid algorithm
Mansouri et al. [31]	Task scheduling in cloud computing	- Low execution time - Low resource usage	- Not considering the precedence of tasks and load balancing - Not incorporating the fault tolerance parameters of cloud - Not testing in real environment	- Algorithm
Sivaranjani et al. [304]	Speckle noise removal in SAR images	- Optimized threshold values - Considering both reference and no-reference metrics for experimentation	- Failure to evaluate scalability	- Algorithm - Framework
Bhattacharya et al. [305]	Detection of permission-based Android malware	- High classification performance - High scalability	- Not handling float datasets - Considering only the permission set	- Hybrid algorithm

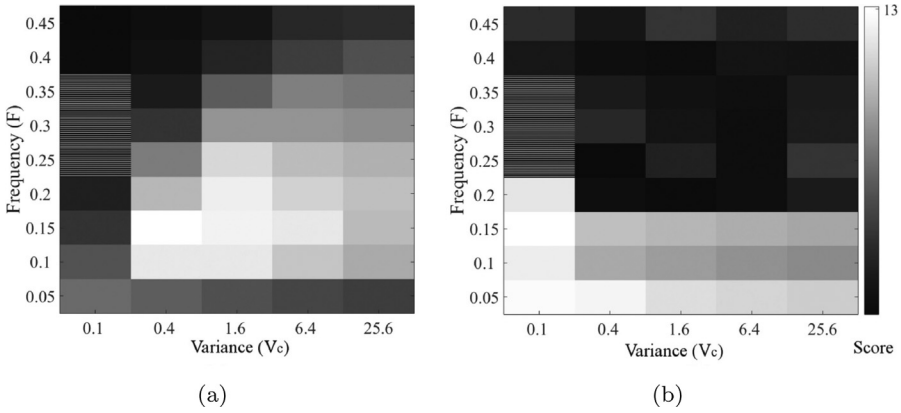


Fig. 8. The ideal movement patterns in (a) low- and (b) high-dimensional search spaces [14].

3. **High-dimensional search space:** Because of the curse of the dimensionality of optimization problems, the effectiveness of PSO applications in the classification of high-dimensional data and computationally expensive problems has recently gained a major concern. For example, the effectiveness of PSO approach for feature selection has been demonstrated. However, due to the large search space, it can still be difficult to apply to high-dimensional datasets that contain tens of thousands of features [307]. Thus, novel approaches need to be developed to simultaneously, yet efficiently minimize the number of selected features and maximize the classification accuracy.
4. **Subpar performance for multi-objective high-dimensional optimization:** Acceptable solutions for multimodal functions are rendered by niching techniques in both static and dynamic environments [308]. However, when the dimensionality of the problem increases, the solution quality falls sharply. PPSO and QPSO are two of the many potential effective solutions to this type of double complexity problems. To guide interested researchers, engineered crossover, mutation, and elite particles have the potential to add to the PSO reliability and efficiency.
5. **Parameter sensitivity:** Solution quality of PSO, like other metaheuristics, is sensitive to its parameter settings, which implies that one parameter selection strategy does not necessarily fit every problem. Thus, researchers are invited to discover further new methods to determine which parameter adaptation strategy obeys each category of optimization problems.
6. **Parameter & topology selection:** In PSO-based algorithms, the best performance can be achieved by elaborately choosing the parameters; however, choosing these parameters is not guided enough. To address the task of parameter selection, future efforts in this regard should involve choosing the best parameters based on simulations, parametric analysis in a computational framework with limited resources, as well as optimization-based hyper-parameter selection. On the other hand, the performance of PSO in engineering applications is affected by topology selection, and each problem has its own optimal topology. In fact, this issue still lacks much study. As a future direction, topology selection for PSO can be better guided by taking into account factors affecting the optimality of algorithmic parameters (i.e., the topological degree and the number of particles) with the aim of selecting a proper class of deterministic regular topologies [309].
7. **Swarm size adaptation:** Recently, swarm size adaptation has been demonstrated to boost the performance [83]. Naturally, a larger swarm is required at the early stage of the search to perform extensive exploration of the search space, while a smaller swarm is required at the end of the search process to conduct a fine search near the best region. Such a linear swarm reduction has shown benefits in works like L-SHADE [310]. Swarm size adaptation needs further research on different optimization scenarios, especially single objective ones.
8. **Fitness landscape analysis:** In the paradigm of SI, there is an apparent lack of how to determine the most appropriate optimization algorithm depending on the features of the problem (among several choices, including PSO, DE, GA, ABC, CMA-ES, etc.). This is quite true when tackling optimization problems in practice. A systematic study does matter, so as to determine how a set of decision variables should correlate or which feature(s) of a particular problem (among ruggedness, neutrality, multimodality of the fitness landscape, etc.) make PSO (or any of its variants) applicable to solve a given objective function. Unfortunately, such type of study has not been adequately conducted to guide choose among the versatile PSO variants available today based on problem features. Diverse properties possessed by real-world problems over the search space makes that challenging, exactly the same as in composition problems [311].
9. **Randomness caused by control parameter adaption:** Finally, while adapting the control parameters (i.e., c_1, c_2, ω, and constriction factor), a controlled amount of randomness is introduced in many variants. It is certainly interesting to find out novel appropriate methods to control when to increase/decrease the degree of randomization. Can we, somehow or other, link the amount of randomness with a type of correlation among the decision variables or with some features of the optimization function? This could be a potential future research avenue, too.
10. **Theoretical analysis:** Theoretical issues, such as the stagnation to local optima, loss of particle diversity, and the particle explosion problem [312] deserve further efforts from researchers.

Hence, numerous issues to be examined are highly posed in the particle swarm optimization context. Furthermore, the above issues can be tackled in different optimization schemes (e.g., large-scale, constrained, multimodal, multi-objective, dynamic, and discrete), as well as collectively.

9.2. *Future prospects*

- **Nuero Fuzzy Network (NFN):** NFN is an intelligent method for system identification/modelling, prediction, and control. In NFN, gradient-based algorithms are generally used for training. But these algorithms have some disadvantages of getting stuck at a local minima, as well as the need for complex gradient computations. Accordingly, [313] used improved PSO to introduce the first embedded high-speed, low-cost implementation of NFN hardware through online training. It is observed that that the effectiveness of the proposed NFN implementation is similar to other approaches in the literature, thereby generating a novel topic for future research.
- **Learning based approaches:** PSO needs further development on learning-based approaches. Even though PSO is tested on numerous optimization problems with varied characteristics, a single problem probably needs to be tackled with minor variations repeatedly over a long period of time [314]. In this regard, minor variants of the same problem can be solved repeatedly by PSO, based on learning-based approaches.

- **Image registration:** Image registration is defined as taking several 2D images from various sources, such as Computer-Assisted Tomography (CAT) and Magnetic Resonance Imaging (MRI) scans, and combining them as a 3D image. Recently, a hybrid approach for registering medical images has been developed by employing a PSO method [315] in addition to adjusting the mutual information as a similarity index. However, there are still trend applications for future research, including registering the images of a printed circuit board placed on a conveyor belt using an improved scale invariant feature transform/extraction technique combined with PSO. Moreover, using PSO algorithm for remote sensing image registration with a less impactful correction rate is a major trend for future research, too.
- **Computational biology:** A long DNA chain first needs to be divided into subset fragments for determining its sequence. Therefore, combinatorial optimization researchers have opted the DNA Fragment Assembly method (DFA) to solve this NP-hard problem of fragment's reassembling. The DFA problem has been tried by applying the overlap-layout-consensus model to maximize the overlapping score measurement using a memetic PSO algorithm based on two initialization operators as well as the local search operator [316]. Future works could address interesting issues, such as reducing the computational time by using DNA sequence compression, improving the initialization method by developing other meta-heuristic algorithms, and using alternative search methods.
- **Recommender systems:** PSO, as a tuning mechanism, has been applied in a novel area, in which software tools are created to develop recommendations to entrepreneurs and even end users. Explicit feedback data (i.e., votes or ratings) are usually used to build most of the current recommender models. However, a real-life scenario does not always contain explicit feedback data. For instance, a hybrid music recommender system was suggested based on implicit feedback data by utilizing graph-based algorithms for making song recommendations based on the user's preferences and behavior [317]. Moreover, PSO-based web page recommendation system has been developed based on health-care multimedia data to track user navigation behavior by utilizing semantic web usage mining [318]. In the future work, these systems can be further extended to include *e*-health care applications and some social networking sites like Twitter and Facebook.

