Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review

Mohammad Reza Bonyadi mrbonyadi@cs.adelaide.edu.au, rezabny@gmail.com
Department of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. Also with the Centre for Advanced Imaging (CAI), the University of Queensland, Brisbane, QLD 4067, Australia, and Complexica Pty Ltd, Adelaide, SA 5021, Australia.

Zbigniew Michalewicz zbyszek@cs.adelaide.edu.au
Department of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. Also with Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland, Polish-Japanese Institute of Information Technology, Warsaw, Poland, and Complexica Pty Ltd, Adelaide, SA 5021, Australia.

doi:10.1162/EVCO_r_00180

Abstract

This paper reviews recent studies on the Particle Swarm Optimization (PSO) algorithm. The review has been focused on high impact recent articles that have analyzed and/or modified PSO algorithms. This paper also presents some potential areas for future study.

Keywords

Particle swarm optimization, stability analysis, local convergence, invariance, topology, parameter selection, constrained optimization.

1 Motivation

Particle swarm optimization (PSO) is a stochastic population-based optimization method proposed by Kennedy and Eberhart (1995). It has been successfully applied to many problems such as artificial neural network training, function optimization, fuzzy control, and pattern classification (Engelbrecht, 2005; Poli, 2008), to name a few. Because of its ease of implementation and fast convergence to acceptable solutions, PSO has received broad attention in recent years (Poli, 2008). Since 1995, different aspects of the original version of PSO have been modified and many variants have been proposed. Although a few review articles on PSO (see Section 2 for details) have been published already (Banks et al., 2007, 2008; Hu et al., 2004; Parsopoulos and Vrahatis, 2002b; Poli, 2008; Poli, Kennedy, et al., 2007; Song and Gu, 2004), there are two important reasons for an additional review paper:

1. The latest comprehensive review paper on PSO was published in 2008; however, many new articles on PSO have been published since then. To estimate the growth of the number of publications in this field, we conducted a search for the exact match of "particle swarm optimization" in the title of documents in five recognized scientific databases (Scopus, Google Scholar, Springer, Web of

Manuscript received: July 30, 2014; revised: February 21, 2015, November 9, 2015, and January 23, 2016; accepted: February 2, 2016.

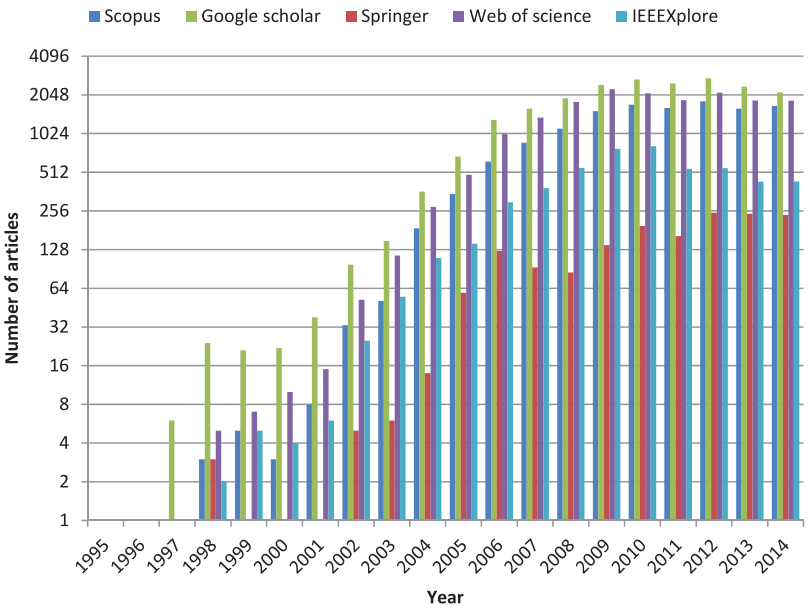

Figure 1: The number of published articles with "particle swarm optimization" in their title in Scopus, Google Scholar, Springer, Web of Science, and IEEE Xplore databases.

Science, and IEEE Xplore; see Fig. 1[1]). The ratio of the number of articles published since 2008 to the ones published before 2008 in Scopus, Google scholar, Springer, Web of science, and IEEE Xplore databases is 3.07, 2.39, 3.09, 2.34, and 2.23, respectively. In total, almost 75% of the articles on PSO have been published since 2008. As the latest comprehensive review paper on PSO was published in 2008, the need for a new review paper seems justified.

2. Several limitations were identified in standard PSO in recent years that have not been analyzed in earlier review papers (see Section 2 for more details). These limitations are related to the sensitivity of search to transformations (rotation, translation, and scale), local convergence, stability, first hitting time, and biases (Auger et al., 2009; Clerc and Kennedy, 2002; Hansen et al., 2008; Lehre and Witt, 2013; Spears et al., 2010; Van den Bergh and Engelbrecht, 2010; Wilke et al., 2007). Addressing these limitations is very important because as long as these limitations exist, good performance of PSO will not be transferable to a wide variety of optimization problems (Hansen et al., 2008, 2011). Therefore, there is a need to review advances that address these limitations in detail.

Because of a large number of published articles related to PSO we designed two criteria to select articles for review purposes as follows:

1. Criterion $C1(y, imp, hind, c)$ is met for an article if:

[1]This data was collected in August 2015. Note that more and more articles have started to use the term "PSO" rather than particle swarm optimization in their titles. However, we did not consider this in our analysis because the term "PSO" might refer to a different topic irrelevant to the particle swarm optimization. Usage of "PSO" might be one of the reasons why the trend of the number of articles per year in the topic has fallen recently.

(a) it has been published in the year y or later, and
(b) it has been published in a journal for which the impact factor[2] (reported by Thomson Reuters) is larger than *imp*, or the article has been cited more than c times, or
(c) it has been published in a conference for which the h-5[3] index (reported by the Google Scholar metric measure[4]) is higher than *hind*, or the article has been cited more than c times.

2. Criterion $C2(y, r)$ is met for an article if:
(a) it has been published before the year y, and
(b) it has been cited at least r times.

The main source for the number of citations for the articles was the Google Scholar. We used the software Harzings Publish or Perish[5] extended by some manual procedures to find articles and check whether they meet the criteria. Criterion C1 is used to select high quality articles[6] that have been published recently (it concentrates on articles published in the year y or later). Note that because a recently published article might not have a high number of citations, we have designed this criterion in such a way that highly cited articles or the articles that have been recently published in recognized journals/conferences are included in our review. Criterion C2 ensures that older articles that have played a significant role in developing the algorithm to the level that it is seen today are also selected. Note that PSO variants that belong to the family of Bare-Bones PSO (Blackwell, 2012; Kennedy, 2003) are not reviewed in this study as this topic needs a separate review paper on its own.

We have categorized the articles included in this review into three groups:

Group 1: Articles which have analyzed PSO variants from a theoretical point of view (including proper proofs in the analyses) and have identified a limitation in a PSO variant (either standard PSO or other variants). No criteria is used in this group to filter the articles,

Group 2: Articles which have proposed new variants to improve the performance of the algorithm to solve "single objective static Unconstrained Optimization Problems," UOPs.[7] The articles for review in this group are filtered by $C1(2008, 2, 30, 30)$ or $C2(2008, 300)$,

[2]See http://wokinfo.com/essays/impact-factor/ for the definition of the impact factor. This measure is reported annually for the journals which have been indexed by the Thomson Reuters Web of Science.

[3]h-5 index is the largest number (h) such that h articles published in the previous 5 years have at least h citations each.

[4]Available at http://scholar.google.com.au/citations?view_op=top_venues&hl=en&vq=eng_evolutionarycomputation.

[5]Available online at http://www.harzing.com/pop.htm, version 4.4.5 was used.

[6]The designed criteria are of course not optimal as it is possible to find low-quality papers at a high-impact journal. However, due to the large number of articles in the area of PSO, we needed to design a preliminary filter to ensure the review spends more effort on articles most likely to be of higher quality.

[7]Although large scale optimization and niching are some types of UOPs, we will not review the papers in these topics because of the extensive number of articles, which do not fit in one single review paper.

Group 3: Articles which have modified PSO to deal with "single objective static Constrained Optimization Problems," COPs (see Michalewicz, 1995 and Michalewicz and Schoenauer, 1996 for definition of COPs). The articles for review in this group are filtered by $C1(2008, 2, 20, 20)$ or $C2(2008, 150)$.

The reason behind changing the parameters for each criterion in different groups is that articles in different groups have different potential to be cited. For example, it is probable that a paper which has analyzed coefficients of the algorithm to deal with UOPs (categorized in Group 1) is cited in a paper that is related to COPs. Thus, a higher number of citations is expected for the articles in Group 2 in order to be included in this survey. Also note, for reasons of space, articles which have investigated/applied PSO for/to a specific application, such as solving the knapsack problem (Bonyadi and Michalewicz, 2012) or the traveling salesman problem (Chen et al., 2010), are not reviewed in this study. Articles in each aforementioned group are divided into several subgroups (topics) and each subgroup is discussed in one subsection. At the end of each subsection, a summary of the discussed topic together with challenges and potential future directions related to that topic are given.

The main aim of this survey is to review the presented ideas, categorize and link the most recent high-quality studies (based on the criteria we introduced previously), and provide a vision for directions that might be valuable for future research. Some conclusions derived by the authors of the included articles are difficult to evaluate as significant additional effort would be required to fully investigate their different aspects. However, we briefly discuss some of these methods from two perspectives: theoretical (convergence to local optima, transformation invariance, and the time complexity of the methods) and experimental (validity of statistical tests (Derrac et al., 2011), number of test cases, and potential biases in test cases). Although such analyses are high-level, they may assist the reader in determining to what extent the findings in these articles are reliable and how they can be tested to be confirmed or disproved.

The rest of this review study is organized as follows. Section 2 provides some information about the original form of PSO as well as earlier reviews on the PSO topic. Section 3 reviews articles which have identified limitations in PSO. Section 4 reviews articles which have modified PSO to have a better performance in solving UOPs, whereas Section 5 reviews articles which have extended PSO to deal with COPs. Section 6 concludes this paper.

2 Original PSO and Earlier Review Papers

In this paper, a UOP (minimization) is defined as:

$$\text{find } \vec{x} \in S \subseteq \mathbb{R}^d \text{ such that } \forall \vec{y} \in S,\ f(\vec{x}) \leq f(\vec{y}), \tag{1}$$

where S is the search space defined by $\{\vec{y} : l_i \leq y_i \leq u_i\}$, y_i is the i^{th} dimension of the vector $\vec{y}$, u_i and l_i are upper bound and lower bound of the i^{th} dimension, respectively, d is the number of dimensions, and $f(.)$ is the objective function. The surface represented by the function $f(.)$ is called the *landscape* of the problem.

PSO (Kennedy and Eberhart, 1995) is based on a population (referred to as a swarm) of $n > 1$ particles;[8] each particle is defined by the following three d-dimensional vectors:

[8]In theoretical studies (see Section 3), the number of particles is sometimes set to 1 to simplify the analyses.

- Position ($\vec{x}_t^i$)—is the position of the i^{th} particle in the t^{th} iteration. The quality of the particle is determined by this vector,
- Velocity ($\vec{V}_t^i$)—is the direction and length of movement of the i^{th} particle in the t^{th} iteration,
- Personal best ($\vec{p}_t^i$)—is the best position[9] that the i^{th} particle has visited in its lifetime (up to the t^{th} iteration). This vector serves as a memory to store the location of highest quality solutions found so far (Eberhart and Kennedy, 1995).

All of these vectors are updated at every iteration t for each particle i:

$$\vec{V}_{t+1}^i = \mu(\vec{x}_t^i, \vec{V}_t^i, N_t^i) \text{ for all } i \tag{2}$$

$$\vec{x}_{t+1}^i = \xi(\vec{x}_t^i, \vec{V}_{t+1}^i) \text{ for all } i \tag{3}$$

$$\vec{p}_{t+1}^i = \begin{cases} \vec{x}_{t+1}^i & f(\vec{x}_{t+1}^i) < f(\vec{p}_t^i) \text{ and } \vec{x}_{t+1}^i \in S \\ \vec{p}_t^i & \text{otherwise.} \end{cases} \text{ for all } i \tag{4}$$

In Eq. (2), N_t^i, known as the neighbor set of particle i, is a subset of personal best positions of the particles that contribute to the velocity update rule of particle i at iteration t, i.e., $N_t^i = \{\vec{p}_t^k | k \in \{T_t^i \subseteq \{1, 2, \ldots, n\}\}\}$ where T_t^i is a set of indices of particles which contribute to the velocity update rule of particle i at iteration t. Clearly, the strategy to determine T_t^i might be different for various types of PSO algorithms and it is usually referred to as the topology of the swarm. Many different topologies have been defined for PSO to date (Mendes et al., 2004), for example, global best topology, ring topology, wheel topology, pyramid typology; each of these have some advantages and disadvantages (Clerc, 2006; Mendes et al., 2004). Topology in fact determines the set of particles from which a particle should learn[10] (connect to). The function $\mu(.)$ calculates the new velocity vector for particle i according to its current position, current velocity $\vec{V}_t^i$, and neighbor set N_t^i. In Eq. (3), $\xi(.)$ is a function which calculates the new position of particle i according to its previous position and its new velocity. Usually $\xi(\vec{x}_t^i, \vec{V}_{t+1}^i) = \vec{x}_t^i + \vec{V}_{t+1}^i$ is used for updating the position of particle i. In Eq. (4), the new personal best position for particle i is updated according to the objective value of its previous personal best position and the current position. In PSO, the three update rules (Eqs. 2, 3, and 4) are applied to all particles iteratively until a predefined stopping criterion is met (e.g., the maximum number of iterations is achieved or a sufficiently good objective value is found). Also, $\vec{x}_0^i$ and $\vec{V}_0^i$ are generated either randomly or by using a heuristic method and $\vec{p}_0^i$ is initialized to $\vec{x}_0^i$ for all particles.

In the first version of PSO (Kennedy and Eberhart, 1995), called "Original Particle Swarm Optimization," OPSO, the set N_t^i contained only two vectors that were the personal best position of the i^{th} particle ($\vec{p}_t^i$) and that of the best position in the whole swarm (known as $\vec{g}_t$), that is, $T_t^i = \{i, \tau_t\}$ where $\tau_t = argmin_{l=\{1,\ldots,n\}}(F(\vec{p}_t^l))$ (the particle τ_t is referred to as the global best particle throughout the review). This topology is called global best topology for PSO. Also, the function $\mu(.)$ in Eq. (3) was defined (Kennedy

[9]Personal best can be a set of best positions, but all PSO types listed in this article use a single personal best.

[10]Learning from a particle refers to gathering information from that particle (either personal best of that particle or its current position).

and Eberhart, 1995) as:

$$\vec{V}^i_{t+1} = \vec{V}^i_t + \varphi_1 R^i_{1t}(\vec{p}^i_t - \vec{x}^i_t) + \varphi_2 R^i_{2t}(\vec{g}_t - \vec{x}^i_t), \tag{5}$$

where φ_1 and φ_2 are two real numbers (usually positive) called cognitive and social weights, also known as acceleration coefficients,[11] and $\vec{p}^i_t$ and $\vec{g}_t$ are the personal best (of particle i) and the global best vectors, respectively, at iteration t. The role of vector factors $CI = \vec{p}^i_t - \vec{x}^i_t$ (Cognitive Influence) and $SI = \vec{g}_t - \vec{x}^i_t$ (Social Influence) is to attract the particles to move toward known quality solutions, that is, personal and global best. R^i_{1t} and R^i_{2t} are two $d \times d$ diagonal matrices[12] (Clerc, 2006; Montes de Oca et al., 2009), where their elements are random numbers distributed uniformly in [0, 1], that is, $U(0, 1)$. Note that matrices R^i_{1t} and R^i_{2t} are generated at each iteration for each particle separately. There is no initialization method for velocity that is superior to other methods in a general case; however, it has been recommended by Engelbrecht (2012) that velocity be initialized to zero.

In 1998, Shi and Eberhart (1998a) introduced a new coefficient ω, called inertia weight, to control the influence of the previous velocity value on the updated velocity.

$$\vec{V}^i_{t+1} = \omega\vec{V}^i_t + \varphi_1 R^i_{1t}(\vec{p}^i_t - \vec{x}^i_t) + \varphi_2 R^i_{2t}(\vec{g}_t - \vec{x}^i_t). \tag{6}$$

Thus, the coefficients ω, φ_1, and φ_2 control influences of the previous velocity, cognitive influence, and social influence on the particle movement. This variant is called "Standard Particle Swarm Optimization," SPSO, throughout the paper. As in OPSO, R^i_{1t} and R^i_{2t} are random diagonal matrices generated for each particle at each iteration. If the random numbers on the diagonal of matrices R^i_{1t} and R^i_{2t} are set to equal values, then these matrices scale the vectors only $(\vec{p}^i_t - \vec{x}^i_t)$ and $(\vec{g}_t - \vec{x}^i_t)$ along their directions. This setup (considering similar random numbers on the diagonal of the random matrices) was introduced as a common error in the implementation of PSO in Clerc (2006, chapter 3, section 2). However, some studies have considered this setup to be another variant of PSO called "Linear Particle Swarm Optimization," LPSO (Bonyadi et al., 2013) which is reviewed later in this review. The gray rectangles in Figure 2 represent the areas in which the terms $\varphi_1 R^i_{1t}(\vec{p}^i_t - \vec{x}^i_t)$ and $\varphi_2 R^i_{2t}(\vec{g}_t - \vec{x}^i_t)$ may be found. Clearly, by applying R^i_{1t} and R^i_{2t} to CI and SI, the resulting vectors can differ from the original ones.