10. Conclusions

In the previous sections, a rigorous yet systematic literature review is presented to highlight the major advances of PSO in terms of paradigm, theory, variation, and application, besides describing a few notable developments in the field, over the last two decades. Some observations of the collective behavior of bird flocking were the inspiration source of the PSO algorithm. Over the years, due to its simplicity and because no assumptions are made on the characteristics of the objective function to be optimized, many researchers have paid great attention to the algorithm. To improve its effectiveness and efficiency, the PSO algorithm has seen inevitably several changes, some of which involves the suggestion of strategies for initialization, control parameter adaption, swarm size adaption, and different neighborhood topologies, and fitness landscape analysis. PSO has been widely applied in various domains, reporting issues with the algorithm in regards to convergence, diversity, and stability. In an attempt to restrain these problems, developments were made in the algorithm, mostly by introducing new parameters or combining the algorithmic components of PSO with those of other algorithms. The algorithm has also been adapted for solving various complex optimization scenarios since its very inception in the mid-1990s. Large-scale, constrained, multimodal, multi-objective, and discrete optimization problems were some of the major problems solved with the algorithm.

In terms of swarm-based meta-heuristics, this study is limited to PSOs. However, the findings may be applicable to other swarm-based techniques for analyzing optimization problems. Releasing a unified algorithmic framework beyond the scope of this study with less user-specified customizations and more intelligent self-adaptation is a promising area for future research.

To conclude, PSOs have demonstrated a noticeable advantage in many application areas, compared to many other optimization algorithms. Despite having some unavoidable drawbacks, those were diminished by diverse strategies and modifications in the canonical ω-PSO. PSO is also problem-independent (i.e., due to its superior capacity for abstraction). It can be applied with versatility, which further highlights its effectiveness. Such flexibility has made PSO a robust optimizer in widely varying, yet challenging optimization scenarios. Now, "This is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.[2]" PSO has been invented for a quarter century and is firmly believed to continue to be so active and vibrant for multi- and inter-disciplinary research in the years to come.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Essam H. Houssein: Supervision, Project administration, Conceptualization, Methodology, Formal analysis, Writing - review & editing. **Ahmed G. Gad:** Conceptualization, Methodology, Validation, Formal analysis, Resources, Visualization, Investigation, Writing - original draft, Writing - review & editing. **Kashif Hussain:** Validation, Writing - review & editing. **Ponnuthurai Nagaratnam Suganthan:** Supervision, Formal analysis, Writing - review & editing.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 4, IEEE, 1995, pp. 1942–1948.
[2] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Ieee, 1995, pp. 39–43.
[3] C.W. Cleghorn, A.P. Engelbrecht, Particle swarm convergence: an empirical investigation, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 2524–2530.
[4] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: International conference on evolutionary programming, Springer, 1998, pp. 591–600.
[5] H. Ye, W. Luo, Z. Li, Convergence analysis of particle swarm optimizer and its improved algorithm based on velocity differential evolution, Comput. Intell. Neurosci. 2013 (2013).
[6] R.B. Larsen, J. Jouffroy, B. Lassen, On the premature convergence of particle swarm optimization, in: 2016 European Control Conference (ECC), IEEE, 2016, pp. 1922–1927.
[7] J.C. Bansal, P. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in: 2011 Third World Congress on Nature and Biologically Inspired Computing, IEEE, 2011, pp. 633–640.
[8] M. Gang, Z. Wei, C. Xiaolin, A novel particle swarm optimization algorithm based on particle migration, Appl. Math. Comput. 218 (11) (2012) 6620–6626.
[9] B. Nakisa, M.N. Rastgoo, M.J. Norodin, et al., Balancing exploration and exploitation in particle swarm optimization on search tasking, Res. J. Appl. Sci. Eng. Technol 8 (12) (2014) 1429–1434.
[10] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2014) 191–204.
[11] G. Xu, Z.-H. Wu, M.-Z. Jiang, Premature convergence of standard particle swarm optimisation algorithm based on Markov chain analysis, Int. J. Wirel. Mob.Comput. 9 (4) (2015) 377–382.
[12] C. Worasucheep, A particle swarm optimization for high-dimensional function optimization, in: ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, IEEE, 2010, pp. 1045–1049.
[13] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.S.-H. Chung, Y. Li, Y.-H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2012) 241–258.
[14] E.T. Oldewage, A.P. Engelbrecht, C.W. Cleghorn, Movement patterns of a particle swarm in high dimensional spaces, Inf. Sci. 512 (2020) 1043–1062.
[15] K. Hussain, M.N.M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (4) (2019) 2191–2233.

[2] Winston Churchill, 10 November, 1942.

[16] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: background and development, Nat. Comput. 6 (4) (2007) 467–484.

[17] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Nat. Comput. 7 (1) (2008) 109–124.

[18] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.

[19] V. Kothari, J. Anuradha, S. Shah, P. Mittal, A survey on particle swarm optimization in feature selection, in: International Conference on Computing and Communication Systems, Springer, 2011, pp. 192–201.

[20] R.V. Kulkarni, G.K. Venayagamoorthy, Particle swarm optimization in wireless-sensor networks: a brief survey, IEEE Trans. Syst. Man Cybern.Part C 41 (2) (2011) 262–267.

[21] S. Alam, G. Dobbie, Y.S. Koh, P. Riddle, S.U. Rehman, Research on particle swarm optimization based clustering: a systematic review of literature and techniques, Swarm Evol. Comput. 17 (2014) 1–13.

[22] M. Imran, R. Hashim, N.E.A. Khalid, An overview of particle swarm optimization variants, Procedia Eng. 53 (2013) 491–496.

[23] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput. 22 (2) (2018) 387–408.

[24] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Prob. Eng. 2015 (2015).

[25] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: where we stand and what's next, Swarm Evol. Comput. 48 (2019) 220–250.

[26] E. Camci, D.R. Kripalani, L. Ma, E. Kayacan, M.A. Khanesar, An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm, Swarm Evol. Comput. 41 (2018) 1–8.

[27] S.T. Ramya, B. Arunagiri, P. Rangarajan, Novel effective x-path particle swarm optimization based deprived video data retrieval for smart city, Clust. Comput. (2017) 1–10.

[28] W. Srisukkham, L. Zhang, S.C. Neoh, S. Todryk, C.P. Lim, Intelligent leukaemia diagnosis with bare-bones PSObased feature optimization, Appl. Soft Comput. 56 (2017) 405–419.

[29] N.H.A. Rahman, A.F. Zobaa, Integrated mutation strategy with modified binary PSOalgorithm for optimal PMUS placement, IEEE Trans. Ind. Inform. 13 (6) (2017) 3124–3133.

[30] D. Pradeepkumar, V. Ravi, Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network, Appl. Soft Comput. 58 (2017) 35–52.

[31] N. Mansouri, B.M.H. Zade, M.M. Javidi, Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory, Comput. Ind. Eng. 130 (2019) 597–633.

[32] E.H. Houssein, A.G. Gad, Y.M. Wazery, Jaya algorithm and applications: a comprehensive review, Metaheurist. Optim. Comput. Electr. Eng. (2021) 3–24.

[33] E.H. Houssein, A.G. Gad, Y.M. Wazery, P.N. Suganthan, Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends, Swarm Evol. Comput. 62 (2021) 100841.

[34] F.A. Hashim, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: a novel physics-based algorithm, Future Gener. Comput. Syst. 101 (2019) 646–667.

[35] E.H. Houssein, M.R. Saad, F.A. Hashim, H. Shaban, M. Hassaballah, Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems, Eng. Appl. Artif.Intell. 94 (2020) 103731.