A number of review articles on PSO have been published to date (Banks et al., 2007, 2008; Hu et al., 2004; Parsopoulos and Vrahatis, 2002b; Poli, 2008; Poli, Kennedy, et al., 2007; Song and Gu, 2004). In 2002, seven years after the original PSO was introduced, the first review paper was published (Parsopoulos and Vrahatis, 2002b). The performance of several PSO variants in locating global optimum, jumping out of local optima, dealing with noise, solving multi-objective optimization problems, etc. were reviewed. In addition to the review part of that article, a PSO variant was proposed that used a function stretching approach (Vrahatis et al., 1996) to jump out of local optima.

In 2004, another review article (Hu et al., 2004) on PSO was published that consisted of four main parts: the basic PSO algorithm, discrete PSO, multi-objective PSO, and applications. The first part contained different strategies for determining parameters of the velocity update rule (inertia, cognitive, and social weights). In the second

[11] These two coefficients control the effect of personal and global best vectors on the movement of particles and they play an important role in the convergence of the algorithm. They are usually determined by a practitioner or by analyzing the dynamics of particle movement.

[12] Alternatively, these two random matrices are often considered as two random vectors. In this case, the multiplication of these random vectors by CI and SI is element-wise.

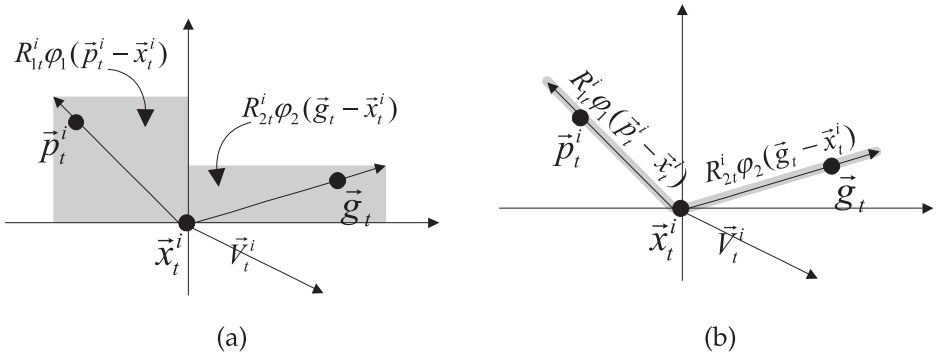

Figure 2: The gray areas are the areas in which $\varphi_1 R^i_{1t}(\vec{p}^i_t - \vec{x}^i_t)$ and $\varphi_2 R^i_{2t}(\vec{g}_t - \vec{x}^i_t)$ may be found, (a) SPSO, (b) LPSO.

part, several PSO variants for solving combinatorial optimization problems were summarized. In the third part, some existing studies related to multi-objective PSO variants were outlined. Some applications of the algorithm in different domains (e.g., feature selection (Agrafiotis and Cedeno, 2002), neural networks (Yao, 1999)) were discussed in the last part.

Another review study was published by Song and Gu (2004). The paper had two main parts: performance improvements and applications. In the first part, several papers which aimed at modifying parameters, increasing diversity, or faster convergence were reviewed. In the second part, applications of PSO to multi-objective optimization, electronics, and training neural networks were discussed.

In 2007, the fourth review paper on PSO was published (Banks et al., 2007). The paper was organized into several sections: a section on the history of the standard PSO was followed by a discussion on discrete and continuous-space PSO variants. The authors then outlined the PSO variants which addressed premature convergence, dealt with dynamic environments, analyzed the swarm behavior with different parameters values, and investigated parallelization of the algorithm. The second part of Banks et al. (2007) was published as a separate article (Banks et al., 2008) and it reviewed hybridization in PSO, combinatorial PSO, constrained optimization using PSO, and applications of PSO. In the same year (2007), another survey paper was published (Poli, Kennedy, et al., 2007). The paper was organized into six parts: i) population dynamic: focused on the original version of PSO, parameters, and analysis of the dynamics of the particles; ii) topology: divided into static and dynamic topologies; iii) specialization: including the discrete space PSO; iv) dynamic environments, etc.; v) theoretical analysis: including deterministic and stochastic models of PSO; and vi) applications of PSO. The paper concluded with a list of open questions.

In 2008, the last survey article was published (Poli, 2008) that discussed the growth of the number of publications in the field of PSO. Results from investigating 700 articles from the IEEE Xplore database showed an expansive growth of publications in the field of PSO at that time. This growth raised a question "What makes PSO so attractive to practitioners?" It was argued that one of the main reasons behind the popularity of PSO was the simplicity of the algorithm that makes it easy to adapt to different applications. Another reason is that the algorithm delivers reasonable results for many different applications. In fact, these characteristics (being simple, reliable, and at the same time

delivering reasonable, though not necessarily the best, results) were found desirable by practitioners who made the algorithm popular (Poli, 2008).

More recently, there have been some other review papers on specific applications of PSO, for example, applications of PSO in data clustering, applications of PSO in dispatching problems (Mahor et al., 2009), applications of PSO in solar photovoltaic systems to name a few. However, they have not been included in this article because their focus was on a specific area only.

3 Identified limitations of PSO

Several limitations of SPSO have been identified so far. The term "limitation" refers to an issue that has been proven to prevent the algorithm from performing well in different aspects of operation such as locating high-quality solutions or being stable. There are two main areas related to the limitations of PSO: *convergence* and *transformation invariance* (Wilke, 2005) (see also Bonyadi and Michalewicz (2014a) for details of these limitations in SPSO). In the following two subsections, articles that have analyzed limitations related to these two topics in PSO are discussed.

3.1 Limitations Related to Convergence

We classify limitations related to convergence in PSO into four groups: *convergence to a point* (also known as *stability*), *patterns of movements*, *convergence to a local optimum*, and *expected first hitting time* (EFHT); these are defined in the following paragraphs.

One of the earliest convergence analyses of stochastic optimization algorithms was published by Matyas (1965) and was followed by Baba (1981) and Solis and Wets (1981). An iterative stochastic optimization algorithm (optimization algorithm for short) is said to converge to a point $\vec{X}$ in a search space in probability (to converge in short) if

$$\forall \varepsilon > 0, \lim_{t \to \infty} P\left(\left|\vec{x}_t - \vec{X}\right| < \varepsilon\right) = 1, \tag{7}$$

where P is the probability measure, $\vec{x}_t$ is a generated solution by the optimization algorithm (a point in the search space) at iteration t, and ε is an arbitrary positive value. There are two possibilities for the point $\vec{X}$:[13]

1. $\vec{X}$ is any point in the search space (including local optima). With this assumption, Eq. (7) refers to the "convergence to a point,"
2. $\vec{X}$ is a local optimum of the objective function in the search space.[14] With this assumption, Eq. (7) refers to the "convergence to a local optimum."

These two types of convergence have been investigated in detail for different optimization algorithms (Birbil et al., 2004; Dasgupta et al., 2009; Matyas, 1965; Solis and Wets, 1981).

Analysis of convergence to a point is usually conducted for an iterative stochastic optimization algorithm to understand whether the sequence of generated solutions produced by the algorithm is convergent. Such analysis sometimes leads to a set of parameter values for the algorithm that guarantee stability. During the convergence

[13]There are also different types of stochastic convergence such as convergence in the n^{th} mean and almost surely convergence. Interested readers are referred to Rudolph (2013) for more detail on different types of convergence.

[14]The point $\vec{c}$ is a local minimum of an objective function f over the search space S if there exists an open interval $I \subseteq S$ such that $\vec{c} \in I$ and $\forall \vec{x} \in I \ f(\vec{c}) \leq f(\vec{x})$.

process, however, the sequence of the solutions found by the algorithm might follow different patterns, for example, high- or low-frequency movements, larger or smaller jumps, faster or slower convergence. These patterns can play an important role in the success of the search process (locating higher quality solutions) as they are closely related to the exploration/exploitation ability of the algorithm (Bonyadi and Michalewicz, 2016).

Analysis of convergence to a local optimum is conducted to understand whether the final solution found by the algorithm is at least a local optimum. Although finding a local optimum is an important (if not essential) property of the algorithm, it is also important to estimate how long it would take for the algorithm to locate that solution (Dorea, 1983). Thus, the expected number of function evaluations to visit a point within an arbitrary vicinity[15] of a local optimum (known as expected first hitting time, EFHT) is analyzed theoretically for optimization methods. This type of analysis, known as EFHT or runtime analysis, has been conducted for evolutionary algorithms (Rudolph, 1998; Eiben and Rudolph, 1999; He and Yao, 2002) (see also Rudolph, 1997 where the convergence rate of evolutionary algorithms, as a related topic to EFHT, has been studied).

Limitations of PSO related to these four topics (convergence to a point, patterns of movement, convergence to a local optimum, and EFHT) are reviewed in the next four subsections.

3.1.1 Convergence to a Point

The velocity vector of particles in SPSO grows to infinity for some values of acceleration and inertia coefficients (Clerc and Kennedy, 2002) (this issue is known as swarm explosion), which causes particles to leave the search space and move to infinity. As the search space is bounded (as defined in Section 2), moving outside of the boundaries is not desirable even if there is a better solution (in terms of the objective value) there. One of the early solutions for this issue was to restrict the value of each dimension of the velocity to a particular interval $[-V_{max}, V_{max}]$ (Helwig et al., 2013; Helwig and Wanka, 2007, 2008; Shi and Eberhart, 1998b). However, this strategy does not prevent swarm explosion in the general case (see Helwig et al., 2013 and Helwig and Wanka, 2007, 2008 for details) because it restricts only the velocity of particles and not the position of particles. Hence, some researchers proposed to also restrict the position of the particles (Helwig and Wanka, 2007). However, even this strategy is not effective because it may restrict the particles to the boundaries of the search space and prevent effective search. A more fundamental solution to the swarm explosion issue can be achieved through analysis of particles' behavior to find out why the sequence of generated solutions might not be convergent (Clerc and Kennedy, 2002; Trelea, 2003; Van Den Bergh, 2002; Poli, 2009; Bonyadi and Michalewicz, 2014d; Cleghorn and Engelbrecht, 2014a)—this is known as stability analysis. The aim of this analysis is to define boundaries for the inertia weight and acceleration coefficients in such a way that the positions of the particles converge to a point in the search space, that is, the position is not divergent. These boundaries are called convergence boundaries throughout this review.[16]

[15] An arbitrary vicinity of a point is defined by a d-dimensional ball around that point with an arbitrary radius.

[16] One should note that determining such boundaries is very helpful for parameter setting purposes in SPSO. The reason is that coefficients that are outside of the convergence boundaries are usually not appropriate for optimization purposes as they cause particles to move unboundedly (recall that

The update rules are sometimes simplified for stability analysis purposes. Velocity and position update rules are analyzed for an arbitrary particle i in a one-dimensional space (we use the notations p and g, instead of $\vec{p}$ and $\vec{g}$, when we analyze $\vec{p}$ and $\vec{g}$ in a one-dimensional space). Also, $\vec{p}_t^i$ and $\vec{g}_t$ are assumed to be steady during the run. Further, the topology of the swarm is ignored in the analysis.

One might argue that the convergence boundaries found under these simplifications are not valid as the simplified system might not exactly reflect the behavior of the original system. However, it should be noted that, in these analyses, the behavior of particles is investigated while they are searching for a new personal best. Considering the one-dimensional space for analysis of SPSO is reasonable because all calculations (including generation of the random values on the diagonal of R_{1t}^i) are done in each dimension independently; hence, all analyses in the one-dimensional case are also generalizable to multidimensional cases as well (Poli, 2009). In addition, validity of the convergence boundaries found when global and personal bests are steady and the topology is ignored were studied by Cleghorn and Engelbrecht (2014b, 2014c, 2015) and Bonyadi and Michalewicz (2016) experimentally. It was found that, although these conditions simplify the update rules, the calculated convergence boundaries under these conditions are in agreement with those found under general conditions determined through experiments. In addition, a recent theoretical study by Liu (2014) showed that the convergence boundaries for the global best particle found under this simplification do not change if $\vec{p}_t^i$ is allowed to be updated (under some conditions) during the run (this paper is reviewed in more detail later in this survey).

Three different types of stability analysis can be found in literature: deterministic model stability analysis, first-order stability analysis, and second-order stability analysis. Deterministic model stability analysis excludes the stochastic component in particles and assumes that the particles are moved through a deterministic formulation (see Clerc, 1999 and Clerc and Kennedy, 2002 for examples). First-order stability analysis studies the expectation of the position of particles (see Trelea, 2003, Van den Bergh and Engelbrecht, 2006, and Cleghorn and Engelbrecht, 2014a for examples) to ensure that this expectation converges. The second-order stability analysis studies the variance of the position of particles (see Jiang et al., 2007b, Poli, 2009, and Liu, 2014 for examples) to ensure this variance converges to zero (the convergence of the variance of the particle position is a necessary condition for second-order stability).

Perhaps the first study that considered theoretical analysis of the trajectory of particles in PSO was conducted by Ozcan and Mohan (1999). Based on a deterministic model for particles (the stochastic components were ignored), the trajectory and step size of movement for particles were analyzed. It was found that, for OPSO, the trajectory of a particle is of the form of a random sinus wave when $\varphi_1 + \varphi_1 < 4$. Although no parameter analysis or convergence boundaries were introduced in that paper, the findings in that paper formed a basis for further analysis.

One of the earliest attempts to analyze the convergence behavior of SPSO to find convergence boundaries[17] was done by Clerc and Kennedy (2002). In that paper, in order to simplify the formulation of update rules, stochastic components (r_{1t}^i and r_{2t}^i)[18]

---

the search space is bounded). Hence, searching for a combination of coefficients that results in a good performance of the algorithm is focused on a smaller boundary.

[17]Van den Bergh (2002) also analyzed the convergence behavior of SPSO in his thesis; however, the related paper to that thesis was published in 2006, which is reviewed later in this survey.

[18]We use the notation r for random scalars as each dimension is updated independently.

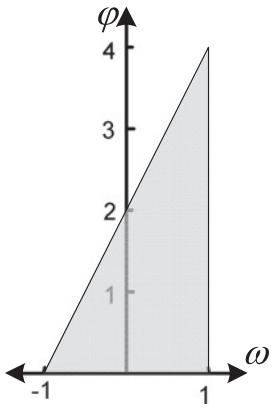

Figure 3: According to Trelea (2003), expectation of particle positions converge to its equilibrium if and only if the value of the parameters are set in such a way that they lie inside the gray triangle (convergence boundaries).

were omitted from the system (deterministic model analysis). It was proven that, by using this simplified model, particle positions converge to a stable point if

$$\chi = \frac{2k}{|2 - c - \sqrt{c^2 - 4c}|}, \tag{8}$$

where χ (a coefficient known as constriction factor) is equal to ω (inertia weight), $c_1 + c_2 = c > 4$, $c_1 = \frac{\varphi_1}{\chi}$, $c_2 = \frac{\varphi_2}{\chi}$, and k is a value in the interval $(0, 1]$ (usually set to 1 (Clerc and Kennedy, 2002)). With these settings the value of χ is in the interval $(0, 1]$. The value of k controls the speed of convergence to a fixed point, that is, the larger the value of k is, the slower the convergence to the fixed point will be. According to these coefficients, the velocity update rule is written as

$$\vec{V}^i_{t+1} = \chi(\vec{V}^i_t + c_1 R_{1t}(\vec{p}^i_t - \vec{x}^i_t) + c_2 R_{2t}(\vec{g}_t - \vec{x}^i_t)), \tag{9}$$

where χ is set by Eq. 8, $c_1 + c_2 > 4$, and usually $c_1 = c_2 = 2.05$ that results in $\chi = 0.7298$ (Clerc and Kennedy, 2002). By using these settings the value of each dimension of the velocity vector converges to zero and, consequently, velocity does not grow to infinity. The PSO variant that uses velocity update rule in Eq. (9) is called "Constriction Coefficient Particle Swarm Optimization," CCPSO. The stable point where each particle i converges to is a point between p and g (namely $\frac{c_2 g + c_1 p}{c_1 + c_2}$).

The stability of particles in SPSO was also analyzed by Trelea (2003). In that study, the random components were replaced by their expected values ($r^i_{1t} = r^i_{2t} = 0.5$), which enabled first-order stability analysis. The parameters of the velocity update rule were analyzed to find out how they should be set to guarantee particles settle to their equilibrium. Obviously, $\vec{V}^i_t = 0$ and $\vec{x}^i_t = \vec{p}^i_t = \vec{g}_t$ at the equilibrium point. It was proven that the expected value of the position of each particle settles to its equilibrium (that is $\frac{\varphi_2 g + \varphi_1 p}{\varphi_1 + \varphi_2}$) if and only if $\omega < 1$, $\varphi > 0$, and $2\omega - \varphi + 2 > 0$ where $\varphi = \frac{\varphi_1 + \varphi_2}{2}$. Figure 3 shows this relationship between ω and φ. The parameter setting procedure by Trelea (2003) showed that $\omega = 0.6$ and $\varphi_1 = \varphi_2 = 1.7$ results in a good performance of the algorithm.

Results reported by Trelea (2003), including convergence boundaries and convergence to a point between p and g, were also confirmed by Van den Bergh and Engelbrecht (2006) (another first-order stability analysis). However, in the latter paper, the speed of

convergence was investigated in more depth (see Section 3.1.2 for more detail). Subsequently, another first-order stability analysis was conducted by Campana et al. (2010) where the findings in Trelea (2003) were also confirmed.

The analysis of convergence to a fixed point was conducted, from a different point of view, in Jiang et al. (2007b). It was proven that the expectation of the position of each particle converges to a point (first-order stability analysis) between the personal and global best vector ($\frac{\varphi_2 g+\varphi_1 p}{\varphi_1+\varphi_2}$) if and only if $0 \leq \omega < 1$ and $0 < \varphi < 4(1+\omega)$ where $\varphi = \varphi_1 + \varphi_2$. However, it was pointed out that the first-order stability is not enough to guarantee convergence of particles and second-order stability should also be guaranteed. In fact, to guarantee stability, not only the expected position of particles should converge to a fixed value but also the variance of the position should converge to 0 (see also Poli, Brattonx, 2007, Poli, 2009, and Bonyadi and Michalewicz, in press). It was proven by Jiang et al. (2007b) that, setting the coefficients to satisfy $\frac{5\varphi-\sqrt{25\varphi^2-336\varphi+576}}{24} < \omega < \frac{5\varphi+\sqrt{25\varphi^2-336\varphi+576}}{24}$, where $\varphi = \varphi_1 = \varphi_2$, guarantees the convergence of the variance of the position of particles to a fixed value (a necessary condition for variance of position to converge). Note that this inequality can be simplified to $\varphi < \frac{12(\omega^2-1)}{5\omega-7}$. This analysis was slightly modified due to a small error and the corrected version was presented in Jiang et al. (2007a). This study also recommended $\omega = 0.715$ and $\varphi_1 = \varphi_2 = 1.7$ for experimental purposes. The authors argued that these coefficient values result in a higher variance during the run that improves the exploration ability of particles.

The expected value and the standard deviation of the sequence of generated positions by SPSO were also analyzed in Poli (2009) and Poli, Brattonx et al. (2007) and the results found by Jiang et al. (2007a) were confirmed. It was proven that convergence of variance to a fixed point (a necessary condition for second-order stability) requires $\varphi < \frac{12(\omega^2-1)}{5\omega-7}$ where $\varphi = \varphi_1 = \varphi_2$, that is, the same as what was found by Jiang et al. (2007a). Also, Poli (2009) proved that the variance of positions converges to $h(\varphi_1, \varphi_2, \omega)|g - p|$ where $h(.,.,.)$ is a function of inertia weight and acceleration coefficients (see Poli, 2009 for details of this function). Hence, if $h(\varphi_1, \varphi_2, \omega) \neq 0$ is guaranteed then particles do not stop moving (non-zero variance) until $p = g$. Therefore, the algorithm is second-order stable only if $p = g$.

The first and second-order stability were investigated by García-Gonzalo and Fernández-Martínez (2014) for SPSO when a generic distribution for the inertia weight and acceleration coefficients was considered, that is, it was assumed that the inertia weight is a random variable from an arbitrary probability distribution with the expected value μ_ω and the variance σ_ω^2 and ϕ_1 and ϕ_2 (ϕ_1 replaces $\varphi_1 R_{1,t}^i$ and ϕ_2 replaces $\varphi_2 R_{2,t}^i$ in Eq. 6) are also random variables from an arbitrary probability distribution with the expected values μ_{ϕ_1} and μ_{ϕ_2} and the variances $\sigma_{\phi_1}^2$ and $\sigma_{\phi_2}^2$. It was proven that SPSO is first-order stable if and only if $-1 < \mu_\omega < 1$ and $0 < \mu_\phi < 2(\mu_\omega + 1)$ where $\phi = \phi_1 + \phi_2$. Also, a necessary condition for the second-order stability is that $-a < \mu_\omega < a$ and $0 < \mu_\phi < b$ where $a = \frac{1}{\sqrt{c^2+1}}$, $b = \frac{2(1-(1+c)\mu_\omega^2)}{1+d^2+(d^2-1)\mu_\omega}$, $c = \sigma_\omega/\mu_\omega$, and $d = \sigma_\phi/\mu_\phi$. It was shown that the regions found to guarantee second-order stability are embedded within the regions that guarantee first-order stability; that is, if a SPSO is second-order stable then it is also first-order stable.

Recently, some studies have considered more general assumptions to find the convergence boundaries under more realistic conditions. For example, Cleghorn and Engelbrecht (2014a) assumed that the personal best of particles and the global best of the swarm are allowed to move and can occupy an arbitrarily large finite number of

unique positions. The main finding of that study was that SPSO is first-order stable if and only if $-1 < \omega < 1$ and $0 < \varphi < 4(1+\omega)$ where $\varphi = \varphi_1 + \varphi_2 < 4$ under this more general assumption. This is indeed the same as what was found by Trelea (2003) and other previous studies through first-order stability analysis. Cleghorn and Engelbrecht (2014a) also showed that the topology does not affect the convergence boundaries, but it might affect the speed of convergence/divergence.

The second-order stability of SPSO was also investigated by Liu (2014). It was assumed that the personal best needs to remain unchanged at least for a limited number of iterations (3 in that study) that is a weaker assumption comparing to what was assumed by previous studies (i.e., the personal best need to always remain constant). The theoretical analysis in that paper proved that the convergence boundaries found by Jiang et al. (2007b) and Poli (2009) are valid under this assumption for the global best particle. Also, Liu (2014) proved that the found regions serve as a necessary and sufficient condition for second-order stability of the global best particle. This study recommended $\omega = 0.42$ and $\varphi_1 = \varphi_2 = 1.55$ based on experiments on an extensive set of benchmark functions.

Analysis of stability was also done for a variant of PSO called "Standard PSO 2011," SPSO2011,[19] by Bonyadi and Michalewicz (2014d, 2016). The velocity update rule for SPSO2011 is written as:

$$\vec{V}_{t+1}^{\,i} = \omega \vec{V}_t^{\,i} + \mathcal{H}_i(\vec{G}_t^{\,i}, |\vec{G}_t^{\,i} - \vec{x}_t^{\,i}|) - \vec{x}_t^{\,i}, \tag{10}$$

where $\mathcal{H}_i(\vec{G}_t^{\,i}, |\vec{G}_t^{\,i} - \vec{x}_t^{\,i}|)$ is a hyperspherical distribution with the center $\vec{G}_t^{\,i}$ and radius $|\vec{G}_t^{\,i} - \vec{x}_t^{\,i}|$ given that $\vec{G}_t^{\,i} = \frac{\vec{L}_t^{\,i} + \vec{P}_t^{\,i} + \vec{x}_t^{\,i}}{3}$, $\vec{P}_t^{\,i} = \vec{x}_t^{\,i} + \varphi_1(\vec{p}_t^{\,i} - \vec{x}_t^{\,i})$, and $\vec{L}_t^{\,i} = \vec{x}_t^{\,i} + \varphi_2(\vec{l}_t^{\,i} - \vec{x}_t^{\,i})$ ($\vec{l}_t^{\,i}$ is the best personal best among the particles in $T_t^{\,i}$) (Clerc, 2011). The analyses of convergence conducted for previous PSO variants are not applicable to SPSO2011. The reason is that updating velocities and positions in all previous PSO variants were done dimension by dimension which enabled researchers to study particles in a one-dimensional space. However, the calculation of $\vec{V}_t^{\,i}$ in SPSO2011 involves generating random points using a hyperspherical distribution with a variance that is dependent on the distance between $\vec{G}_t^{\,i}$ and $\vec{x}_t^{\,i}$. In order to repeat the dimension-wise analysis (similar to that of done for SPSO) in SPSO2011, one needs to decompose the random-point generation procedure done by the hyperspherical distribution into dimension-wise calculations which might need further effort. Hence, the stability analysis for SPSO2011 was done experimentally in Bonyadi and Michalewicz (2014d, 2016) where it was shown that convergence boundaries for particles in SPSO2011 are dependent on the number of dimensions and these boundaries are different from that of other PSO variants (e.g., SPSO). Thus, good choices for coefficient values in previous PSO variants do not necessarily lead to a good performance of SPSO2011.[20] Bonyadi and Michalewicz (2016) also experimentally showed that the convergence boundaries that guarantee second-order stability are not affected even if the global best and personal bests are updated (through a uniform random distribution) during the runs for both SPSO and SPSO2011.

Stability of a stochastic recurrence relation that formulates the position update rule of particles in a wide range of PSO variants (including SPSO) was studied by Bonyadi and Michalewicz (in press). In order to weaken the stagnation assumption, it was assumed that the global and personal bests in that relation are updated through

[19]This variant also follows the general form in Eq. (2) for velocity updating.

[20]The shape of the convergence boundaries for this variant is very sensitive to the procedure to generate the hyperspherical distribution. See Bonyadi and Michalewicz (2016) for further details.

an arbitrary random distribution. It was proven that the necessary and sufficient conditions to guarantee convergence of expectation and variance of generated positions by that relation are independent of the mean and variance of the random distribution by which the global and personal bests are updated. Hence, convergence boundaries for parameters that guarantee the convergence of expectation and variance of the positions trajectory are not affected by the mean and variance of that distribution.

Summary: Experiments showed that the velocity vector grows to infinity for all particles in the swarm for some values of acceleration coefficients and inertia weight (known as swarm explosion). Some studies investigated this issue from the theoretical point of view to find the reasons behind this divergence. Three approaches were considered: analysis of deterministic models, for example, Clerc and Kennedy (2002), first-order stability of particles (Trelea, 2003; Van den Bergh and Engelbrecht, 2006; Cleghorn and Engelbrecht, 2014a), and second-order stability (Jiang et al., 2007a; Poli, 2009; Liu, 2014; García-Gonzalo and Fernández-Martínez, 2014). Although some of these analyses were conducted under simplified conditions (e.g., personal best and global best are not updated), further studies (Cleghorn and Engelbrecht, 2014b, 2014c; Bonyadi and Michalewicz, 2016) experimentally showed that the convergence boundaries found under these simplified conditions are in agreement with the cases with no simplification, especially for the studies which considered second-order stability. Articles that studied the first-order stability of particles found that the point each particle converges to is a point on the line segment that connects its personal best and the global best vectors (Trelea, 2003; Van den Bergh and Engelbrecht, 2006). Similar analyses for the first-order stability were also conducted under more general conditions by Cleghorn and Engelbrecht (2014a) where it was assumed that the personal best and global best vectors can occupy an arbitrarily large finite number of unique locations in the search space. In addition to first-order stability, analyses conducted by Jiang et al. (2007b) and Poli (2009) showed that particles do not stop moving (convergence of the variance of the particles' positions, i.e., second-order stability) on this line segment until their personal bests coincide with the global best of the swarm. In addition, second-order stability was studied later by Liu (2014) under more general conditions for the global best particle. It was concluded that the convergence boundaries found under simplified assumption (personal best and global best are not updated) considered in earlier studies (for example in Jiang et al., 2007b and Poli, 2009) through second-order stability analysis is also valid under this more general assumption for the global best particle. In addition, it was proven that these convergence boundaries are both necessary and sufficient for second-order stability. As reported by Liu (2014), the combination ($\omega = 0.45, \varphi = 1.55$) works better than other previously used coefficient values on a large benchmark of test functions. Further, as previous studies considered only a uniform distribution for random components, García-Gonzalo and Fernández-Martínez (2014) studied the convergence boundaries for both first- and second-order stability for an arbitrary distribution assuming that the personal best and global best are constant.

Clearly, investigation of the convergence of variance and second-order stability under generic assumptions (the personal best and global best vectors are updated during the run) is of great value. This analysis for different topologies can be another area for further research. Although there have been many studies that have analyzed the stability of particles, these analyses have been limited to SPSO and have been only recently done for SPSO2011. There are many other PSO variants that are substantially different from SPSO and their convergence behavior is unknown (such as the PSO variants described in Bonyadi, Michalewicz, and Li (2014) and Wilke et al., 2007). Thus, it would be

important to conduct convergence analysis for these variants as well. Another potential direction for further studies is to investigate the differences/similarities between PSO variants and Evolutionary Strategy (ES) from theoretical point of view, for example, how are the dynamics of the two methods different and how do these differences, if any, impact the search ability of these methods? Angeline (1998) conducted such a comparison from philosophical and experimental points of view; however, there have been many theoretical investigations on both algorithms (ES and PSO) since then that enable researchers to conduct more in-depth theoretical comparisons between these two algorithms.

3.1.2 Patterns of Movement

If the coefficients of SPSO are selected in a way that they are inside the convergence boundaries, then the sequence of the positions of particles is not divergent. During the run, however, particles oscillate with different patterns around their equilibrium point until they converge (Trelea, 2003; Bonyadi and Michalewicz, 2016). The difference between these patterns is a consequence of picking different values for coefficients that play an important role on the performance of the algorithm. For example, a particle that moves smoothly in the search space can potentially be more effective in the exploitation phase than a particle that jumps all over the search space. Hence, investigation of these patterns and calculation of corresponding coefficients which exhibit different patterns can be helpful for practitioners. In addition, the speed of convergence, that is, how fast the particles positions approach the equilibrium point, can be of importance to determine appropriate coefficient values.

Trelea (2003) investigated the roots of the characteristic equation of the expected position of particles (Eq. 11) for each dimension to categorize different oscillations particles' positions might exhibit:

$$E(x_{t+1}) + (\varphi - \omega - 1)E(x_t) + \omega E(x_{t-1}) = \varphi P, \tag{11}$$

where $E(.)$ is the expectation operator, $\varphi = \frac{\varphi_1+\varphi_2}{2}$, and $P = \frac{\varphi_1}{\varphi_1+\varphi_2} p + \frac{\varphi_2}{\varphi_1+\varphi_2} g$. The pattern of generated points by this equation (i.e., expectation of positions) are:

- **harmonic** if the imaginary component of both roots of the characteristic equation is non-zero and the real component of the roots is positive,
- **zigzagging** if the imaginary component of both roots of the characteristic equation is zero and the real component of at least one of the roots is negative,
- **harmonic-zigzagging** if the imaginary component of both roots of the characteristic equation is non-zero and the real component of both roots is negative,
- **non-oscillatory** if the imaginary component of both roots of the characteristic equation is zero and the real component of both roots is positive,

Also, experiments showed that convergence is faster if the values of ω and φ are closer to the "center" (close to the point (0, 1)) of the triangle in Figure 3. Thus, particles spend more iterations for exploitation with this setting. Similar analysis for the behavior of particles before convergence was conducted by Campana et al. (2010) where the same oscillation patterns were observed.

The speed of convergence of the expectation of positions was investigated by Van den Bergh and Engelbrecht (2006) who proved that the speed of convergence is

directly related to $c = \max\{||\gamma_1||, ||\gamma_2||\}$ where $\gamma_1 = \frac{1+\omega-\varphi_1-\varphi_2+\alpha}{2}$, $\gamma_2 = \frac{1+\omega-\varphi_1-\varphi_2-\alpha}{2}$ and $\alpha = \sqrt{(1+\omega-\varphi_1-\varphi_2)^2 - 4\omega}$; that is, the larger the value of c, the faster the expectation of the position of the particle converges to its equilibrium. It was also proven that the expected value of the particle's positions converge to a fixed point (namely $\frac{\varphi_2 g+\varphi_1 p}{\varphi_1+\varphi_2}$) if and only if $c < 1$.

Bonyadi and Michalewicz (2016) investigated the behavior of particles before convergence through an experimental approach. They considered the generated sequence of particle positions as a time series and they used frequency domain analysis to understand how particles oscillate during the run. They categorized the oscillation patterns into four groups based on the maximum frequency of oscillation (low, low-mid, mid-high, and high frequencies), corresponding to the patterns introduced by Trelea (2003). They showed that the results found by their experimental approach is very similar to what Trelea (2003) has found. Hence, they used the same approach to analyze the oscillation of particles in SPSO2011. They found that the boundaries of coefficients corresponding to different patterns of oscillation for SPSO2011 are different from those of SPSO. Their experiments also showed that these boundaries are not sensitive to the number of dimensions.