[36] X. Liang, W. Li, Y. Zhang, Y. Zhong, M. Zhou, Recent advances in particle swarm optimization via population structuring and individual behavior control, in: 2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2013, pp. 503–508.

[37] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 69–73.

[38] A. Serani, C. Leotardi, U. Iemma, E.F. Campana, G. Fasano, M. Diez, Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems, Appl. Soft Comput. 49 (2016) 313–334.

[39] A. Engelbrecht, Particle swarm optimization: velocity initialization, in: 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.

[40] S. Gunasundari, S. Janakiraman, S. Meenambal, Velocity bounded boolean particle swarm optimization for improved feature selection in liver and kidney disease diagnosis, Expert Syst. Appl. 56 (2016) 28–47.

[41] A. Marandi, F. Afshinmanesh, M. Shahabadi, F. Bahrami, Boolean particle swarm optimization and its application to the design of a dual-band dual-polarized planar antenna, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 3212–3218.

[42] P. Bhambu, S. Kumar, K. Sharma, Self balanced particle swarm optimization, Int. J. Syst. Assur. Eng. Manag. 9 (4) (2018) 774–783.

[43] T. Sen, N. Pragallapati, V. Agarwal, R. Kumar, Global maximum power point tracking of PV arrays under partial shading conditions using a modified particle velocity-based PSOtechnique, IET Renew. Power Gener. 12 (5) (2017) 555–564.

[44] J. Guo, B. Wang, Particle swarm optimization with gaussian disturbance, in: 2017 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), IEEE, 2017, pp. 266–269.

[45] G. Tambouratzis, Modifying the velocity in adaptive PSO to improve optimisation performance, in: 2017 Ninth International Conference on Advanced Computational Intelligence (ICACI), IEEE, 2017, pp. 149–156.

[46] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: 2007 IEEE Swarm Intelligence Symposium, IEEE, 2007, pp. 120–127.

[47] M. Pluhacek, R. Senkerik, A. Viktorin, T. Kadavy, Study on velocity clamping in PSO using CEC'13 benchmark, in: ECMS, 2018, pp. 150–155.

[48] J.F. Schutte, A.A. Groenwold, A study of global optimization using particle swarms, J. Glob. Optim. 31 (1) (2005) 93–108.

[49] S. Sakamoto, T. Oda, M. Ikeda, L. Barolli, F. Xhafa, Implementation of a new replacement method in WMN-PSOsimulation system and its performance evaluation, in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2016, pp. 206–211.

[50] M.S. Kiran, Particle swarm optimization with a new update mechanism, Appl. Soft Comput. 60 (2017) 670–678.

[51] K.M. Ang, W.H. Lim, N.A.M. Isa, S.S. Tiang, C.H. Wong, A constrained multi-swarm particle swarm optimization without velocity for constrained optimization problems, Expert Syst. Appl. 140 (2020) 112882.

[52] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2–4) (2000) 311–338.

[53] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1951–1957.

[54] M. Isiet, M. Gadala, Sensitivity analysis of control parameters in particle swarm optimization, J. Comput. Sci. 41 (2020) 101086.

[55] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 1, IEEE, 2001, pp. 94–100.

[56] F. Van den Bergh, A. Engelbrecht, Particle swarm weight initialization in multi-layer perceptron artificial neural networks, Dev. Pract. Artif. Intell.Tech. 41 (1999).

[57] Y. Li, Z.-H. Zhan, S. Lin, J. Zhang, X. Luo, Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems, Inf. Sci. 293 (2015) 370–382.

[58] E. Ozcan, C.K. Mohan, Particle swarm optimization: surfing the waves, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1939–1944.

[59] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), 1, IEEE, 2000, pp. 84–88.

[60] M.U. Farooq, A. Ahmad, A. Hameed, Opposition-based initialization and a modified pattern for inertia weight (IW) in PSO, in: 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), IEEE, 2017, pp. 96–101.

[61] A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Comput. Oper. Res. 33 (3) (2006) 859–871.

[62] I.K. Gupta, A. Choubey, S. Choubey, Particle swarm optimization with selective multiple inertia weights, in: 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2017, pp. 1–6.

[63] H. Liang, F. Kang, Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight, Optik 127 (19) (2016) 8036–8042.

[64] C. Ze, D. Mengnan, Y. Tiankai, H. Lijie, Extraction of solar cell model parameters based on self-adaptive chaos particle swarm optimization algorithm, Trans. China Electrotech. Soc. 29 (9) (2014) 245–252.

[65] M.R. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, Inf. Sci. 294 (2015) 182–202.

[66] B. Borowska, Nonlinear inertia weight in particle swarm optimization, in: 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 1, IEEE, 2017, pp. 296–299.

[67] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. 38 (2016) 281–295.

[68] A. Samanta, K. Basu, A novel particle swarm optimization with fuzzy adaptive inertia weight for reliability redundancy allocation problems, Intell. Decis. Technol. 13 (1) (2019) 91–99.

[69] G.L. Dias, R.R. Magalhães, D.D. Ferreira, B.H.G. Barbosa, Young's modulus and poisson's ratio estimation based on PSO constriction factor method parameters evaluation, Int. J. Manuf. Mater. Mech. Eng. 9 (2) (2019) 33–46.

[70] H.S. Maharana, S.K. Dash, Comparative optimization analysis of ramp rate constriction factor based PSO and electro magnetism based PSO for economic load dispatch in electric power system, in: 2019 International Conference on Applied Machine Learning (ICAML), IEEE, 2019, pp. 63–68.

[71] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE, 1997, pp. 303–308.

[72] X. Fei, S. Youfu, R. Xuejun, A simulation analysis method based on PSO-RBF model and its application, Clust. Comput. 22 (1) (2019) 2255–2261.

[73] M. Clerc, The way of life cheap-PSO, an adaptive PSO, in: Proceedings of IEEE Congress on Evolutionary Computation, 3, 2000, pp. 1951–1957.

[74] M.Z. Shirazi, T. Pamulapati, R. Mallipeddi, K.C. Veluvolu, Particle swarm optimization with ensemble of inertia weight strategies, in: International Conference on Swarm Intelligence, Springer, 2017, pp. 140–147.

[75] J. Wu, C. Song, C. Fan, A. Hawbani, L. Zhao, X. Sun, DenPSO: a distance evolution nonlinear PSO algorithm for energy-efficient path planning in 3d UASNS, IEEE Access 7 (2019) 105514–105530.

[76] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255 .