Summary: The particle position oscillates during the run until it converges to a point. This oscillation, however, might show different patterns; for example, the position might jump from one place to another, it might be smooth through the search space. The pattern of oscillations as well as the rates of oscillations are affected by changing the values for coefficients. Hence, some researchers studied these patterns and rates to find which coefficients values correspond to different behaviors (Trelea, 2003; Van den Bergh and Engelbrecht, 2006; Campana et al., 2010; Bonyadi and Michalewicz, 2016). For example, Trelea (2003) found that four groups of oscillation patterns can be observed in particles' expectation of positions that are zigzagging, harmonic, a combination of the two, and non-oscillatory. Trelea (2003) found the coefficients' values corresponding to each of these patterns. These findings were also confirmed by Campana et al. (2010) and experimentally tested by Bonyadi and Michalewicz (2016) through analysis of frequency. The speed of convergence of expectation was also investigated by Van den Bergh and Engelbrecht (2006) and conditions for coefficients corresponding to the rate of convergence of expectation were introduced.

As a potential future direction, it would be valuable to study benefits of different convergence behaviors, for example, zigzagging and harmonic (see Trelea, 2003), on different types of landscapes (neutral, rugged, etc.). Such study can potentially lead to an adaptive approach that alters the movement patterns of the particles according to the type of landscape encountered during the run (see also Bonyadi and Michalewicz, 2016 for a discussion on this topic). Note that these patterns are closely related to the explorative/exploitative ability of the algorithm. More research can also be conducted on the speed of convergence to enable practitioners to choose between fast and potentially low-quality solutions (more exploitation and less exploration) and slow but potentially high-quality solutions (more exploration and less exploitation); see Van den Bergh and Engelbrecht (2006), for example. In addition, there has not been any theoretical work on the speed of the convergence of variance. In fact, the value of the variance that the particles' positions converge to and the speed of convergence to that value are related to the explorative and exploitative behavior of particles (particles are more exploitative for smaller variance of position), so that idea can be considered as another potential future work. Another potential future research direction is to study the covariance of particles'

movement to understand how correlated each sample is to the previous ones and how this correlation is affected by the coefficients' values. Also, the trajectory of movement of each particle follows a particular pattern that may result in different total distance covered. Because this distance is an important factor for physical systems which use PSO method for search (e.g., seeking the source of a signal using a swarm of robots; Zou et al., 2015), one can study the expectation of the total distance that would be covered through different trajectory patterns and then minimize this expectation by choosing appropriate coefficient values.

3.1.3 Convergence to a Local Optimum

Convergence to a point ensures that a particle settles to its equilibrium and does not move to infinity. It was proven that all particles converge to $\vec{x}_t^i = \vec{p}_t^i = \vec{g}_t$ and $\vec{V}_t^i = 0$ (stagnation) if coefficients (acceleration and inertia) are in the convergence boundaries. However, for any value of coefficients, there is no guarantee that the point that the particles converged to is a local optimum. A particle is said to be *locally convergent* if

$$\forall \varepsilon > 0, \lim_{t \to \infty} P(|\vec{p}_t^i - \vec{X}| < \varepsilon) = 1$$

where $\vec{X}$ is a local optimum (Bonyadi and Michalewicz, 2014a). An alternative for this definition (Van den Bergh and Engelbrecht, 2010) is, a PSO algorithm is said to be locally convergent if

$$\forall \varepsilon > 0, \lim_{t \to \infty} P(|\vec{g}_t - \vec{X}| < \varepsilon) = 1.$$

Local convergence was first analyzed by Van den Bergh and Engelbrecht (2002) for SPSO. It was observed that SPSO does not perform well when the swarm size is too small. This is called the swarm size issue throughout the paper. The explanation given in Van den Bergh and Engelbrecht (2002) was that particles in a SPSO with smaller swarm size (e.g., 2) have larger probability to stagnate. A variant of SPSO called "Guaranteed Convergent Particle Swarm Optimization," GCPSO (Van den Bergh and Engelbrecht, 2002), was proposed in which the local convergence issue was addressed. In GCPSO, a new velocity update rule was introduced for the global best particle τ_t. Eq. (12) shows the velocity update rule for GCPSO.

$$\vec{V}_{t+1}^i = \begin{cases} -\vec{x}_t^i + \vec{g}_t + \omega \vec{V}_t^i + \vec{\rho} & \text{if } i = \tau_t \\ \omega \vec{V}_t^i + \varphi_1 R_{1t}^i (\vec{p}_t^i - \vec{x}_t^i) + \varphi_2 R_{2t}^i (\vec{g}_t - \vec{x}_t^i) & \text{otherwise.} \end{cases} \tag{12}$$

where ρ^j (the value of the j^{th} dimension of $\vec{\rho}$) is a random value determined through an adaptive approach that applies a perturbation to the velocity vector. At each iteration of GCPSO, a random location in a hyper-rectangle with the j^{th} side length equal to ρ^j is generated. This random location is moved from $\vec{g}_t$ by the vector $\omega \vec{V}_t^i$ to form the new position for the particle τ_t (global best particle). Figure 4 shows the hyper-rectangle and the area within which $\vec{x}_{t+1}^{\tau_t}$ might be located. The velocity update rule for the particle τ_t in GCPSO forces that particle to always move (implying a non-zero value for $\vec{V}_{t+1}^{\tau_t}$ for any t). It was proven (Van den Bergh and Engelbrecht, 2010) that SPSO is not locally convergent while GCPSO is locally convergent. Also, GCPSO could provide acceptable results with small swarm size (in reported experiments, the number of particles was set to 2). Stagnation and its relation to small swarm size was also investigated in Bonyadi et al. (2013) and Bonyadi and Michalewicz (2014b), which that are reviewed in Section 5 as these methods were used to deal with COPs.

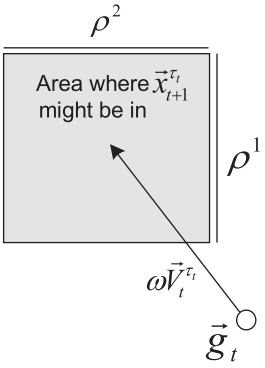

Figure 4: In GCPSO $\vec{x}_{t+1}^{\tau_t}$ is taken randomly from the gray area while all other particles follow the original position and velocity update rule.

The local convergence issue for SPSO was also discussed in Schmitt and Wanka (2013), where it was proven that SPSO is locally convergent for one-dimensional problems; however, it is not locally convergent when the number of dimensions is larger than one. Their main idea to address the local convergence issue was to detect stagnation and then to avoid it by introducing a perturbation operator. The value $\beta_t^i = ||\vec{V}_t^i|| + ||\vec{g}_t - \vec{x}_t^i||$ was used to detect the stagnation of particle i. If $\beta_t^i < \delta$ (where δ is a small positive value), then the particle i is expected to stagnate. In such a case the velocity vector of particle i was regenerated randomly within a hyper-rectangle with the side length δ. It was proven that by using this approach for the velocity vector, the algorithm is locally convergent.

The issue of local convergence was also investigated by Bonyadi and Michalewicz (2014a) for SPSO. The formulation for SPSO was revised in that paper as follows:

$$\vec{V}_{t+1}^i = \omega\vec{V}_t^i + \sum_{k\in T_t^i} \varphi_k r_{kt}^i (m_{kt}^i(\vec{p}_t^k) - \vec{x}_t^i), \tag{13}$$

where the operator $m_{kt}^i : \mathbb{R}^d \to \mathbb{R}^d$ applies perturbation on its inputs and $r_{kt}^i \in \mathbb{R}$ is a random scalar (rather than a random diagonal matrix) in the interval [0, 1]. This variant was called "Locally convergent Rotationally invariant Particle Swarm Optimization," LcRiPSO (see Section 3.2 for details about rotation invariance property of the algorithm). It was proven that, in LcRiPSO, if the operator m is designed in such a way that

$$\forall\vec{y} \in S, \exists A_y \subseteq S, \text{ such that } \forall\vec{z} \in A_y, \forall\delta > 0, P(|m(\vec{y}) - \vec{z}| < \delta) > 0, \tag{14}$$

then the algorithm is locally convergent, where P is the probability measure and A is an open set; that is, for any input vector $\vec{y}$ in the search space, there exists an open set A which contains $\vec{y}$ and $m(\vec{y})$ can be located anywhere in A. It was proven that a normal distribution with the mean equal to $\vec{y}$ and a non-zero variance is an instance of m. It was proposed to use a proportion of the last non-zero distance between $\vec{x}_i$ and $\vec{p}_i$ for this variance. The algorithm was applied to 20 benchmark test functions (CEC05 and CEC08 benchmarks) and its results were compared with other PSO variants. Experiments (based on the Wilcoxon test) showed that LcRiPSO provides acceptable results with small swarm sizes. Also, as all particles in this algorithm are locally convergent and the algorithm performs well with small swarm size, this algorithm can be a good candidate to locate niches in a search space—see Bonyadi and Michalewicz (2014b) for details.

Bonyadi and Michalewicz (2016) investigated the local convergence property of SPSO2011. They introduced a recursive relation that represented a class of PSO methods (including SPSO and SPSO2011) and studied conditions to guarantee local convergence for that relation. Accordingly, they modified SPSO2011 in a way that the local convergence was guaranteed for that algorithm.

Summary: Selecting values of coefficients within the convergence boundaries prevents particles from moving unboundedly. However, there is no guarantee that the point a particle converges to is a local optimum. If the personal best of a particle is guaranteed to converge to a local optimum, then it is said that the particle is locally convergent. If a PSO method guarantees convergence of $\vec{g}_t$ to a local optimum, then that PSO method is said to be locally convergent. It has been proven that SPSO is not locally convergent. Also, convergence to a non-optimal solution takes place more frequently if the swarm size is small. To overcome the local convergence issue in SPSO, a mutation operator, for example, replacing global/personal best vectors by a randomly selected point around the global best vector (Bonyadi and Michalewicz, 2014a; Van den Bergh and Engelbrecht, 2010) or regeneration of velocity vector (Schmitt and Wanka, 2013) may be applied to prevent particles from stagnating.

Although a mutation operator addresses the local convergence issue, it might slow down the search process (see Section 3.1.4). One of the issues with these mutation operators is that known information from the search space, such as potentially good directions toward better solutions, is ignored. Instead of considering a uniform distribution in a hyper-rectangle for picking the random point (mutation operator), one can consider an adaptive normal distribution, potentially stretched toward specific directions that are known to be good, to speed up convergence to local optima similar to the approaches for ES (Beyer and Schwefel, 2002; Hansen, 2000, 2006) that introduced CMA-ES. Note that a simple version of adaptive mutation based on normal distribution was proposed also in Bonyadi and Michalewicz (2014b)—this paper is discussed later in Section 5. However, that simple adaptive approach only considered generating the normal distribution for each dimension separately that ignored the correlation between different dimensions (i.e., only the diagonal of the covariance matrix).

3.1.4 Expected First Hitting Time (EFHT)

In 2009, the lack of EFHT (runtime) analysis for PSO was pointed out by Witt (2009). The author argued that most convergence analyses for SPSO (including Clerc and Kennedy, 2002, Jiang et al., 2007b, and Trelea, 2003) have been limited to the concept of convergence to an attractor and not to a particular optimum. As GCPSO was the only PSO variant that was proven to be locally convergent by that time, the EFHT analysis was conducted for this variant. Analyses showed that the EFHT of GCPSO on the sphere function is asymptotically the same as a simple (1+1) ES—see Jägersküpper (2008) for analysis of the EFHT for randomized direct search methods with an isotropic sampling, including (1+1) ES. However, GCPSO is not rotation invariant while (1+1) ES is. The author concluded that, "Still, the aim of PSO is to solve optimisation problems, and empirically, it fulfils this requirement quite well. In this respect, PSO is a heuristic which presently eludes a theoretical treatment of its practical success."

The analysis of EFHT for SPSO was also conducted in Lehre and Witt (2013) where the algorithm was applied to a sphere function. It was proven that EFHT for SPSO is potentially infinite even when it is applied to a one dimensional sphere. It was shown that, in some situations that occur with non-zero probability, the algorithm cannot locate the optimal solution even for a one dimensional sphere. Note that the setting

of the parameters in that paper was different from that of Schmitt and Wanka (2013), where it was proven that SPSO is locally convergent for one-dimensional problems. A variant of PSO, called "Noisy Particle Swarm Optimization," NPSO, was introduced (Lehre and Witt, 2013) in which a uniform randomly generated vector in a hyper-rectangle with the side length $\delta > 0$ was added to the velocity vector of each particle. The authors proved that EFHT for NPSO to optimize a sphere function is finite; that is, NPSO converges to a local optimum of the sphere function in a finite number of function evaluations. It was concluded that more efforts are needed to study the EFHT of SPSO and understand how the algorithm works. A similar analysis was also done by Bonyadi and Michalewicz (2014d) for SPSO2011, where it was proven that, with specific setting of parameters, EFHT of the algorithm for any one dimensional optimization function is infinite; for example, SPSO2011 does not guarantee to converge to a local optimum under assumptions that take place with non-zero probability.

Summary: Although addressing the local convergence issue (discussed in Section 3.1.3) is of a great significance, it is also important to understand what is the expected time (average number of function evaluations) to hit a point within a vicinity of the local optimum (Dorea, 1983). There have not been many attempts to analyze/improve EFHT for PSO variants. A potential reason behind the lack of EFHT analysis in PSO variants is that there are not many PSO variants that have been proven to be locally convergent while local convergence is a prerequisite for EFHT analysis in an algorithm. It has been shown that the performance of one of the locally convergent PSO variants is almost the same as that of a simple ES method in terms of EFHT on a simple problem (i.e., sphere function). Thus, it would be worthwhile to put some effort on analysis of the EFHT of PSO variants that are locally convergent and to introduce modifications to improve the EFHT for those variants.

3.2 Limitations Related to Transformation Invariance

An algorithm $\mathcal{A}$ is invariant under the transformation T, $T : \mathbb{R}^d \to \mathbb{R}^d$, if the performance of $\mathcal{A}$ on any objective function $F(\vec{x})$, $F : \mathbb{R}^d \to \mathbb{R}$, is the same as the performance of $\mathcal{A}$ on $F(T(\vec{x}))$, given that the initial state is "properly" set in both cases. More formally:

Let $\mathcal{T} \subset \{U : \mathbb{R}^d \to \mathbb{R}^d\}$ be a set of transformations (as a subset of all possible transformations U), Y is the state space of the search algorithm and $\mathcal{A}_F : Y \to Y$ is one iteration of the algorithm under the objective function F. The algorithm $\mathcal{A}_F$ is invariant under $\mathcal{T}$ if for all $T \in \mathcal{T}$ there exists a bijection $J_T : Y \to Y$ such that for all $F : \mathbb{R}^d \to \mathbb{R}$ and $y \in Y$

$$J_T \; o \; \mathcal{A}_{F \; o \; T}(y) = \mathcal{A}_F \; o \; J_T(y).$$

For stochastic algorithms, this equality needs to hold almost surely or in distribution (see Hansen et al., 2011 for complete discussion). Intuitively, this definition means that the performance of the algorithm is independent of how the coordinate axes, that is, the frame, are placed on the search space.

In addition to invariance, it is also expected that the optimization algorithm is as independent as possible from the initial state (so-called adaptivity). Therefore, in order to study the transformation invariance property of optimization algorithms, some researchers took the adaptivity for granted and assumed that the initial state is also transformed using the same transformation that the search space is transformed with (i.e., T); see Hansen et al. (2011) for a complete discussion.

Transformation invariance is an important characteristic of an optimization algorithm. If an algorithm is invariant of a transformation T then, by definition, the performance of the algorithm on a problem P can be generalized to the set of problems

C ($P \in C$) that are defined by $(P; T)$. This enables researchers to make stronger statements (statements that are applicable to a larger class of problems), favorable or unfavorable, about the performance of an algorithm. Scaling, rotation, and translation are well-known transformations that are frequently used in different areas. Hence, it is valuable to understand if an algorithm is invariant under these transformations (see Wilke et al., 2007, Hansen et al., 2011, and Bonyadi and Michalewicz, 2014a, 2016). One can define the transformation $T(\vec{x})$ by $T(\vec{x}) = sQ\vec{x} + \vec{b}$ to formulate these transformations (scaling, rotation, and translation) where $s \in \mathbb{R}$ is a scale factor and $s \neq 0$, $Q \in \mathbb{R}^d \times \mathbb{R}^d$ is a rotation matrix, and $\vec{b} \in \mathbb{R}^d$ is a translation vector.