[77] K.R. Harrison, B.M. Ombuki-Berman, A.P. Engelbrecht, An analysis of control parameter importance in the particle swarm optimization algorithm, in: International Conference on Swarm Intelligence, Springer, 2019, pp. 93–105.
[78] M.S. Arumugam, M. Rao, On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems, Discret. Dyn. Nat. Soc. 2006 (2006).
[79] P. Cazzaniga, M.S. Nobile, D. Besozzi, The impact of particles initialization in PSO: parameter estimation as a case in point, in: 2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), IEEE, 2015, pp. 1–8.
[80] H. Djellali, N. Ghoualmi, Improved chaotic initialization of particle swarm applied to feature selection, in: 2019 International Conference on Networking and Advanced Systems (ICNAS), IEEE, 2019, pp. 1–5.
[81] A. Tharwat, M. Elhoseny, A.E. Hassanien, T. Gabel, A. Kumar, Intelligent Bézier curve-based path planning model using chaotic particle swarm optimization algorithm, Clust. Comput. 22 (2) (2019) 4745–4766.
[82] Q. Li, S.-Y. Liu, X.-S. Yang, Influence of initialization on the performance of metaheuristic optimizers, Appl. Soft Comput. (2020) 106193.
[83] A.P. Piotrowski, J.J. Napiorkowski, A.E. Piotrowska, Population size in particle swarm optimization, Swarm Evol. Comput. (2020) 100718.
[84] M. Prathabrao, A. Nawawi, N.A. Sidek, Swarm size and iteration number effects to the performance of PSO algorithm in RFID tag coverage optimization, in: AIP Conference Proceedings, 1831, AIP Publishing LLC, 2017, p. 020051.
[85] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2, IEEE, 2002, pp. 1671–1676.
[86] J.F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka, A.D. George, Parallel global optimization with the particle swarm algorithm, Int. J. Numer. Methods Eng. 61 (13) (2004) 2296–2315.
[87] M.Z. Ali, P.N. Suganthan, R.G. Reynolds, A.F. Al-Badarneh, Leveraged neighborhood restructuring in cultural algorithms for solving real-world numerical optimization problems, IEEE Trans. Evol. Comput. 20 (2) (2015) 218–231.
[88] B.-Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2012) 387–402.
[89] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1931–1938.
[90] Y. Shi, H. Liu, L. Gao, G. Zhang, Cellular particle swarm optimization, Inf. Sci. 181 (20) (2011) 4460–4493.
[91] E. Alba, E. Talbi, G. Luque, N. Melab, Metaheuristics and parallelism, Parallel Metaheurist. (2005) 79–104.
[92] J. Cohoon, W. Martin, J. Lienig, Island (migration) models: evolutionary algorithms based on punctuated equilibria, Handb. Evol. Comput. C 6.3 (1997) 1–6.
[93] N. Lynn, M.Z. Ali, P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm Evol. Comput. 39 (2018) 24–35.
[94] Y. Davidor, A naturally occuring niche and species phenomenon: the model and first results, in: Proc. 4th Int. Conf. on Genetic Algorithms (San Diego, CA.), 1991, pp. 257–263.
[95] S.-Z. Zhao, P. Suganthan, Two-lbests based multi-objective particle swarm optimizer, Eng. Optim. 43 (1) (2011) 1–17.
[96] S.-Z. Zhao, M.W. Iruthayarajan, S. Baskar, P.N. Suganthan, Multi-objective robust PID controller tuning using two lbests multi-objective particle swarm optimization, Inf. Sci. 181 (16) (2011) 3323–3335.
[97] A.B. Hashemi, M.R. Meybodi, Cellular PSO: a PSO for dynamic environments, in: International Symposium on Intelligence Computation and Applications, Springer, 2009, pp. 422–433.
[98] L. Gao, J. Huang, X. Li, An effective cellular particle swarm optimization for parameters optimization of a multi-pass milling process, Appl. Soft Comput. 12 (11) (2012) 3490–3499.
[99] R. Eberhart, P. SimPSOn, R. Dobbins, Computational intelligence pc tools, 1996.
[100] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1958–1962.
[101] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Appl. Soft Comput. 48 (2016) 584–596.
[102] X. Xia, L. Gui, Z.-H. Zhan, A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting, Appl. Soft Comput. 67 (2018) 126–140.
[103] R. Brits, A.P. Engelbrecht, F. Van den Bergh, A niching particle swarm optimizer, in: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, 2, 2002, pp. 692–696.
[104] F. Van den Bergh, A.P. Engelbrecht, A new locally convergent particle swarm optimiser, in: IEEE International Conference on Systems, Man and Cybernetics, 3, IEEE, 2002, pp. 6–pp.
[105] L. Huang, C.-T. Ng, A.H. Sheikh, M.C. Griffith, Niching particle swarm optimization techniques for multimodal buckling maximization of composite laminates, Appl. Soft Comput. 57 (2017) 495–503.
[106] Y. Li, Y. Chen, J. Zhong, Z. Huang, Niching particle swarm optimization with equilibrium factor for multi-modal optimization, Inf. Sci. 494 (2019) 233–246.
[107] J.-J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., IEEE, 2005, pp. 124–129.
[108] S.-Z. Zhao, J.J. Liang, P.N. Suganthan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3845–3852.
[109] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 9–16.
[110] W. Ye, W. Feng, S. Fan, A novel multi-swarm particle swarm optimization with dynamic learning strategy, Appl. Soft Comput. 61 (2017) 832–843.
[111] A. Ayari, S. Bouamama, A new multiple robot path planning algorithm: dynamic distributed particle swarm optimization, Robot. Biomimet. 4 (1) (2017) 8.
[112] S. Akhmedova, V. Stanovov, E. Semenkin, Soft island model for population-based optimization algorithms, in: International Conference on Swarm Intelligence, Springer, 2018, pp. 68–77.
[113] H. Ikegami, H. Mori, Development of dePSO island model with particle speed limit for distribution network reconfigurations, IFAC-PapersOnLine 51 (28) (2018) 552–557.
[114] S. Lalwani, H. Sharma, S.C. Satapathy, K. Deep, J.C. Bansal, A survey on parallel particle swarm optimization algorithms, Arab. J. Sci. Eng. 44 (4) (2019) 2899–2923.
[115] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE Trans. Evol. Comput. 4 (4) (2000) 337–352.
[116] S. Wright, The roles of mutation, inbreeding, crossbreeding, and selection in evolution, 1, na, 1932.
[117] B. Chopard, M. Tomassini, Particle swarm optimization, in: An Introduction to Metaheuristics for Optimization, Springer, 2018, pp. 97–102.
[118] A. Engelbrecht, P. Bosman, K. Malan, The influence of fitness landscape characteristics on particle swarm optimisers, Nat. Comput. (2021) 1–11.
[119] D.A. Levinthal, Adaptation on rugged landscapes, Manag. Sci. 43 (7) (1997) 934–950.
[120] R. Palmer, Optimization on rugged landscapes, in: Molecular Evolution on Rugged Landscapes, CRC Press, 2018, pp. 3–25.
[121] V.K. Vassilev, T.C. Fogarty, J.F. Miller, Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application, in: Advances in evolutionary computing, Springer, 2003, pp. 3–44.
[122] W.A. van Aardt, A.S. Bosman, K.M. Malan, Characterising neutrality in neural network error landscapes, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1374–1381.
[123] K. Li, Z. Liang, S. Yang, Z. Chen, H. Wang, Z. Lin, Performance analyses of differential evolution algorithm based on dynamic fitness landscape, Int. J. Cognit. Inform.Nat. Intell. 13 (1) (2019) 36–61.
[124] M. Lunacek, D. Whitley, The dispersion metric and the CMA evolution strategy, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006, pp. 477–484.
[125] T. Jones, S. Forrest, et al., Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: ICGA, 95, 1995, pp. 184–192.
[126] F. Van den Bergh, A.P. Engelbrecht, Cooperative learning in neural networks using particle swarm optimizers, S. Afr. Comput. J. 2000 (26) (2000) 84–90.
[127] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[128] X. Li, X. Yao, Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 1546–1553.
[129] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2011) 210–224.
[130] H. Ge, L. Sun, G. Tan, Z. Chen, C.P. Chen, Cooperative hierarchical PSO with two stage variable interaction reconstruction for large scale optimization, IEEE Trans. Cybern. 47 (9) (2017) 2809–2823.
[131] F. Bourennani, Cooperative asynchronous parallel particle swarm optimization for large dimensional problems, Int. J. Appl. Metaheurist.Comput. 10 (3) (2019) 19–38.
[132] K.E. Parsopoulos, M.N. Vrahatis, et al., Particle swarm optimization method for constrained optimization problems, Intell. Technol.–Theory Appl. 76 (1) (2002) 214–220.
[133] X. Hu, R. Eberhart, Solving constrained nonlinear optimization problems with particle swarm optimization, in: Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics, 5, Citeseer, 2002, pp. 203–206.
[134] X. Hu, R.C. Eberhart, Y. Shi, Engineering optimization with particle swarm, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 53–57.
[135] S. He, E. Prempain, Q. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Optim. 36 (5) (2004) 585–605.
[136] C.-l. Sun, J.-c. Zeng, J.-s. Pan, An improved vector particle swarm optimization for constrained optimization problems, Inf. Sci. 181 (6) (2011) 1153–1163.
[137] K. Parsopoulos, V. Plagianakos, G. Magoulas, M. Vrahatis, Stretching technique for obtaining global minimizers through particle swarm optimization, in: Proceedings of the Particle Swarm Optimization Workshop, 29, Indianapolis, USA, 2001, pp. 1–8.
[138] K. Parsopoulos, M. Vrahatis, Modification of the particle swarm optimizer for locating all the global minima, in: Artificial Neural Nets and Genetic Algorithms, Springer, 2001, pp. 324–327.
[139] R. Brits, A. Engelbrecht, F. Van den Bergh, Solving systems of unconstrained equations using particle swarm optimization, in: IEEE International Conference on Systems, Man and Cybernetics, 3, IEEE, 2002, pp. 6–pp.
[140] K.E. Parsopoulos, M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 211–224.

[141] X. Hu, R. Eberhart, Multiobjective optimization using dynamic neighborhood particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2, Ieee, 2002, pp. 1677–1681.

[142] C.C. Coello, M.S. Lechuga, MoPSO: a proposal for multiple objective particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2, IEEE, 2002, pp. 1051–1056.

[143] S. Mostaghim, J. Teich, Strategies for finding good local guides in multi-objective particle swarm optimization (moPSO), in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 26–33.

[144] X. Zhang, X. Zheng, R. Cheng, J. Qiu, Y. Jin, A competitive mechanism based multi-objective particle swarm optimizer with fast convergence, Inf. Sci. 427 (2018) 63–76.

[145] M. Adhikari, S.N. Srirama, Multi-objective accelerated particle swarm optimization with a container-based scheduling for internet-of-things in cloud environment, J. Netw. Comput. Appl. 137 (2019) 35–61.

[146] W. Cook, L. Lovász, P.D. Seymour, et al., Combinatorial Optimization: Papers From the DIMACS Special Year, 20, American Mathematical Soc, 1995.

[147] B. Jarboui, N. Damak, P. Siarry, A. Rebai, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, Appl. Math. Comput. 195 (1) (2008) 299–308.

[148] F. Afshinmanesh, A. Marandi, A. Rahimi-Kian, A novel binary particle swarm optimization method using artificial immune system, in: EUROCON 2005-The International Conference on" Computer as a Tool", 1, IEEE, 2005, pp. 217–220.

[149] K.V. Deligkaris, Z.D. Zaharis, D.G. Kampitaki, S.K. Goudos, I.T. Rekanos, M.N. Spasos, Thinned planar array design using boolean PSO with velocity mutation, IEEE Trans. Magn. 45 (3) (2009) 1490–1493.

[150] W.-N. Chen, J. Zhang, H.S. Chung, W.-L. Zhong, W.-G. Wu, Y.-H. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evol. Comput. 14 (2) (2009) 278–300.

[151] S. Nema, J. Goulermas, G. Sparrow, P. Cook, A hybrid particle swarm branch-and-bound (HPB) optimizer for mixed discrete nonlinear programming, IEEE Trans. Syst. Man Cybern.-Part A 38 (6) (2008) 1411–1424.

[152] S. Chowdhury, J. Zhang, A. Messac, Avoiding premature convergence in a mixed-discrete particle swarm optimization (mdPSO) algorithm, in: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA, 2012, p. 1678.

[153] M. Gong, J. Yan, B. Shen, L. Ma, Q. Cai, Influence maximization in social networks based on discrete particle swarm optimization, Inf. Sci. 367 (2016) 600–614.

[154] S. Aminbakhsh, R. Sonmez, Discrete particle swarm optimization method for the large-scale discrete time–cost trade-off problem, Expert Syst. Appl. 51 (2016) 177–185.

[155] X. Xu, H. Rong, M. Trovati, M. Liptrott, N. Bessis, Cs-PSO: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems, Soft Comput. 22 (3) (2018) 783–795.

[156] J. Stork, M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, T. Tušar, Open issues in surrogate-assisted optimization, in: High-Performance Simulation-Based Optimization, Springer, 2020, pp. 225–244.

[157] Y. Jin, M. Olhofer, B. Sendhoff, A framework for evolutionary optimization with approximate fitness functions, IEEE Trans. Evol. Comput. 6 (5) (2002) 481–494.

[158] C. Praveen, R. Duvigneau, Low cost PSO using metamodels and inexact pre-evaluation: application to aerodynamic shape design, Comput. Methods Appl. Mech.Eng. 198 (9–12) (2009) 1087–1096.

[159] Y. Tang, J. Chen, J. Wei, A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions, Eng. Optim. 45 (5) (2013) 557–576.

[160] H. Wang, Y. Jin, J. Doherty, Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems, IEEE Trans. Cybern. 47 (9) (2017) 2664–2677.

[161] H. Yu, Y. Tan, J. Zeng, C. Sun, Y. Jin, Surrogate-assisted hierarchical particle swarm optimization, Inf. Sci. 454 (2018) 59–72.

[162] C. Sun, Y. Jin, R. Cheng, J. Ding, J. Zeng, Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems, IEEE Trans. Evol. Comput. 21 (4) (2017) 644–660.

[163] X. Hu, R.C. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2, IEEE, 2002, pp. 1666–1670.

[164] X.-F. Xie, W.-J. Zhang, Z.-L. Yang, Adaptive particle swarm optimization on individual level, in: 6th International Conference on Signal Processing, 2002, 2, IEEE, 2002, pp. 1215–1218.

[165] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern.Part B 39 (6) (2009) 1362–1381.

[166] M. Isiet, M. Gadala, Self-adapting control parameters in particle swarm optimization, Appl. Soft Comput. 83 (2019) 105653.

[167] K.R. Harrison, A.P. Engelbrecht, B.M. Ombuki-Berman, Self-adaptive particle swarm optimization: a review and analysis of convergence, Swarm Intell. 12 (3) (2018) 187–226.

[168] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 1, IEEE, 2001, pp. 101–106.

[169] P. Bajpai, S. Singh, Fuzzy adaptive particle swarm optimization for bidding strategy in uniform price spot market, IEEE Trans. Power Syst. 22 (4) (2007) 2152–2160.

[170] M.S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, G. Pasi, Fuzzy self–tuning PSO: a settings-free algorithm for global optimization, Swarm Evol. Comput. 39 (2018) 70–85.

[171] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.

[172] E.M. Mansour, A. Ahmadi, A novel clustering algorithm based on fully-informed particle swarm, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 713–720.

[173] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

[174] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. 178 (15) (2008) 3096–3109.

[175] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.

[176] V.L. Huang, P.N. Suganthan, J.J. Liang, Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems, Int. J. Intell. Syst. 21 (2) (2006) 209–226.

[177] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.

[178] M.A.M. De Oca, J. Pena, T. Stutzle, C. Pinciroli, M. Dorigo, Heterogeneous particle swarm optimizers, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 698–705.

[179] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533–548.

[180] J.-J. Wang, G.-Y. Liu, Saturated control design of a quadrotor with heterogeneous comprehensive learning particle swarm optimization, Swarm Evol. Comput. 46 (2019) 84–96.

[181] D. Yousri, D. Allam, M. Eteiba, P.N. Suganthan, Static and dynamic photovoltaic models' parameters identification using chaotic heterogeneous comprehensive learning particle swarm optimizer variants, Energy Convers. Manag. 182 (2019) 546–563.

[182] J. Kennedy, Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 80–87.

[183] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (6) (2003) 317–325.

[184] R. Vafashoar, M.R. Meybodi, Cellular learning automata based bare bones PSO with maximum likelihood rotated mutations, Swarm Evol. Comput. 44 (2019) 680–694.

[185] A.M. Durán-Rosal, P.A. Gutiérrez, Á. Carmona-Poyato, C. Hervás-Martínez, A hybrid dynamic exploitation barebones particle swarm optimisation algorithm for time series segmentation, Neurocomputing 353 (2019) 45–55.

[186] S. Das, A. Abraham, S.K. Sarkar, A hybrid rough set–particle swarm algorithm for image pixel classification, in: 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), IEEE, 2006, p. 26.

[187] B. Alatas, E. Akin, Rough particle swarm optimization and its applications in data mining, Soft Comput. 12 (12) (2008) 1205–1218.

[188] J.-C. Fan, Y. Li, L.-Y. Tang, G.-K. Wu, RoughPSO: rough set-based particle swarm optimisation, Int. J. Bio-Inspired Comput. 12 (4) (2018) 245–253.