Transformation invariance has been investigated for many optimization algorithms (Hansen, 2000; Salomon, 1995). Among the transformation of rotation, scale, and translation, rotation has received most attention by researchers in the field of PSO. Probably the main reason for this special attention is that the rotation of the search space potentially makes an objective function non-separable (Hansen et al., 2011). Moreover, dealing with non-separable optimization functions is of more interest as there are many optimization problems with this characteristic. Translation invariance is probably the most basic characteristic among these transformations that an optimization algorithm must have. The reason is that, if an algorithm is sensitive to translation, the algorithm in fact is "making an assumption" about the location of the optimal solution of the problem at hand. Hence, if the optimal solution is shifted, that assumption is not valid anymore and this, in turn, causes a change to the performance of the algorithm.

Transformation invariance of SPSO was first analyzed in 2007, where it was shown (Wilke, 2005; Wilke et al., 2007) that SPSO is scale and translation invariant but is rotation variant. The reason that SPSO is rotation variant is that the effect of random diagonal matrices on the direction of movement is sensitive to the rotation of the coordinate system. In contrast, it was proven that LPSO (see Section 2) is rotation, scale, and translation invariant. However, LPSO suffers from a limitation called line search. In LPSO, if $(\vec{p}_t^i - \vec{x}_t^i) || (\vec{g}_t - \vec{x}_t^i)$ and $\vec{V}_t^i || (\vec{p}_t^i - \vec{x}_t^i)$, particle i oscillates between its personal best and the global best (see also Bonyadi et al., 2013) and it is not able to search in other directions. A new PSO variant was proposed in Wilke et al. (2007) which was rotation invariant while it did not have the line search limitation. That algorithm (called "Rotation invariant Particle Swarm Optimization," RPSO) used random rotation matrices rather than random diagonal matrices to perturb the direction of movement in each iteration. A method called *exponential map* was used to generate the rotation matrices. The idea of exponential map is that the exponential of any skew-symetric matrix $W\theta$ (i.e, $e^{W\theta}$), where W is a $d \times d$ matrix and θ is a scale factor, is a rotation matrix with the rotation angle θ (Moler and Van Loan, 2003). The exponential of a matrix $W\theta$ is defined by:

$$M = I + \sum_{i=1}^{maxC} \frac{1}{i!}(W\theta)^i, \tag{15}$$

where W is a $d \times d$ skew-symetric matrix, I is the identity matrix, θ is a scalar, and $maxC \rightarrow \infty$. In order to generate a random rotation matrix, one can generate a random W as $W = (A - A^T)$, where A is a $d \times d$ matrix with elements generated randomly in the interval [-0.5, 0.5]. Also, the angle of rotation in degrees is α (a real scalar) where $\theta = \frac{\alpha\pi}{180}$. It is clear in Eq. (15) that for a finite value for $maxC$, the matrix M becomes an "approximation" of a rotation matrix with the rotation angle α (see Moler and Van Loan, 2003 for details on how the truncation error is calculated). Wilke et al. (2007) set the value of $maxC$ to 1, limiting the approximation to one term of the summation only,

so that the time complexity of generating the rotation is reduced. Also, they claimed that, as the rotation is considered only for small values of α, this approximation does not impact the overall results. The time complexity for generating and multiplying the approximated rotation matrix (with $maxC = 1$) into a vector is in $O(d^2)$, including the cost of generating the approximated matrix ($O(d^2)$) plus vector-matrix multiplication ($O(d^2)$). For larger values of $maxC$, the order of complexity of generating M grows.

The idea of replacing random diagonal matrices by rotation matrices was further investigated by Bonyadi, Michalewicz, and Li (2014). A method for generating accurate random rotation matrices was proposed. The method was based on Euclidean rotation matrices that could be used to rotate vectors in any combination of hyper planes. Two approaches for rotation of the velocity vector were investigated in that paper: rotation in all possible planes (complexity in $O(d^2)$) and rotation in one randomly selected plane (complexity in $O(d)$). Experiments showed that rotation in one plane is just slightly worse (in terms of the objective value) than rotation in all planes; however, it is much faster. The random diagonal matrices were replaced by rotation matrices in all planes (generated by the proposed method) in several PSO variants proposed in Clerc and Kennedy (2002), Eberhart and Shi (2001), Nickabadi et al. (2011), PSO (2006), Shi and Eberhart (1998b), Van den Bergh and Engelbrecht (2010), and Zheng et al. (2003). Results (based on the Wilcoxon test) showed that PSO variants usually benefit from random rotation matrices to solve tested benchmark functions (18 test functions that included separable and non-separable test problems).

Rotating the search space of a problem potentially makes the problem non-separable (Salomon, 1995). SPSO was applied to several optimization problems when their search space were rotated (Hansen et al., 2011, 2008) to test if the performance of the algorithm changed. Results of applying SPSO to these problems were compared with "Covariance Matrix Adaptation ES" (Hansen and Ostermeier, 1996), CMA-ES, and "Differential Evolution," DE (Storn and Price, 1997). Experiments showed that, although the performance of SPSO in separable optimization problems is acceptable, the algorithm performs poorly when it is applied to non-separable optimization problems. In addition, the performance of the algorithm changes when the search space of the problem is rotated which indicates that the algorithm is rotation variant. Hansen et al. (2011) also studied the performance of these methods in dealing with ill-conditioned[21] functions. It was found that the performance of SPSO is almost independent of the condition number for ill-conditioned separable optimization functions. Experiments showed that SPSO outperforms CMA-ES on such functions. However, the performance of SPSO drops almost linearly with the condition number when it is applied to ill-conditioned non-separable functions with condition number larger than 100. Experiments showed that CMA-ES outperforms SPSO in optimizing such functions. A similar comparison was also conducted in Auger et al. (2009) where the findings presented in Hansen et al. (2011, 2008) related to the good performance of SPSO for separable optimization problems and its poor performance in non-separable optimization problems were confirmed. In addition, the authors argued that the main reason behind this poor performance is

[21] An optimization problem is called ill-conditioned if the condition number of its objective function is significantly large (Trefethen and Bau III, 1997). The condition number of a function F is a measure to understand how sensitive that function is to the changes of the values of its variables. Condition number of a function F is defined by the maximum relative condition number of F over all possible points in its domain. The relative condition number of the function F at a point $\vec{x}$ is defined by the maximum ratio of $\frac{||F(\vec{x}+\delta\vec{x})-F(\vec{x})||}{||F(x)||}$ to $\frac{||\delta\vec{x}||}{||\vec{x}||}$ when $||\delta\vec{x}|| \leq \epsilon$ as ϵ goes to zero (see Trefethen and Bau III, 1997).

that all calculations in SPSO are performed for each dimension separately (Auger et al., 2009), which makes the algorithm rotationally variant.

It was shown (Spears et al., 2010) that not only is SPSO rotation variant, but also particles in SPSO are biased toward the lines parallel to the coordinate axes. In fact, by tracking the positions of particles in the search space during the optimization process, it was found that particles tend to sample more points along the lines parallel to coordinate axes than other points in the search space; that is, particles' positions are "biased" to the lines parallel to the coordinate axes. The authors used these two issues (rotation variant and bias) to design test problems that are hard or easy for SPSO to optimize.

The rotation invariance and local convergence in PSO were integrated in Bonyadi and Michalewicz (2014a)–see Eq. (13) for the variant proposed in that study, called LcRiPSO. It was proven that LcRiPSO is rotationally invariant and locally convergent if the operator m is invariant under any rotation and it satisfies the condition formulated in Eq. (14). It was proven that if the operator m generates a new point following the normal distribution, then it satisfies conditions for local convergence and rotation invariance of the algorithm. The time complexity of LcRiPSO using the normal distribution for m is in $O(d)$, that is faster than the variants proposed by Wilke et al. (2007) and Bonyadi, Michalewicz, and Li (2014).

Summary: Importance of transformation invariance in iterative optimization algorithms has been emphasized in many articles (Bonyadi and Michalewicz, 2014a; Hansen et al., 2011, 2008; Wilke, 2005; Wilke et al., 2007). Among all transformations (shear, stretch, rotation, translation, and scale), rotation has received special attention in PSO related articles because SPSO is sensitive to the rotation of the search space. It is believed that the poor performance of SPSO in dealing with non-separable search spaces stems from being rotation variant (Hansen et al., 2008). One way to address rotation variance issue in SPSO is to use random rotation matrices rather than random diagonal matrices in the velocity update rule as was done in Wilke et al. (2007) and Bonyadi, Michalewicz, and Li (2014). However, the time complexity of the velocity update rule using this idea is in $O(d^2)$, which makes this approach inefficient to deal with large-scale problems. Thus, it would be valuable to investigate other ideas to make the algorithm rotationally invariant while keeping the time complexity of the algorithm linear. An example for such idea can be found in Bonyadi, Michalewicz, and Li (2014) where it was proposed to conduct the rotation in only one plane (selected randomly) that reduces the time complexity to $O(d)$. Also, Bonyadi, Michalewicz, and Li (2014) experimentally showed that the use of rotation matrices not only makes the algorithm rotationally invariant but it improves the performance of different PSO methods when they are applied to standard benchmarks. Another idea to make the algorithm rotation invariant was proposed in Bonyadi and Michalewicz (2014a) where the random diagonal matrices were replaced by random scalars and a perturbation was added to move personal best of particles to avoid stagnation. It was proven that the algorithm is rotation invariant and the time complexity of the velocity update rule is in $O(d)$.

As a future direction for research, it might be interesting to compare the results of rotationally invariant PSO variants to other optimization algorithms such as CMA-ES and DE. Another potential direction would be to analyze the stability of PSO variants that are rotationally invariant (e.g., SPSO2011, RPSO). Note that this analysis might be different from that of SPSO because all convergence analyses conducted on SPSO considered that the velocity for each particle is updated for each dimension independently (see Section 3.1.1). However, this is not the case when rotation matrices are used because the procedure of rotation is not performed dimension by dimension (see Bonyadi and

Michalewicz, 2014d, 2016 for details). These analyses might lead researchers to identify the convergence boundaries for the coefficients for these algorithms. Moreover, one can use the principal components to adjust the search directions to improve the algorithm in non-separable search spaces (see also Voss, 2005). As the particles move in the search space, they collect samples that can be used to estimate local gradients and correlations between dimensions, including multiple correlations or principal components. This information can be used to verify/modify the particles velocities so that the swarm can find better solutions. This can be also useful to deal with ill conditioning.

4 Modifications of PSO to Deal with Unconstrained Optimization Problems (UOP)

Three different categories of approaches were considered for improving PSO: setting parameters, modifying components of the algorithm, and combining the algorithm with other algorithms. Setting parameters refers to setting the topology, coefficients (acceleration coefficients or inertia weight), and population size. Modifying components refers to changes of the velocity or position update rule (including adding new components; modifying the way they are calculated). Combining the algorithm with other algorithms refers to hybridization of PSO with other methods. In the following subsection we review PSO variants which have improved the performance of SPSO through specific settings of its parameters (Section 4.1), modification of the velocity/position update rules (Section 4.2), and hybridization of the algorithm (Section 4.3).

4.1 Parameter Setting

Parameter setting has been a challenge in iterative optimization methods in the past years (Eiben et al., 1999). Two different approaches for setting parameters of an evolutionary algorithm might be considered: parameter tuning and parameter control. Parameter tuning refers to setting parameters of an algorithm through experiments to some constant values. Parameter control, however, refers to design of a strategy which changes the value of parameters during the run. Parameter control approaches are categorized further into three groups: deterministic, adaptive, and self-adaptive. In deterministic parameter control approaches a rule (called time-varying rule) is designed to calculate the value of a parameter based on the iteration number. In adaptive parameter control approaches, a function is designed that maps some feedback from the search into the value of the parameter. In a self-adaptive parameter control approach, the parameters are encoded into individuals and are modified during the run by the optimization algorithm.

In this subsection, articles that have studied different parameters for PSO (topology, coefficients, and population size) are reviewed.

4.1.1 Topology

The concept of topology in PSO was taken from social psychology, the science of how people behave in a society. In societies, each individual communicates with other individuals with whom he or she is connected. The network of individuals in a society suggests the same concept as the topology in PSO variants does. Probably the earliest attempt to modify the topology (see Section 2 for the definition of topology) in PSO was conducted in Kennedy (1999).[22] The idea was that topology should affect the explorative

[22]Two simple topologies were actually introduced earlier in Eberhart and Kennedy (1995); however, that study was not focused on the topic of topology.

and exploitative behavior of the swarm as different topologies impose different speed of propagation of information among particles. Four different topologies were tested:

1. Circle (local best or ring) topology: $T_t^i = \{i, \psi_t\}$ where $\psi_t = \underset{l=\{i-1,i,i+1\}}{argmin} \{F(\vec{p}_t^l)\}$.
2. Wheels topology: $T_t^1 = \{1, \tau_t\}$ where $\tau_t = \underset{l=\{1,\ldots,n\}}{argmin} \{F(\vec{p}_t^l)\}$ and $T_t^{i\neq 1} = \{1\}$.
3. Star (global best) topology: $T_t^i = \{i, \tau_t\}$ where $\tau_t = \underset{l=\{1,\ldots,n\}}{argmin} \{F(\vec{p}_t^l)\}$.
4. Random edge topology: $T_t^i = \{i, j\}$ where j is randomly selected among other individuals.

Kennedy (1999) tested all these topologies on a benchmark of four functions. Experiments showed that the global best topology is the fastest communication topology, while the ring topology is the slowest in the tested benchmark set. This results in the highest exploitative behavior in the former and the highest explorative behavior in the latter topology. This finding was also investigated in Kennedy and Mendes (2002) and later in Mendes et al. (2004). Mendes et al. (2004) proposed a more general formulation for CCPSO, called "Fully Informed Particle Swarm," FIPS, and different topologies were tested under that formulation. All these studies, however, suffer from the lack of proper statistical analysis of the results or an adequate number test cases that affect the generality of the conclusions drawn.

To address the lack of statistical analysis and the number of test cases to draw conclusions about the performance of different topologies, an extensive comparison (60 benchmark test functions) between the global best and local best topologies was conducted by Engelbrecht (2013). Experiments on 60 test functions showed that these two topologies perform almost the same on separable, non-separable, unimodal, and multimodal problems with a slight favor toward the global best topology in terms of the quality of the final solutions. Also, it was experimentally shown that the diversity of the swarm is not consistently higher or lower when either of these two topologies is used. Hence, it appears that the performance of these topologies and the diversity they impose on the swarm also depends on the structure of the landscape of the problem at hand.

Apart from well-known topologies such as global best and local best, some other topologies were also proposed in the PSO literature. For example, based on the assumption that it is usually better to make high-quality particles influence the position update of other particles, a "Hierarchical Particle Swarm Optimization," HPSO, was proposed by Janson and Middendorf (2005). Particles in HPSO were arranged in a tree structure with a predefined branching factor (a hierarchy). The tree was updated at each iteration based on the quality of the personal best of particles in a way that better particles were placed at higher levels. The velocity of each particle was then updated according to its own personal best, together with the personal best of the particle in its parent node in the tree. Experiments (6 test functions, all had their optimum solutions at the center of the coordinate system) showed that larger values for the branching factor are more appropriate for the beginning of the optimization process, while a smaller branching factor performs better at the later stage of the optimization process.

Selection of a good topology during the run is a natural extension that can be considered in SPSO. This idea was proposed in Elsayed et al. (2012) where two PSO variants with different topologies, the global best topology and a random tournament

topology, were used in two independent populations. According to the performance of these populations, the particles were taken from one population and added to the other to make more use of the population that shows better performance. Also, a local search was added to improve the results found by the algorithm. The proposed method was applied to 13 standard benchmark problems and results were compared with other methods such as variants of ES and DE using the Wilcoxon test.

As the topology in PSO has its roots in social networks, one might borrow the structure of such networks and apply the idea to communication between particles in PSO. For example, it has been shown (Albert and Barabási, 2002) that in real society networks there are a few nodes that are connected to a large number of nodes in the network. Also, most nodes have a small degree, meaning that most nodes are connected to only a few nodes in the network. When a new node is added to a real network, it usually prefers to attach to the nodes that have more connections; that is, riches get richer (a network with these properties is called a scale-free network). These analogies were adopted and implemented for updating the topology in PSO (Zhang and Yi, 2011). Two types of particles (in SPSO) were defined in a swarm: active and inactive. The active particles move around the search space and look for optima, while inactive particles are not updated until they become activated. At each iteration, all inactive particles are activated by connecting to one of the existing active particles. This method was called "Scale-Free Fully Informed Particle Swarm Optimization," SFIPSO. Some experiments were conducted to compare the diversity of particles when the proposed topology, ring topology, or global best topology is used. It was found that the proposed topology offers a better diversity for particles in the search space compared to two other tested topologies. Comparison results (supported by Wilcoxon test) on 8 benchmark test functions showed that the proposed dynamic topology can improve the performance of the algorithm on the tested functions.

Another idea taken from social psychology, "six degrees of separation" (Newman et al., 2006) (any two arbitrary persons are connected to each other by at most six steps in the network of their friends) was used to determine connections between particles in PSO (Gong and Zhang, 2013). Every certain number of generations, k other particles are selected randomly to contribute into update rules of each dimension j of a particle i in the swarm. This in fact enables particles to update their different dimensions using different network of particles, which is a generalization of the concept of topology. The algorithm was applied to 13 standard test functions and the results were compared (using mean, standard deviation, and reported box plots) with some other PSO based methods.