[189] A. Bhattacharya, R.T. Goswami, K. Mukherjee, A feature selection technique based on rough set and improvised PSO algorithm (PSOrs-fs) for permission based detection of android malwares, Int. J. Mach. Learn.Cybern. (2018) 1–15.

[190] F. Van den Bergh, A.P. Engelbrecht, A convergence proof for the particle swarm optimiser, Fundam. Inform. 105 (4) (2010) 341–374.

[191] N. Higashi, H. Iba, Particle swarm optimization with gaussian mutation, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 72–79.

[192] E.S. Peer, F. Van den Bergh, A.P. Engelbrecht, Using neighbourhoods with the guaranteed convergence PSO, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 235–242.

[193] J. Sun, B. Feng, W. Xu, Particle swarm optimization with particles having quantum behavior, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), 1, IEEE, 2004, pp. 325–331.

[194] W. Fang, J. Sun, H. Chen, X. Wu, A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population, Inf. Sci. 330 (2016) 19–48.

[195] Q. Qian, J. Wu, Z. Wang, Optimal path planning for two-wheeled self-balancing vehicle pendulum robot based on quantum-behaved particle swarm optimization algorithm, Pers. Ubiquitous Comput. (2019) 1–11.

[196] D. Gies, Y. Rahmat-Samii, Reconfigurable array design using parallel particle swarm optimization, in: IEEE Antennas and Propagation Society International Symposium. Digest. Held in conjunction with: USNC/CNC/URSI North American Radio Sci. Meeting (Cat. No. 03CH37450), 1, IEEE, 2003, pp. 177–180.

[197] S. Baskar, P.N. Suganthan, A novel concurrent particle swarm optimization, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), 1, IEEE, 2004, pp. 792–796.

[198] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE, 2003, pp. 174–181.

[199] F. Han, W. Cui, G. Wei, S. Wu, Application of parallel PSO algorithm to motion parameter estimation, in: 2008 9th International Conference on Signal Processing, IEEE, 2008, pp. 2493–2496.

[200] Ş. Gülcü, H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, Eng. Appl. Artif.Intell. 45 (2015) 33–45.

[201] B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, K. Kang, Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization, IEEE Access 5 (2017) 8214–8221.

[202] B. Rymut, B. Kwolek, Gpu-supported object tracking using adaptive appearance models and particle swarm optimization, in: International Conference on Computer Vision and Graphics, Springer, 2010, pp. 227–234.
[203] J. Li, D. Wan, Z. Chi, X. Hu, An efficient fine-grained parallel particle swarm optimization method based on gpu-acceleration, Int. J. Innov. Comput.Inf. Control 3 (6) (2007) 1707–1714.
[204] Y. Hung, W. Wang, Accelerating parallel particle swarm optimization via gpu, Optim. Methods Softw. 27 (1) (2012) 33–51.
[205] C.-L. Liao, S.-J. Lee, Y.-S. Chiou, C.-R. Lee, C.-H. Lee, Power consumption minimization by distributive particle swarm optimization for luminance control and its parallel implementations, Expert Syst. Appl. 96 (2018) 479–491.
[206] O. Awwad, A. Al-Fuqaha, G. Ben Brahim, B. Khan, A. Rayes, Distributed topology control in large-scale hybrid RF/FSO networks: Simt gpu-based particle swarm optimization approach, Int. J. Commun. Syst. 26 (7) (2013) 888–911.
[207] P.J. Angeline, Using selection to improve particle swarm optimization, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 84–89.
[208] B. Yang, Y. Chen, Z. Zhao, A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems, in: 2007 IEEE International Conference on Control and Automation, IEEE, 2007, pp. 166–170.
[209] B. Jana, S. Mitra, S. Acharyya, Repository and mutation based particle swarm optimization (rmPSO): a new PSO variant applied to reconstruction of gene regulatory network, Appl. Soft Comput. 74 (2019) 330–355.
[210] M. Løvbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimiser with breeding and subpopulations, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 469–476.
[211] V. Miranda, N. Fonseca, EPSO-best-of-two-worlds meta-heuristic applied to power system problems, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02Th8600), 2, IEEE, 2002, pp. 1080–1085.
[212] K. Premalatha, A. Natarajan, Discrete PSO with GA operators for document clustering, Int. J. Recent Trends Eng. 1 (1) (2009) 20.
[213] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.-S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[214] J. Robinson, S. Sinton, Y. Rahmat-Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna, in: IEEE Antennas and Propagation Society International Symposium (IEEE Cat. No. 02CH37313), 1, IEEE, 2002, pp. 314–317.
[215] X. Shi, Y. Lu, C. Zhou, H. Lee, W. Lin, Y. Liang, Hybrid evolutionary algorithms based on PSO and ga, in: The 2003 Congress on Evolutionary Computation, 2003. CEC'03., 4, IEEE, 2003, pp. 2393–2399.
[216] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern.Part B 34 (2) (2004) 997–1006.
[217] R.F. Abdel-Kader, Genetically improved PSO algorithm for efficient data clustering, in: 2010 Second International Conference on Machine Learning and Computing, IEEE, 2010, pp. 71–75.
[218] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Appl. Math. Comput. 274 (2016) 292–305.
[219] C. Li, R. Zhai, H. Liu, Y. Yang, H. Wu, Optimization of a heliostat field layout using hybrid PSO-ga algorithm, Appl. Therm. Eng. 128 (2018) 33–41.
[220] F. Moslehi, A. Haeri, F. Martínez-Álvarez, A novel hybrid GA–PSO framework for mining quantitative association rules, Soft Comput. 24 (6) (2020) 4645–4666.
[221] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2010) 4–31.
[222] T. Hendtlass, A combined swarm differential evolution algorithm for optimization problems, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2001, pp. 11–18.
[223] W.-J. Zhang, X.-F. Xie, DePSO: hybrid particle swarm with differential evolution operator, in: SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483), 4, IEEE, 2003, pp. 3816–3821.
[224] H. Talbi, M. Batouche, Hybrid particle swarm with differential evolution for multimodal image registration, in: 2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT'04., 3, IEEE, 2004, pp. 1567–1572.
[225] Z.-F. Hao, G.-H. Guo, H. Huang, A particle swarm optimization algorithm with differential evolution, in: 2007 International Conference on Machine Learning and Cybernetics, 2, IEEE, 2007, pp. 1031–1035.
[226] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential evolution, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.
[227] S. Wang, Y. Li, H. Yang, Self-adaptive mutation differential evolution algorithm based on particle swarm optimization, Appl. Soft Comput. 81 (2019) 105496.
[228] G. Yang, D. Chen, G. Zhou, A new hybrid algorithm of particle swarm optimization, in: International Conference on Intelligent Computing, Springer, 2006, pp. 50–60.
[229] X.-H. Wang, J.-J. Li, Hybrid particle swarm optimization with simulated annealing, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), 4, IEEE, 2004, pp. 2402–2405.
[230] F. Zhao, Q. Zhang, D. Yu, X. Chen, Y. Yang, A hybrid algorithm based on PSO and simulated annealing and its applications for partner selection in virtual enterprise, in: International Conference on Intelligent Computing, Springer, 2005, pp. 380–389.
[231] N. Sadati, T. Amraee, A.M. Ranjbar, A global particle swarm-based-simulated annealing optimization technique for under-voltage load shedding problem, Appl. Soft Comput. 9 (2) (2009) 652–657.
[232] H.-L. Shieh, C.-C. Kuo, C.-M. Chiang, Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification, Appl. Math. Comput. 218 (8) (2011) 4365–4383.
[233] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092.
[234] T. Niknam, M.R. Narimani, M. Jabbari, Dynamic optimal power flow using hybrid particle swarm optimization and simulated annealing, Int. Trans. Electr. Energy Syst. 23 (7) (2013) 975–1001.
[235] X. Wang, Q. Sun, The study of k-means based on hybrid sa-PSO algorithm, in: 2016 9th International Symposium on Computational Intelligence and Design (ISCID), 2, IEEE, 2016, pp. 211–214.
[236] F. Javidrad, M. Nazari, A new hybrid particle swarm and simulated annealing stochastic optimization method, Appl. Soft Comput. 60 (2017) 634–654.
[237] M.S. KıRan, M. GüNdüZ, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, Appl. Soft Comput. 13 (4) (2013) 2188–2203.
[238] D. Sedighizadeh, H. Mazaheripour, Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering precedence constraints, Alex. Eng. J. 57 (4) (2018) 2225–2239.
[239] S. Singh, P. Chauhan, N. Singh, Capacity optimization of grid connected solar/fuel cell energy system using hybrid abc-PSO algorithm, Int. J. Hydrogen Energy (2020).
[240] P. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[241] W. Xiong, C. Wang, A novel hybrid clustering based on adaptive aco and PSO, in: 2011 International Conference on Computer Science and Service System (CSSS), IEEE, 2011, pp. 1960–1963.
[242] Y. Liu, M. Feng, S. Shahbazzade, The container truck route optimization problem by the hybrid PSO-ACO algorithm, in: International Conference on Intelligent Computing, Springer, 2017, pp. 640–648.
[243] S. Arunachalam, T. AgnesBhomila, M.R. Babu, Hybrid particle swarm optimization algorithm and firefly algorithm based combined economic and emission dispatch including valve point effect, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2014, pp. 647–660.
[244] X. Xia, L. Gui, G. He, C. Xie, B. Wei, Y. Xing, R. Wu, Y. Tang, A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm, J. Comput. Sci. 26 (2018) 488–500.
[245] N.A. Al-Thanoon, O.S. Qasim, Z.Y. Algamal, A new hybrid firefly algorithm and particle swarm optimization for tuning parameter estimation in penalized support vector machine with application in chemometrics, Chemom. Intell. Lab. Syst. 184 (2019) 142–152.
[246] V. Enireddy, R.K. Kumar, Improved cuckoo search with particle swarm optimization for classification of compressed images, Sadhana 40 (8) (2015) 2271–2285.
[247] J. Dash, B. Dam, R. Swain, Optimal design of linear phase multi-band stop filters using improved cuckoo search particle swarm optimization, Appl. Soft Comput. 52 (2017) 435–445.
[248] T.P. Jacob, K. Pradeep, A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization, Wirel. Pers. Commun. 109 (1) (2019) 315–331.
[249] S. Manoj, S. Ranjitha, H. Suresh, Hybrid bat-PSO optimization techniques for image registration, in: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, 2016, pp. 3590–3596.
[250] A. Zarei, S.-F. Mousavi, M.E. Gordji, H. Karami, Optimal reservoir operation using bat and particle swarm algorithm and game theory based on optimal water allocation among consumers, Water Resour. Manag. 33 (9) (2019) 3071–3093.
[251] G. Villarrubia, J.F. De Paz, P. Chamoso, F. De la Prieta, Artificial neural networks used in optimization problems, Neurocomputing 272 (2018) 10–16.
[252] R.C. Eberhart, X. Hu, Human tremor analysis using particle swarm optimization, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1927–1930.
[253] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (5) (1993) 26–33.
[254] C. Zhang, H. Shao, Y. Li, Particle swarm optimisation for evolving artificial neural network, in: SMC 2000 Conference Proceedings. 2000 IEEEInternational Conference on Systems, Man and Cybernetics.'Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions'(cat. no. 0, 4, IEEE, 2000, pp. 2487–2490.
[255] T. Ince, S. Kiranyaz, M. Gabbouj, A generic and robust system for automated patient-specific classification of ECG signals, IEEE Trans. Biomed. Eng. 56 (5) (2009) 1415–1426.
[256] H. Quan, D. Srinivasan, A. Khosravi, Short-term load and wind power forecasting using neural network-based prediction intervals, IEEE Trans. Neural Netw. Learn.Syst. 25 (2) (2013) 303–315.
[257] M. Hamada, M. Hassan, Artificial neural networks and particle swarm optimization algorithms for preference prediction in multi-criteria recommender systems, in: Informatics, 5, Multidisciplinary Digital Publishing Institute, 2018, p. 25.
[258] E.K. Tang, P.N. Suganthan, X. Yao, Feature selection for microarray data using least squares SVM and particle swarm optimization, in: 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, IEEE, 2005, pp. 1–8.
[259] P. Ghamisi, J.A. Benediktsson, Feature selection based on hybridization of genetic algorithm and particle swarm optimization, IEEE Geosci. Remote Sens. Lett. 12 (2) (2014) 309–313.