There have been some other good articles related to topology (Bratton and Kennedy, 2007; Mendes, 2004). These articles addressed the analysis of the influence of topology on the performance of SPSO (Mendes, 2004) (an extensive study related to topology), setting parameters (including topology) experimentally (Bratton and Kennedy, 2007), time-varying topologies (Bonyadi, Michalewicz, and Li, 2014; Emara, 2009), and an adaptive topology PSO (Cooren et al., 2009). However, these studies have been excluded from this review because of the selection criteria (see Section 1).

Summary: Communication among individuals in a society is of great importance. This is also true in swarm-based optimization algorithms where topology defines the communication lines among individuals. It was shown (on a limited number of test functions) that the topology influences the speed of convergence and diversity of the swarm (Kennedy, 1999; Kennedy and Mendes, 2002) in SPSO. For example, global best topology favors exploitative (over explorative) behavior of the swarm and fast

convergence to the global best vector while the ring topology favors explorative (over exploitative) behavior and slow convergence to the global best vector (Kennedy and Mendes, 2002). Although these claims appeared intuitive, the lack of adequate experiments made these observations difficult to confirm. To address this, Engelbrecht (2013) provided an extensive experiment to test these claims and, interestingly, they were disproved through these extensive experiments. Experiments showed that the diversity of particles and the performance of the algorithm do not only depend on topology, but also depend on the problem at hand and thus such general claims about topologies are not correct. Hence, one can expect adaptive topologies to improve the search ability of the algorithm. An example of adaptive topologies is to change the topology so that high-quality particles contribute more in updating the position of other particles; see Janson and Middendorf (2005) and Zhang and Yi (2011) for examples. These studies experimentally showed (though on a small set of test functions) that such ideas can improve the performance of the algorithm. Another topology adaptation approach was proposed in Elsayed et al. (2012) where two PSO variants, one with the global best topology and the other with a random tournament topology, cooperated during the run. While this idea looks interesting, the algorithm proposed in Elsayed et al. (2012) contained other components (e.g., a local search) and, because of the limited comparisons in that article, it is not possible to conclude that the proposed adaptive topology was responsible for the good performance observed in the algorithm.

In terms of future work, it seems that topology affects the performance and diversity of the algorithm. However, both of these measures (performance and diversity) are also affected by the characteristics of the landscape. Hence, it would be valuable if all these topologies (ring, global best, hierarchical, etc.) are implemented and classified based on their performances or diversities they impose to the swarm according to the problem characteristics (using a decision tree for example). This direction can lead to designing an adaptive topology according to the landscape characteristics detected during the run. In addition, there are not many articles on the theoretical aspects of topologies and how they influence the search process—such studies can be considered a promising future research direction. In fact, this was one of the challenges in PSO introduced in 2007 (Poli, Kennedy, et al., 2007): "How do the details of the equations of motion or the communications topology affect the dynamics of the PSO?," which has not yet been addressed properly!

4.1.2 Coefficients

The performance of SPSO is affected by changing the values of its coefficients, that is, acceleration coefficients and inertia weight (see Section 3.1.1). Thus, several attempts were made (Clerc, 1999; Shi and Eberhart, 1998a; Eberhart and Shi, 2000; Carlisle and Dozier, 2001) to tune the values of these coefficients for a set of benchmark functions. These studies found that the best values for the inertia weight falls in [0.4, 1.1] while the best values for acceleration coefficients falls in [1.5, 3] on most experiments. Also, these studies experimentally (on a set of fewer than 6 selected test functions) showed that the coefficients affect the exploration and exploitation ability of the algorithm. In particular, it was shown that, for constant acceleration coefficients, larger values for the inertia weight result in a better exploration and smaller values for the inertia weight result in a better exploitation. Thus, a linear decreasing inertia weight (ω decreased linearly from 0.9 to 0.4 during the run) was tested by Shi and Eberhart (1998b) on one test problem—this variant was called "Decreasing Inertia Particle Swarm Optimization," DIPSO. Later, Zheng et al. (2003) tested an opposite approach, increasing the inertia

weight during the run from 0.4 to 0.9, and they found that this idea actually works better (based on the average of the objective value) than decreasing inertia weight on some test cases.

The idea of decreasing inertia weight was extended by Chatterjee and Siarry (2006) in a way that a non-linear function was used to decrease the value of the inertia weight: $\omega_i = \frac{(maxi-i)^n}{maxi^n}(\omega_s - \omega_e) + \omega_e$, where ω_s and ω_e were starting and final inertia weights, and $maxi$ was the maximum allowed iterations. With $n = 1$, this formulation is exactly the same as that in DIPSO. After experimenting with the sphere function, the parameters of this variant were set to: $\omega_s = 0.2$, $\omega_e = -0.3$, $n = 1.2$, and $maxi = 2{,}000$. This method with these settings was compared (based on the average of the quality of found solutions) with other optimization methods such as a genetic algorithm and a differential evolution on 17 benchmark problems.

Although inertia weight influences exploration and exploitation behavior of the algorithm, acceleration coefficients also play an important role in this regard. A time-varying approach was proposed (Ratnaweera et al., 2004) where the acceleration coefficients were changed during the run (the method was called "Time Varying Acceleration Coefficient Particle Swarm Optimization," PSO-TVAC). In that method, the value of φ_1 (cognitive weight) decreased while the value of φ_2 (social weight) increased during the run of the algorithm. Thus, particles tend to conduct exploration at the beginning and exploitation at the latter stages of the optimization process. The authors further modified PSO-TVAC by considering an adaptation rule to reinitialize the velocity of particles and another variant (Ratnaweera et al., 2004) called "self-organizing Hierarchical Particle Swarm Optimization," HPSO-TVAC, emerged. It was also proposed to reinitialize the velocity of particles when its value becomes zero (with some precision). The authors concluded that, based on the results, HPSO-TVAC is the best choice among tested methods. This conclusion was supported by only 5 test functions and results were analyzed with regard to the mean and standard deviation values. One should note that, the idea of reinitializing each dimension of velocity when it converges to zero is very similar to the idea proposed in Schmitt and Wanka (2013) (see Section 3.1.3) where velocity is reinitialized whenever a particle was stagnating. Hence, although a proper proof might be needed for the local convergence of an algorithm, we conjecture that HPSO-TVAC is locally convergent as it overcomes the stagnation.

As experiments confirmed potential benefits of time-varying approaches on some test functions, an adaptive approach for setting the value of coefficients during the run can be also beneficial as it adjusts the coefficients values according to the current situation. An example of such idea was proposed by Arumugam et al. (2008) where the values of inertia weight and acceleration coefficients were related to the objective value of the personal best and global best vectors, that is, smaller acceleration coefficients and larger inertia weight if the personal best of a particle is not as high-quality as the global best vector. Also, the authors experimented with mutation and crossover operators to understand if these operators can improve SPSO. The experiments were conducted on 13 standard test functions and mean, variance, median, minimum, and maximum of found solutions were compared with that of other methods. It was concluded that the tested PSO variants most likely benefit from crossover and mutation operators according to the experiments. Also, the proposed adaptation showed a good potential to enhance the algorithm to find better solutions in the tested problems.

Another adaptive approach, called "Adaptive Particle Swarm Optimization," APSO, was proposed by Zhan et al. (2009) in which three modifications took place: adaptive changes of the inertia weight according to the distribution of particles,

adaptive changes of the acceleration coefficients according to the evolution state, and new update rules in specific evolution states. Evolution state was defined as one of the states that a particle might be in, including exploration, exploitation, convergence, and jumping out. The distance of the position of particles from the global best vector was used as a measure to identify the evolution state of particles. In addition to changing the values of acceleration coefficients, a perturbation procedure was applied to the global best particle when all particles are close to each other. The concept of "closeness" was defined by a fuzzy membership function. APSO was compared with some other PSO variants when they were applied to 12 test functions. Results showed that APSO is significantly (based on a t-test[23]) better than other tested PSO variants in most cases. The optimal solution of most selected functions in that study was in the center of the coordinate system. One should note that, although local convergence of an algorithm entails proper theoretical analyses, the perturbation strategy used in APSO is very likely to help the algorithm to converge to prevent stagnation and lead the particles to a local optimum.

In order to balance the exploration and exploitation behavior of the particles, another adaptive approach, called "Adaptive Inertia Weight Particle Swarm Optimization," AIWPSO, was proposed (Nickabadi et al., 2011). In AIWPSO, the value of the inertia weight was adaptively changed during the search process. The authors proposed a linear relation between the inertia weight in the current iteration and the number of particles improved in the previous iteration. The idea behind this approach was that if success rate is high, then the current direction is promising and, hence, it is better to have a higher value for the inertia weight so that particles are encouraged to move with larger steps in the current direction. However, when the success rate is low, the current direction is not promising and hence the inertia weight is decreased. Although the idea is simple, experimental results on 15 benchmark test functions showed a considerable improvement (in terms of mean and standard deviation) in solution qualities in comparison to many other adaptive/time-varying strategies for controlling the value of the inertia weight.

Recently, another adaptive approach was proposed in which both inertia weight and acceleration coefficients were modified during the search (Leu and Yeh, 2012). The method used a similarity measure called the gray relation (Deng, 1989) (the PSO variant was thus called "Gray Particle Swarm Optimization," GrPSO). The idea was based on the following observation: If a particle is less similar to the global best vector (relative to the other particles), then it is in the exploration phase. Thus, larger values for inertia and cognitive weights for that particle are preferred. Also, as the value of social and cognitive weight were related to each other by $\varphi_1 + \varphi_2 = 4.0$ (Clerc and Kennedy, 2002), changing the value of cognitive weight enforces changes the social weight as well. Experiments were done for 12 standard test functions and results were compared with other PSO variants, such as HPSO-TVAC and APSO, based on the mean values of the found solutions.

There have been some other high-quality studies that have proposed different approaches in setting the coefficients of SPSO such as Bratton and Kennedy (2007), Ghosh

[23]The t-test is not necessarily the best choice to be used, unless the data are approximately normal or the effective sample size of the experiment is large enough. While in practice this test is robust to moderate deviations of normality, it is particularly sensitive to heavily skewed distributions, which are relatively common in algorithmic results. A less sensitive test for this data is the Wilcoxon test, which does not assume normality (even though it has its own—often ignored—assumptions; see Sheskin, 2003). See Derrac et al. (2011) for details on statistical tests for comparing stochastic methods.

et al. (2010), and Shi and Eberhart (2001). For example, in Bratton and Kennedy (2007), a set of parameters for SPSO were suggested which resulted in a good performance of the algorithm on some benchmark problems. Also, two inertia adaptive PSOs were proposed in Ghosh et al. (2010) and Shi and Eberhart (2001). However, all these articles have been excluded from this review because of the selection criteria (see Section 1).

Summary: There have been many studies to set the inertia weight (ω) and acceleration coefficients (φ_1 and φ_2) of SPSO. Some studies tried to tune these coefficients (Carlisle and Dozier, 2001; Liu, 2014) through experiments. However, as tuning is done on a limited number of test functions and the results are not generalizable, a parameter control approach is usually preferable. For instance, several time-varying approaches were introduced to change the values of coefficients of SPSO during the run. The main idea for time-varying approaches was based on an observation in most iterative optimization algorithms: Usually encouraging exploration at the beginning and exploitation at the latter stages of the optimization process is beneficial. In the context of PSO, under the condition that the coefficients values are inside the convergence boundaries (see Section 3.1.1); Ratnaweera et al. (2004) and Shi and Eberhart (1998b) experimentally showed that large values of inertia and cognitive weights (φ_1) relative to the social weight usually encourage exploration, and a large value for social weight (φ_2) relative to the inertia and cognitive weights usually encourages exploitation around the best ever-found solutions. For example, Shi and Eberhart (1998b) experimented with reducing the values of inertia weight and Ratnaweera et al. (2004) experimented with changing the acceleration coefficients. Although most of these hypotheses were intuitively sound, the lack of accurate experimental analyses can be clearly observed in many of these articles (i.e., in Clerc, 1999, Shi and Eberhart, 1998a, Eberhart and Shi, 2000, Carlisle and Dozier, 2001, Ratnaweera et al., 2004, and Chatterjee and Siarry, 2006). There were no proper statistical analyses reported in these articles and the number of test cases was very limited which makes the drawn conclusions specific to a small number of cases. The reason behind such shortcomings could be the lack of appropriate benchmark functions or the hardware/software limitations at that time. The consequence of such limited experiments became more apparent when it was shown (Zheng et al., 2003) that increasing the value of ω from 0.4 to 0.9 actually works better than decreasing it on some test cases. One challenge in time-adaptive approaches is that it is not easy to determine how fast the coefficient values should be changing during the run. Indeed, the speed of these changes depends on the characteristics of the landscape (e.g., if the landscape is very smooth, then radical changes of the coefficients might not be desirable). An adaptive approach to control these variables can deal with this issue as it makes no assumptions about the landscape of the problem (see Arumugam et al., 2008, Leu and Yeh, 2012, Nickabadi et al., 2011, and Zhan et al., 2009). In spite of the presence of some shortcomings in the experimental comparisons in these articles (e.g., selection of biased test cases in Zhan et al., 2009 and lack of statistical tests in Leu and Yeh, 2012), most results confirmed that adaptive approaches can improve the performance of the algorithm. The main challenge in adaptive approaches is, however, to recognize which behavior (explorative or exploitative) is preferable at the current stage of the search (the so called evolution state in Zhan et al., 2009). For example, the distance between particles can be used as a measure to determine the evolution state (used in Zhan et al., 2009); however, this calculation increases the time complexity of the algorithm. Similarity measure (Leu and Yeh, 2012) and success rate (Nickabadi et al., 2011) are two other candidate measures to determine the state of the search; however, there is no concrete evidence (apart from experiments) to show that these strategies are

effective in all situations. Note that changes of the coefficients' values does not affect the transformation and convergence properties (e.g., local convergence) of the algorithm.

In summary, there have been many advances on the parameter setting of SPSO and different approaches have been introduced including time varying, adaptive, and tuning. One potential future direction related to the time-varying approaches would be to study the effects of parameters on a large benchmark of test functions (e.g., CEC or BBOB benchmark sets) using proper statistical tests (such comprehensive study for parameter tuning has been already done by Liu, 2014 for SPSO). Another interesting topic related to the parameter tuning would be to study the performance of the algorithm according to the landscape characterization and then classify the results (see Caamao et al., 2010, for example) based on the achieved performances. Also, comparison (philosophical discussion, theoretical, or experimental) of different adaptive approaches and measures to determine the evolution stage can be very beneficial for further enhancements of the algorithm. Another possibility would be to use analysis presented in Trelea (2003) (Section 3.1.2) to set the behavior of the particles (oscillating, zigzagging, and convergence) adaptively according to the current state of the particle (see also Bonyadi and Michalewicz, 2016 for a discussion on this topic).

4.1.3 Population Size

Generally, most iterative search methods encourage exploration during the early stages of the iterative process while they encourage exploitation in the latter stages. It is also well known that concentrating individuals only on the area of the highest potential during the exploitation phase is usually beneficial. Thus, as this area is small relative to the whole search space, a small swarm may perform almost as well as a large swarm. Hence, it might be better to reduce the population size at the latter stages of the optimization process to save on the number of function evaluations.

Probably the first attempt to design an adaptive population size in an evolutionary algorithm was presented in 1994 by Arabas et al. (1994). The idea of adapting population size was adopted (Chen and Zhao, 2009) for PSO later in 2009 through a method called "Ladder Particle Swarm Optimization," LDPSO. In LDPSO, at every predefined period of time, the diversity of the swarm was evaluated and the swarm size was adjusted accordingly. The diversity measure was based on the average Manhattan distance between particles. Comparisons showed that the algorithm outperforms some other PSO variants in terms of the average solution quality when it was applied to 5 standard benchmark problems.

A population manager, a solution sharing technique, and a search range sharing strategy were proposed in Hsieh et al. (2009) (this variant was called "Efficient Population Utilization Strategy Particle Swarm Optimization," EPUS-PSO). The population manager in EPUS-PSO adjusted the population size by:

1. removing a particle from the swarm when the global best vector was improved at least once in the previous k consecutive iterations,
2. adding a new particle to the swarm if the global best vector did not change in the previous k consecutive iterations,
3. replacing an existing one when adding is not permitted (the upper bound of the number of particles in the swarm has been reached).

EPUS-PSO also used the global best vector in the velocity update rule of each particle with the probability P and used the personal best of another particle (selected

by a tournament selection approach) with probability $1 - P$ (this approach was called solution sharing strategy). The aim of the solution sharing strategy was to give a chance to particles to learn from other particles rather than always using the global best vector in the velocity update rule. In addition, a solution range-sharing strategy was presented to prevent the particles from premature convergence. The algorithm was applied to 15 benchmark test functions, together with their rotated versions in 10 and 30 dimensions. Results (based on the average and standard deviation) showed that the algorithm is comparable with other PSO variants.

A strategy called "Incremental Social Learning," ISL (Montes de Oca and Stützle, 2008), was used to set the population size of SPSO during the run (Montes de Oca et al., 2011). ISL suggests an increasing population-size approach that in some cases facilitates the scalability of systems composed of multiple learning agents. Two variants of PSO were proposed based on the ISL idea. In the first variant, called IPSO, whenever the algorithm could not find a satisfactory solution, a new particle was added to the population. In the second variant (called IPSOLS, IPSO with a local search), a local search approach was run to gather local information around the current position. If the local search procedure was not successful (no better solution was found around the current position), then it was concluded that the particle is already in a local optimum. In this situation, a new particle was added to the swarm that was placed in the search space through a simple social learning approach, that is, the personal best location of one of the existing particles (called "model" particle) was considered for the location of the new particle. The algorithm was applied to 12 standard optimization test functions and the mean and median of the results were compared with those of other PSO methods. Results showed that the proposed method is capable of finding high-quality solutions and it is comparable with other methods on the tested problems.

Summary: The majority of particles are very close to each other at the exploitation phase to improve the best ever found solution. In contrast, during the exploration phase, particles are far from each other to explore the search space to locate as many basins of attractions as possible. Thus, a large population size during the exploration phase and small population size during the exploitation phase seem to constitute a reasonable choice (Chen and Zhao, 2009) for effective optimization. Chen and Zhao (2009) used the diversity of the swarm as an indicator for the exploration/exploitation tendency and increased/reduced the size of the swarm accordingly. The experiments in that article were, however, not supported by sufficient number of test cases. Also, the calculation of the swarm diversity increases the time complexity of the algorithm which might be an issue in dealing with large-scale problems. A measure based on the improvement rate of the global best particle was used by Hsieh et al. (2009) and Montes de Oca et al. (2011) to determine exploration/exploitation tendency of the swarm. Experiments in those articles showed that the methods can improve the performance of the algorithm; however, both articles included several different components (e.g., a local search, solution sharing strategies) to their proposed methods that makes it impossible to understand which component plays the most important role in this improvement.

It is not a trivial task to determine which behavior (exploration/exploitation) is preferred in each iteration to set the population size accordingly. As a direction for future study, it would be valuable to explore different options to determine the exploration/exploitation tendency of the swarm to adjust the population size during the run. This needs to be evaluated very carefully through appropriate selection of benchmarks and statistical tests. It is also important to not overwhelm the algorithm so that only the population sizing strategy is tested. Another interesting research topic is to investigate

what is the best strategy to add/remove a particle to/from the swarm. For example, one needs to determine the position, velocity, and personal best of the particle that is going to be added to make sure the new particle can improve the search ability of the swarm. Finally, theoretical studies related to the population size are missing in the literature related to PSO. One can study the effect of population size on the search ability or EFHT of the algorithm. Further, diversity of the swarm is another related topic that can be investigated under different test functions or theoretically. For example, one can imagine that the diversity of the swarm is higher when the coefficients are set so that particles' oscillation is zigzagging (see Section 3.1.2). This, however, is an intuition that can be tested and, further, adopted for a more effective search.

4.2 Modification of the Velocity/Position Update Rules

The introduction of the inertia weight in OPSO was probably the first modification of the velocity in the original algorithm (Shi and Eberhart, 1998a). (This algorithm, called SPSO, was reviewed in Section 2.) Since then, many attempts were made to improve the velocity and position update rule of SPSO further.

An issue called "two steps forward one step back" was identified (Van den Bergh and Engelbrecht, 2004) in SPSO: As all dimensions of the position of particles are updated at every iteration, there is a chance that some components in this vector move closer to a better solution, while others actually move away from that good solution. As long as the objective value of the new position vector is better than the objective value of the previous position, SPSO assumes that the solution has been improved, regardless whether some dimension values have moved away from the good solution. To address this issue, it was proposed to divide the search space to d subproblems (each variable becomes a subproblem) and different variables are updated by different sub-swarm—this is called "Cooperative Particle Swarm Optimization," CPSO (Van den Bergh and Engelbrecht, 2004). Although CPSO performed better than many other variants of PSO, it had two potential issues (Van den Bergh and Engelbrecht, 2004) as follows:

- It might converge to pseudominima, that is, minima that were generated because of the partitioning of the search space, and
- Its performance depends on the correlation of the subproblems.

As SPSO does not suffer from these issues, a hybrid method was also proposed (Van den Bergh and Engelbrecht, 2004) that combined SPSO with CPSO. Experimental results for 5 test functions were analyzed in that paper based on the average and standard deviation of the quality of final solutions and number of function evaluations.

The global best vector in SPSO is considered as an indicator of a potentially high-quality area in the search space. Thus, it would be worthwhile to use this location together with the current location of a particle to generate a new location. This idea was used in a PSO variant called "extrapolation Particle Swarm Optimization," ePSO (Arumugam et al., 2009), in which the position update rule was revised as follows:

$$\vec{x}_{t+1}^{\,i} = \vec{g}_t + \alpha\vec{g}_t + \gamma(\vec{g}_t - \vec{x}_t^{\,i}), \tag{16}$$

where $\alpha = k_1 r_t^i$, $\gamma = k_1 e^{k_2\beta}$, $k_1 = k_2 = e^{\frac{-current\ iteration}{\max number\ of\ iterations}}$, r_t^i is a random value between 0 and 1, and $\beta = \frac{f(\vec{g}_t) - f(\vec{x}_t^{\,i})}{f(\vec{g}_t)}$ (there is no velocity vector in this variant). The term $\vec{g}_t + k_1 r_t^i \vec{g}_t$ generates a random point around the global best vector. Note that, as k_1 becomes smaller over time (iterations), the generated points by $\vec{g}_t + k_1 r_t^i \vec{g}_t$ become closer to $\vec{g}_t$ that results

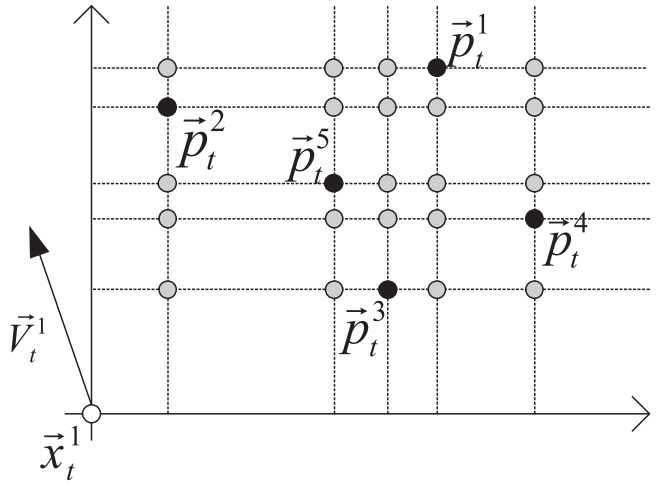

Figure 5: There are 5 particles in the swarm. The personal best of each particle is shown by a black circle. The gray circles are potential locations for $\vec{P}_t^1$ in CLPSO. The position of particle 1 is depicted at the origin of the coordinates (white circle).

in better exploitation around the global best vector at the later stages of the run. The last term $k_1 e^{k_2\beta}(\vec{g}_t - \vec{x}_t^i)$ determines the step size that the current position ($\vec{x}_t^i$) takes to go toward the global best vector. This step size is controlled by the objective value of the current location and the objective value of the global best vector. Experiments on 13 standard optimization functions showed that ePSO performs better than SPSO on almost all of the tested functions, supported by the Wilcoxon test.[24]

According to the velocity update rule of SPSO, attraction toward the personal or global best vectors does not depend only on the acceleration coefficients, but also depends on the average values of $\vec{p}_t^i - \vec{x}_t^i$ and $\vec{g}_t - \vec{x}_t^i$. It was observed that in SPSO the value of $\vec{p}_t^i - \vec{x}_t^i$ is usually smaller than $\vec{g}_t - \vec{x}_t^i$ for any particle i and any iteration t that results in larger attraction towards $\vec{g}_t$ than $\vec{p}_t^i$. Hence, particles tend to concentrate around $\vec{g}_t$, which reduces the diversity of the swarm (Liang et al., 2006). To overcome this issue, a variant of PSO called "Comprehensive Learning Particle Swarm Optimization," CLPSO, was proposed (Liang et al., 2006). In this variant, the velocity update rule was modified to the following form:

$$\vec{V}_{t+1}^i = \omega \vec{V}_t^i + \varphi R_t^i (\vec{P}_t^i - \vec{x}_t^i), \tag{17}$$

where the value of the j^{th} dimension of $\vec{P}_t^i$ ($\vec{P}_t^{i,j}$) is calculated by $P_t^{i,j} = p_t^{l_i(j),j}$ where $l_i(j) \in \{1, 2, \ldots, n\}$ (index of a particle). In fact, each dimension of $\vec{P}_t^i$ is potentially taken from the personal best of different particles. Figure 5 shows potential locations where $\vec{P}_t^i$ might be selected from for a particle i in a swarm of size 5 as an example. The

[24]The position update rule of ePSO is sensitive to translation of the search space. The reason is that, in this algorithm, the point that is generated by $\vec{g} + \alpha\vec{g}$ is sensitive to the position of $\vec{g}$ relative to the center of the coordinate system. In fact, the area where the point $\vec{g} + \alpha\vec{g}$ is sampled from changes if the coordinate system is shifted. This means that the algorithm is sensitive to the translation of the coordinate system. If $\alpha\vec{g}$ was replaced by $\alpha(\vec{g} - \vec{x}_i^t)$ then this algorithm becomes translation invariant because the subtraction operator is independent of translation of the coordinate system.

value of $l_i(j)$ was set to i with the probability Pr while it was set through a tournament selection to select the value of $l_i(j)$ with probability $1 - Pr$. Results on 16 benchmark test functions showed that the algorithm outperforms several other PSO variants on multimodal functions, while it performs almost the same as the others on unimodal functions. The discussion over the results was supported by the mean and variance of the found solutions.

The good performance of CLPSO on multimodal optimization problems stems from its effectiveness in exploration. Also, as mentioned, CLPSO does not perform as well on unimodal optimization problems, which reflects the weak exploitation ability of the algorithm. To improve the ability of the algorithm for exploitation, CLPSO was revised by Huang et al. (2010) in a way that it emphasized learning from elite particles (the particles with the best personal best vectors in the swarm). The new variant was called "Example-based Learning Particle Swarm Optimizer," ELPSO where Eq. (17) was revised as follows:

$$\vec{V}^i_{t+1} = \omega \vec{V}^i_t + \varphi_1 R^i_{1t}(\vec{P}^i_t - \vec{x}^i_t) + \varphi_2 R^i_{2t}(\vec{G}_t - \vec{x}^i_t), \tag{18}$$

where $\vec{P}^i_t$ is defined similarly to the one in CLPSO (except the strategy of setting $l_i(j)$ which was selected randomly by a uniform distribution over all particles) and each dimension j of $\vec{G}_t$ (shown by $\vec{G}^j_t$) is randomly taken from a set of previously sampled positions by $\vec{g}_t$. It was shown that the searching interval of each dimension[25] by each particle is larger than what it was in CCPSO and CLPSO, which indicates a better diversification in the swarm. Experiments were conducted on 16 benchmark test functions and the t-test was used to compare the results. These experiments showed that ELPSO is more effective than CLPSO in multimodal and unimodal optimization problems. A very similar approach to ELPSO was also proposed in Nasir et al. (2012).

To prevent SPSO from premature convergence,[26] the velocity update rule of SPSO was revised (Xinchao, 2010) and a method called "perturbed Particle Swarm Algorithm," pPSA, was proposed. In pPSA, the vector $\vec{g}_t$ in the velocity update rule of the standard PSO was substituted for $N(\vec{g}_t, \sigma^2 I)$, where N is the normal distribution and σ is the standard deviation. The value of σ was set through a very simple time-varying strategy. The results of applying the algorithm to some standard unimodal/multimodal benchmarks showed that the algorithm performs better than SPSO in terms of the quality of solutions and robustness. In comparison to GCPSO, results showed that both methods perform almost the same on multimodal functions while GCPSO performs better than pPSA in unimodal functions. Results were compared based on mean, median, and standard deviation, which appears sufficient to show the advantages. No theoretical analyses were included in that article; however, it seems the algorithm can escape stagnation and guarantees local convergence as it uses a mutated global best vector with non-zero variance. In addition, the algorithm is the same as SPSO in terms of transformation invariance, that is, rotation variance with scale and translation invariance—see Bonyadi and Michalewicz (2014a) for the proof.

[25]The term "searching interval of a dimension" refers to the average of size of the interval that is sampled by the optimization algorithm in a dimension. The larger the value of searching interval, the more diverse the individuals have been.

[26]The term "premature convergence" is usually misused in literature and different researchers define it differently. It sometimes refers to converging to a local optimum and addressing stagnation, while sometimes it refers to escaping from a local optimum and finding better solutions that might not necessarily be a local optimum, but better than what was found previously. In this context, it refers to escaping stagnation and converging to a local optimum.

Diversity of particles is related to the explorative and exploitative behavior of the particles. An algorithm called "Diversity enhanced with Neighborhood Search Particle Swarm Optimization," DNSPSO, was proposed (Wang et al., 2013) in which the explorative behavior was controlled by enhancing the diversity of the particles in the swarm. At each iteration, the position and velocity of each particle is updated by the rules in SPSO. The new position $\vec{x}_{t+1}^{i}$ is then combined with $\vec{x}_{t}^{i}$ to generate a trial particle. This is done by taking either the value of $x_t^{i,j}$ or $x_{t+1}^{i,j}$ for each $j \in \{1, \ldots, d\}$ with probability Pr. The personal best of the particle i is updated according to the trial particle. Also, two local searches were used to search the neighborhood of the personal best and global best vectors, which actually improved the exploitation ability of the algorithm. The usage of these neighborhood search strategies was shown to be beneficial for accelerating the convergence rate (experimentally tested). The algorithm was applied to 30 benchmark functions, 10 of them with up to 50 dimensions and 20 of them with 1,000 dimensions, and the results were compared with other methods such as CLPSO, APSO, and CPSO using Friedman test. Comparisons showed that the algorithm is comparable with others for both groups of test functions.

There have been some other modifications proposed for update rules of SPSO (Krink et al., 2002; Chang-Huang and Jia-Shing, 2009; Kaucic, 2013). In Krink et al. (2002) a strategy was proposed to maintain the level of diversity of the particles; in Chang-Huang and Jia-Shing (2009) SPSO without CI component was compared with SPSO without an SI component; and Kaucic (2013) experimented with a multi-start method. However, these studies have been excluded from this review because of the selection criteria defined in Section 1.

Summary: Many approaches have been proposed to improve the velocity update rule. For example, Van den Bergh and Engelbrecht (2004) proposed a cooperative coevolution approach (CPSO) in which the problem was divided into several subproblems and each subproblem was solved by a separate SPSO. Although the idea is somewhat interesting, experimental results in that study were very limited which makes the reader wonder if the method can perform well on other optimization problems. In addition, from a theoretical perspective, the proposed method is not guaranteed to converge to a local optimum as it is constructed from SPSO that is not locally convergent by itself (a stagnation scenario was also presented in that paper that is another support to show that the algorithm is not locally convergent). Further, as the sub-swarms are instances of SPSO, the algorithm is sensitive to rotation. This can be also concluded from the presented results where the performance of method was tested when the search space was rotated. Recently, comprehensive learning approaches in PSO (CLPSO) have attracted a lot of interest (Liang et al., 2006; Huang et al., 2010; Tang et al., 2011; Changhe et al., 2012). The reason is that these approaches have shown a better exploration ability (Liang et al., 2006) compared to SPSO in experiments. From a theoretical perspective, it is apparent that the algorithm (and its derivative, ELPSO) can be stagnated and, hence, it is not locally convergent (if all particles are at the same point in the search space). Also, the algorithm is not rotationally invariant as it uses the random diagonal matrices the way it was used in SPSO. This can be also observed by comparing the results reported in that paper for rotated and original test functions. Some approaches have tried to make more use of the location of the global best vector with the hope of finding better solutions in fewer iterations. For example, Arumugam et al. (2009) proposed a method (ePSO) in which the position of the global best vector was "extrapolated" (in relation to the current position of the particle) to find new solutions. This method, however, was sensitive to the shift of the coordinate axes that makes it in appropriate for optimization

in general. Another idea was to mutate the global best vector (Xinchao, 2010) during the run to avoid premature convergence. This method can address the local convergence issue while still being sensitive to the rotation of the search space.