[260] T.T. Hoang, M.-Y. Cho, M.N. Alam, Q.T. Vu, A novel differential particle swarm optimization for parameter selection of support vector machines for monitoring metal-oxide surge arrester conditions, Swarm Evol. Comput. 38 (2018) 120–126.
[261] M. Jiang, Y.P. Luo, S.Y. Yang, Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm, Inf. Process. Lett. 102 (1) (2007) 8–16.
[262] R. Poli, D. Broomhead, Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, 2007, pp. 134–141.
[263] C.W. Cleghorn, Particle swarm optimization: understanding order-2 stability guarantees, in: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Springer, 2019, pp. 535–549.
[264] M.R. Bonyadi, Z. Michalewicz, Stability analysis of the particle swarm optimization without stagnation assumption, IEEE Trans. Evol. Comput. 20 (5) (2015) 814–819.
[265] Z.-G. Liu, X.-H. Ji, Y.-X. Liu, Hybrid non-parametric particle swarm optimization and its stability analysis, Expert Syst. Appl. 92 (2018) 256–275.
[266] K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization for solving constrained engineering optimization problems, in: International Conference on Natural Computation, Springer, 2005, pp. 582–591.
[267] J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Trans. Syst. Man Cybern. Part C 36 (4) (2006) 515–519.
[268] S. Helwig, R. Wanka, Theoretical analysis of initial particle swarm behavior, in: International Conference on Parallel Problem Solving From Nature, Springer, 2008, pp. 889–898.
[269] M.R. Bonyadi, Z. Michalewicz, Impacts of coefficients on movement patterns in the particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 21 (3) (2016) 378–390.
[270] M. Ehteram, F. Binti Othman, Z. Mundher Yaseen, H. Abdulmohsin Afan, M. Falah Allawi, A. Najah Ahmed, S. Shahid, V. P Singh, A. El-Shafie, et al., Improving the muskingum flood routing method using a hybrid of particle swarm optimization and bat algorithm, Water 10 (6) (2018) 807.
[271] Y. Cao, Y. Ye, H. Zhao, Y. Jiang, H. Wang, Y. Shang, J. Wang, Remote sensing of water quality based on HJ-1a HSIimagery with modified discrete binary particle swarm optimization-partial least squares (mdbPSO-pls) in inland waters: a case in weishan lake, Ecol. Inform. 44 (2018) 21–32.
[272] V.P. Kour, S. Arora, Particle swarm optimization based support vector machine (P-SVM) for the segmentation and classification of plants, IEEE Access 7 (2019) 29374–29385.
[273] M. Rahgoshay, S. Feiznia, M. Arian, S.A.A. Hashemi, Simulation of daily suspended sediment load using an improved model of support vector machine and genetic algorithms and particle swarm, Arab. J. Geosci. 12 (9) (2019) 277.
[274] S. Kumar, S.K. Pal, R. Singh, A novel hybrid model based on particle swarm optimisation and extreme learning machine for short-term temperature prediction using ambient sensors, Sustain. Cities Soc. 49 (2019) 101601.
[275] M. Sato, Y. Fukuyama, T. Iizaka, T. Matsui, Total optimization of energy networks in a smart city by multi-swarm differential evolutionary particle swarm optimization, IEEE Trans. Sustain. Energy (2018).
[276] W. Hu, H. Wang, Z. Qiu, C. Nie, L. Yan, A quantum particle swarm optimization driven urban traffic light scheduling model, Neural Comput. Appl. 29 (3) (2018) 901–911.
[277] K. Ma, S. Hu, J. Yang, X. Xu, X. Guan, Appliances scheduling via cooperative multi-swarm PSO under day-ahead prices and photovoltaic generation, Appl. Soft Comput. 62 (2018) 504–513.
[278] A.R. Jordehi, Binary particle swarm optimisation with quadratic transfer function: a new binary optimisation algorithm for optimal scheduling of appliances in smart homes, Appl. Soft Comput. (2019).
[279] L. Li, L. Qin, X. Qu, J. Zhang, Y. Wang, B. Ran, Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm, Knowl.-Based Syst. 172 (2019) 1–14.
[280] Y. Zhang, Q. Zhang, A. Farnoosh, S. Chen, Y. Li, Gis-based multi-objective particle swarm optimization of charging stations for electric vehicles, Energy 169 (2019) 844–853.
[281] S. Abid, A. Zafar, R. Khalid, S. Javaid, U. Qasim, Z.A. Khan, N. Javaid, Managing energy in smart homes using binary particle swarm optimization, in: Conference on Complex, Intelligent, and Software Intensive Systems, Springer, 2017, pp. 189–196.
[282] S. Raj, K.C. Ray, Ecg signal analysis using DCT-based dost and PSO optimized SVM, IEEE Trans. Instrum.Meas. 66 (3) (2017) 470–478.
[283] Y. Li, X. Bai, L. Jiao, Y. Xue, Partitioned-cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation, Appl. Soft Comput. 56 (2017) 345–356.
[284] I. Jain, V.K. Jain, R. Jain, Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification, Appl. Soft Comput. 62 (2018) 203–215.
[285] N. Zeng, H. Qiu, Z. Wang, W. Liu, H. Zhang, Y. Li, A new switching-delayed-PSO-based optimized SVMalgorithm for diagnosis of Alzheimer's disease, Neurocomputing 320 (2018) 195–202.
[286] R.F. Lopes, F.F. Costa, A. Oliveira, A.C.d.C. Lima, Algorithm based on particle swarm applied to electrical load scheduling in an industrial setting, Energy 147 (2018) 1007–1015.
[287] C. Qi, A. Fourie, Q. Chen, Neural network and particle swarm optimization for predicting the unconfined compressive strength of cemented paste backfill, Constr. Build. Mater. 159 (2018) 473–478.
[288] M. Song, K. Chen, J. Wang, Three-dimensional wind turbine positioning using gaussian particle swarm optimization with differential evolution, J. Wind Eng. Ind. Aerodyn. 172 (2018) 317–324.
[289] N. Ghorbani, A. Kasaeian, A. Toopshekan, L. Bahrami, A. Maghami, Optimizing a hybrid wind-PV-battery system using GA-PSO and moPSO for reducing cost and increasing reliability, Energy 154 (2018) 581–591.
[290] G. Liu, W. Chen, H. Chen, Quantum particle swarm with teamwork evolutionary strategy for multi-objective optimization on electro-optical platform, IEEE Access 7 (2019) 41205–41219.
[291] A.A. Alnaqi, H. Moayedi, A. Shahsavar, T.K. Nguyen, Prediction of energetic performance of a building integrated photovoltaic/thermal system thorough artificial neural network and hybrid particle swarm optimization models, Energy Convers. Manag. 183 (2019) 137–148.
[292] L.M. Maiyar, J.J. Thakkar, Environmentally conscious logistics planning for food grain industry considering wastages employing multi objective hybrid particle swarm optimization, Transp. Res. Part E 127 (2019) 220–248.
[293] T. Yi, H. Zheng, Y. Tian, J.-p. Liu, Intelligent prediction of transmission line project cost based on least squares support vector machine optimized by particle swarm optimization, Math. Prob. Eng. 2018 (2018).
[294] J. Shen, L. Han, Design process optimization and profit calculation module development simulation analysis of financial accounting information system based on particle swarm optimization (PSO), Inf. Syst. e-Bus. Manag. (2019) 1–14.
[295] R. Jiao, X. Huang, H. Ouyang, G. Li, Q. Zheng, Z. Jiang, Optimal electric business centre location by centre–decentre quantum particle swarm optimization, Syst. Sci. Control Eng. 7 (1) (2019) 222–233.
[296] F. Sheikholeslami, N.J. Navimipour, Service allocation in the cloud environments using multi-objective particle swarm optimization algorithm based on crowding distance, Swarm Evol. Comput. 35 (2017) 53–64.
[297] S. Suresh, S. Lal, Multilevel thresholding based on chaotic darwinian particle swarm optimization for segmentation of satellite images, Appl. Soft Comput. 55 (2017) 503–522.
[298] M. Alswaitti, M. Albughdadi, N.A.M. Isa, Density-based particle swarm optimization algorithm for data clustering, Expert Syst. Appl. 91 (2018) 170–186.
[299] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, A.C. Ammari, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, J. Intell. Manuf. 29 (3) (2018) 603–615.
[300] S. Thabit, A. Mohades, Multi-robot path planning based on multi-objective particle swarm optimization, IEEE Access 7 (2018) 2138–2147.
[301] Z. Sun, Y. Liu, L. Tao, Attack localization task allocation in wireless sensor networks based on multi-objective binary particle swarm optimization, J. Netw. Comput. Appl. 112 (2018) 29–40.
[302] Y. Zhong, J. Lin, L. Wang, H. Zhang, Discrete comprehensive learning particle swarm optimization algorithm with metropolis acceptance criterion for traveling salesman problem, Swarm Evol. Comput. 42 (2018) 77–88.
[303] G. Lin, J. Guan, Z. Li, H. Feng, A hybrid binary particle swarm optimization with tabu search for the set-union knapsack problem, Expert Syst. Appl. (2019).
[304] R. Sivaranjani, S.M.M. Roomi, M. Senthilarasi, Speckle noise removal in SAR images using multi-objective PSO (moPSO) algorithm, Appl. Soft Comput. 76 (2019) 671–681.
[305] A. Bhattacharya, R.T. Goswami, K. Mukherjee, A feature selection technique based on rough set and improvised PSO algorithm (PSOrs-fs) for permission based detection of android malwares, Int. J. Mach. Learn.Cybern. 10 (7) (2019) 1893–1907.
[306] F. Liu, H. Huang, X. Li, Z. Hao, Automated test data generation based on particle swarm optimisation with convergence speed controller, CAAI Trans. Intell. Technol. 2 (2) (2019) 73–79.
[307] B. Tran, M. Zhang, B. Xue, A PSO based hybrid feature selection algorithm for high-dimensional classification, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 3801–3808.
[308] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, Evaluating the performance of dnPSO in dynamic environments, in: 2008 IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2008, pp. 2640–2645.
[309] Q. Liu, W. Wei, H. Yuan, Z.-H. Zhan, Y. Li, Topology selection for particle swarm optimization, Inf. Sci. 363 (2016) 154–173.
[310] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1658–1665.
[311] J.-J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005, IEEE, 2005, pp. 68–75.
[312] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[313] C. Karakuzu, F. Karakaya, M.A. Çavuşlu, FPGA implementation of neuro-fuzzy system with improved PSO learning, Neural Netw. 79 (2016) 128–140.
[314] P. Suganthan, Numerical optimization by nature inspired algorithms. keynote speeches at bric cci 2015, icsi 2015, ichsa 2015, 2015.
[315] M. Abdel-Basset, A.E. Fakhry, I. El-Henawy, T. Qiu, A.K. Sangaiah, Feature and intensity based medical image registration using particle swarm optimization, J. Med. Syst. 41 (12) (2017) 197.
[316] K.-W. Huang, J.-L. Chen, C.-S. Yang, C.-W. Tsai, A memetic particle swarm optimization algorithm for solving the dna fragment assembly problem, Neural Comput. Appl. 26 (3) (2015) 495–506.
[317] R. Katarya, O.P. Verma, Efficient music recommender system using context graph and particle swarm, Multimed. Tools Appl. 77 (2) (2018) 2673–2687.
[318] R. Manikandan, V. Saravanan, A novel approach on particle agent swarm optimization (paso) in semantic mining for web page recommender system of multimedia data: a health care perspective, Multimed. Tools Appl. (2019) 1–23.