There are two main challenges in cooperative co-evolution: how to divide the problem (search space) into subproblems (decomposition) and which type of PSO should be applied to which subproblem. For example, Van den Bergh and Engelbrecht (2004) divided the problem into d subproblems (each dimension was considered as one subproblem) and SPSO was used to deal with each subproblem. However, this choice causes the algorithm to have the local convergence and transformation variance issues. To address the decomposition issue in cooperative co-evolution approaches one can consider the approach by Omidvar et al. (2014) to decompose the space so that the emergence of pseudominima is prevented. In addition, the cooperative co-evolution approaches can benefit from a locally convergent and transformation invariant algorithms (e.g., LcRiPSO in Bonyadi and Michalewicz, 2014b) to deal with subproblems. It is also worthwhile to investigate comprehensive learning approaches from a theoretical perspective to understand why they can offer a better exploration compared to SPSO and how their exploitation behavior can be enhanced. Also, these methods can be studied from a convergence and invariance point of view to design a PSO variant with enhanced and scalable performance. Another potential direction for further study is to investigate differences between PSO, DE, and ES methods. Once these differences are identified, it would be possible to borrow ideas from ES and DE methods to improve the velocity/position update rules in PSO. In addition, because both update rules in PSO contain vector calculations, it would be interesting to run these calculations in hyper-spherical coordinates rather than Cartesian coordinates. One benefit of hyper-spherical coordinates might be that calculations of vector rotations are easier, which potentially enables the algorithm to deal with non-separable problems more efficiently.

4.3 Hybridization

Optimization methods have their merits and drawbacks in dealing with different optimization problems. For example, the Newton's method is locally convergent (under some assumptions; see Kelley, 1999) and usually is able to find a local optimum fast (i.e., effective exploitation). Also, stochastic search methods are usually effective in locating many basins of attraction (effective exploration). The aim of a hybrid method is to combine different optimization methods to take advantage of the merits of each of them. In this subsection, articles that have combined PSO with other methods are reviewed. Many combinations are possible, for instance using one method to adjust parameters of the other (see Parsopoulos and Vrahatis, 2002b and Alatas et al., 2009 where DE and a chaotic map were used to set the coefficients of SPSO during the run, respectively), or running different methods iteratively to improve the outcome of one another—see Kao and Zahara (2008), a study in which a GA was combined with SPSO.

Zhan et al. (2011)[27] investigated two issues, namely "oscillation" of particles (as each particle is attracted by two vectors, i.e., personal best and global best vectors, it might keep oscillating between the two) (Parsopoulos and Vrahatis, 2004) and "two steps forward, one step back"(see Section 4.2 for details on this issue). To address these two issues, it was proposed to combine global and personal best and use the combined vector in the velocity update rule instead. In order to combine these two vectors, the

[27]Note that these two issues were not considered in Section 3 as they have not been properly supported from a theoretical point of view in the related articles.

"Orthogonal Experimental Design," OED, approach was applied (Montgomery, 1984) (the new PSO variant was called "Orthogonal Learning Particle Swarm Optimization," OLPSO). The combined position was then used in the velocity update rule exactly as in CLPSO (see Eq. 17) where $\vec{P}_i^t$ was replaced by the vector generated by OED approach. This strategy was tested on SPSO and LPSO where results on 16 test cases showed significant improvement (based on the *t*-test) on the performance of these algorithms when OED is used. The optimal solution of most of the tested functions (11 out of all 16) was at the center of the coordinate system.

Different strategies (related to PSO) have been proposed to improve different components of the algorithm (e.g., a better topology and adaptive coefficients). Thus, it might be beneficial if best strategies for updating each component are taken from previous studies and combined to generate a new PSO variant. This was done in Montes de Oca et al. (2009) and the method was called "Frankenstein's Particle Swarm Optimization," FPSO. The main purpose of the method was to combine different PSO variants together and create a new method that enables them to overcome each other's deficiencies. FPSO components were the following: the inertia weight setting was taken from DIPSO, the velocity was updated by FIPS formulation, acceleration coefficients and setting of V_{max} were set by the method used in HPSO-TVAC, and the topology was set by the method proposed in Janson and Middendorf (2005). Experimental results showed that FPSO is effective and outperforms the methods it has been composed of. Also, the same idea was tested in Tang et al. (2011) and a variant called "Feedback Learning Particle Swarm Optimization," FLSPO, was proposed. FLPSO used DIPSO for the inertia weight, an adaptive approach for φ_1 and φ_2 (similar to HPSO-TVAC with some modification), an adaptive strategy to use personal best or global best vector in the velocity update rule, and a mutation operator applied to a randomly selected dimension of the global best vector. As such a mutation operator prevents stagnation, it is very likely that FLPSO is locally convergent.

As there are many different position/velocity update rules and each of them contain their pros and cons in different situations, it would be beneficial to investigate which update rule is most beneficial at the current situation (iteration) and use that strategy for further iterations. For example, a self-adaptive method was proposed in which the most effective PSO variant was selected during the run for the problem at hand (Wang et al., 2011). This variant was called "Self-Adaptive Learning Particle Swarm Optimization," SALPSO, where four velocity update rules (taken from different PSO variants) were considered and the best update rule was selected to be used in each iteration. As each velocity update rule had its own capabilities in different situations (e.g., for exploration or exploitation), it was expected that the overall performance of the algorithm is improved. For each particle and every G iterations (a constant experimentally set to 10), the probability of using each update rule was updated based on the improvement it offered during the last G iterations. The velocity update rules were selected for each particle at each iteration to update the particles according to these probabilities. Experiments (based on average and standard deviation) showed that SLPSO is effective in dealing with both unimodal and multimodal problems. A very similar idea was also presented by Changhe et al. (2012) where the method was called "Self-Learning Particle Swarm Optimization," SLPSO. The only difference between SLPSO and SALPSO was in the implementation details such as how to update probabilities and which velocity update rules are used.

The ideas used in SALPSO and SLPSO triggered the emergence of another PSO variant called "Multiple Adaptive Methods for Particle Swarm Optimization,"

PSO-MAM (Hu et al., 2012). In PSO-MAM, all particles were updated by the update rules in SPSO except the global best particle. To update the global best particle, two approaches, a mutation and a gradient descent approach, were designed and one of them was selected randomly (based on a probability distribution that was updated according to the current performance) at each iteration. An extensive experiment was conducted (31 functions and comparison with PSO variants such as CPSO and CLPSO) that showed PSO-MAM is comparable with other PSO variants. We conjecture that the algorithm is locally convergent as it uses a mutation operator based on a Cauchy distribution with the center of the current position and a non-zero scale parameter.

There are some other high-quality articles related to the hybridization of PSO such as Muller et al. (2009), where SPSO was combined with CMA-ES; however, these articles have been excluded from this review according to the selection criteria defined in Section 1.

Summary: It can be beneficial to combine different methods in such a way that they overcome each other's shortcomings. For example, Hu et al. (2012) combined PSO with gradient descent and a mutation operator that were applied randomly to the global best particle. Of course the local convergence and rotation invariance of such methods is dependent on the components they are composed of. For example, in Hu et al. (2012), we conjecture that the algorithm is locally convergent as it uses a mutation operator based on a Cauchy distribution with the center of the current position and a non-zero scale parameter. Zhan et al. (2011) experimented with an idea based on replacing personal best and global best vectors with a combination of the both to improve the performance of the algorithm. That study used a method called OED for this combination and experimentally showed it can improve the performance of the LPSO and SPSO algorithms. Experiments in that study were, however, biased toward test functions where their optimum solution is at the center of the coordinate systems. It seems that the method is sensitive to the rotation of the search space as can be observed in the reported results in the paper. Another approach for hybridization is to design one method while taking its different components (e.g., topology adaptation and coefficients adaptation) from different PSO variants (e.g., Montes de Oca et al., 2009 and Tang et al., 2011). These strategies were shown experimentally to be effective on the benchmark functions. The local convergence and rotation variance of these methods are, again, dependent on the components they use for the update rules. For example, as a gradient based approach is used in Tang et al. (2011), it is likely that the algorithm is locally convergent. Another hybridization approach found in PSO literature is related to implementing several algorithms and selecting the best algorithm during the run (using some adaptive selection probability). Wang et al. (2011) and later Changhe et al. (2012) used this idea and selected different update rules randomly during the run while the probability of selection was updated.

One potential direction related to hybridization in PSO is to investigate other approaches for combining personal best and global best vectors (e.g., by a crossover operator) and determine which combination approaches can be beneficial for different landscapes. Also, investigation of the selection method to select appropriate variants during the run can be worthwhile for improvement of the algorithm performance. For example, a method can be used to recognize the landscape characteristics (e.g., rugged, smooth, etc.—see Malan and Engelbrecht, 2009, for example) and select the most appropriate update rule accordingly. One should note that some methods might impose extra time overhead to the algorithm that makes the time complexity analysis an essential part of such articles.

5 Modification of PSO to Deal with Constrained Optimization Problems (COP)

A single objective COP is defined by

$$\text{find } \vec{x} \in \mathcal{F} \subseteq S \subseteq \mathbb{R}^d \text{ such that } \forall \vec{y} \in \mathcal{F},\ f(\vec{x}) \leq f(\vec{y}), \tag{19}$$

where $\mathcal{F} = \{\vec{y} : g_j(\vec{y}) \leq 0 \text{ and } h_k(\vec{y}) = 0 \text{ for all } j \in \{1, \ldots, p\} \text{ and } k \in \{1, \ldots, q\}\}$, g_j are inequality constraints and h_k are equality constraints, both of them are $\mathbb{R}^d \to \mathbb{R}$ for all j and k, and S is defined the same as the one for UOPs (see Eq. 1).

Although comparing two solutions a and b in a UOP is usually easy, this comparison is not straightforward in COPs. For example, comparing two infeasible solutions or a feasible and an infeasible solution may depend on many parameters such as the shape of the objective function and constraints. Some methods involve modification of the solutions so that they become comparable. For example, in "repair" (a constraint handling method; see Michalewicz and Schoenauer, 1996), solutions are first modified according to a predefined procedure to make them feasible and then the feasible solutions are compared based on their objective values. Bonyadi and Michalewicz (2014b) provided further discussion on the complexity of comparisons in COPs.

There have been a few attempts to extend PSO to deal with COPs. One of the first studies in this area was conducted in Parsopoulos and Vrahatis (2002a) where a penalty function was incorporated into SPSO to deal with constraints. The method was applied to several benchmark functions of COP and its results were compared with a variant of ES introduced in Yang et al. (1997). Results showed that SPSO outperforms ES on many benchmark COPs. This result encouraged researchers to test other PSO variants with different constraint handling methods, for example, preserving feasibility of solutions (Hu and Eberhart, 2002), closeness to the feasible region (based on the constraint violation value) (Pulido and Coello, 2004), and penalty functions (Coath and Halgamuge, 2003), on more complex COPs.

LPSO was modified (Paquet and Engelbrecht, 2007) and applied to COPs with linear equality constraints (recall that LPSO was a variant of PSO in which the random matrices, R_{1t} and R_{2t}, were replaced by random values, r_{1t} and r_{2t}). Linear equality constraints are formulated as $A\vec{x} = \vec{b}$ where A is a m by d matrix, $\vec{x}$ is a d dimensional vector, and $\vec{b}$ is a m dimensional vector. It was proven that if all personal best and velocities of particles are initialized such that $Ap_i^0 = b$ and $Av_i^0 = 0$, then all further generated solutions by LPSO are feasible. Hence, LPSO is ideal to search through linear equality constraints. Although results showed that LPSO performs well in dealing with linear equality constraints, the algorithm suffered from the line search and stagnation issues. To overcome these issues, the velocity update rule for the global best vector was modified and another type of LPSO called "Guaranteed Converging Linear Particle Swarm Optimization," GCLPSO, was proposed (Paquet and Engelbrecht, 2003, 2007). The idea was similar to what was discussed for GCPSO with some modifications to guarantee feasibility after mutation. Experiments with a small number of benchmark functions showed that GCLPSO on some problems were comparable with that of Genocop II (Michalewicz and Janikow, 1996) (a GA-based optimization approach for COPs).

In 2007 a "Co-evolutionary Particle Swarm Optimization for Constraint Optimization Problems," CPSO-COP, approach was proposed (He and Wang, 2007). SPSO was used as an optimizer in CPSO-COP while a new method to handle constraints was proposed. Two swarms were considered in CPSO-COP, one to store penalty coefficients (swarm S_1 with the size n_1) and the other to store solutions (swarm S_2). Each particle in S_1 provided penalty coefficients to evaluate particles in S_2. S_2 contained n_1 sub-swarms,

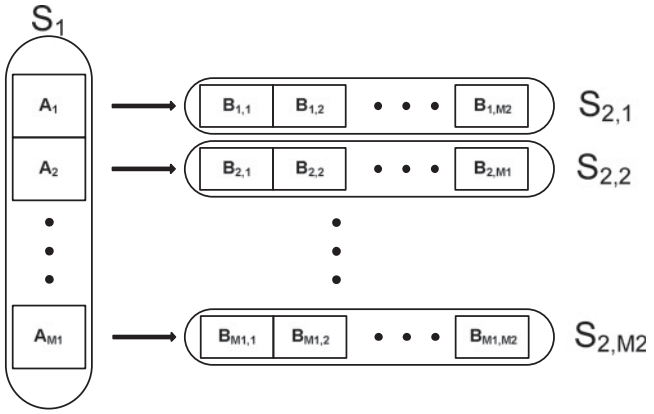

Figure 6: A_i is the i^{th} particle of the swarm S_1, which provides penalty factors for swarm $S_{2,i}$. M1 is the size of the swarm type one (S_1) and M2 is the size of each swarm of type two (each $S_{2,i}$).

the sub-swarm i was evaluated based on the coefficients provided by the particle i in S_1 (see Fig. 6). At each generation of the co-evolutionary process, every $S_{2,i}$ (sub-swarm i of the S_2) was evolved using SPSO for a certain number of generations and it used particle i from S_1 for evaluation. After the $S_{2,i}$ were evolved for one iteration, S_1 was evolved using SPSO also for one iteration. This process was re-run until a termination criterion was met. The authors compared their results with GA-based optimization methods for solving several engineering COP benchmarks. Experimental results showed that CPSO-COP converges to better solutions (in terms of the mean, min, max, standard deviation of the results) in a fewer number of iterations in comparison to the GA (Coello Coello, 2000).

A "Cooperation Comprehensive Learning Particle Swarm Optimization," Co-CLPSO, was implemented and applied to COPs (Liang et al., 2010). The comprehensive learning approach in PSO (e.g., see the CLPSO description in Section 4.2) was used as an optimization method to solve COPs. Also, to handle the constraints, particles were adaptively assigned to either satisfy different constraints or optimize the objective value. A local search based on "Sequential Quadratic Programming," SQP, was used to improve the solutions during the run. This approach received the fourth place in the competition of CEC 2010 in solving COPs (Mallipeddi and Suganthan, 2010).

A PSO variant was proposed by Sun et al. (2011) to deal with COPs. In that method, it was suggested to update the position of the particles by $x^i_{t+1} = x^i_t + M v^i_{t+1}$ where M is a $d \times d$ diagonal matrix where each element on the diagonal of M belongs to [0, 1]. If x^i_{t+1} for $M = I$ (I is the identity matrix) is feasible then the new position is accepted; otherwise, another SPSO instance is run to find a matrix M in such a way that x^i_{t+1} is a high-quality feasible point. This instance of SPSO in fact searches through a hyper-cube where x^i_t is a vertex and v^i_{t+1} is its diagonal. The comparison results (based on best, worst, and average of the found solutions on a standard benchmark of COPs with 13 test cases) showed that this method is comparable with other PSO variants.

A multi-start PSO was proposed (Bonyadi et al., 2013) where a mutation operator was added to LPSO to address the stagnation, line search, and swarm size issues (this variant was called "Mutation Linear Particle Swarm Optimization," MLPSO). MLPSO

was then used as an optimizer to solve COPs while a simple approach was used to compare solutions. It was assumed that any feasible solution is better than any infeasible solution, feasible solutions are compared based on their objective value, and infeasible solutions are compared based on their constraint violation value (calculated by $\sum_{i=1}^{p} max\{0, g_i(x)\} + \sum_{i=1}^{q} max\{0, h_i(x)\}$). This was indeed a simplified version of "Epsilon Level Constraint Handling," ELCH (Takahama and Sakai, 2005). Also, a method based on the CMA-ES was proposed that used the same technique to handle constraints. Experiments showed that MLPSO has a better performance in finding feasible solutions while CMA-ES performs better in optimizing the objective function. Thus, a hybrid method was proposed that ran PSO to find the first feasible solution and then that solution was improved by CMA-ES. However, to prevent MLPSO from finding feasible solutions in a poor-quality feasible region, a multi-start strategy was incorporated in the algorithm so that several instances of MLPSO were run one after another to generate different feasible solutions. This idea potentially leads to locating disjoint feasible regions that increases the probability of finding the region which includes the optimal solution. Then the best among those solutions were fed into CMA-ES for further improvement. MLPSO was further enhanced in Bonyadi, Li, and Michalewicz (2014) with a time-adaptive topology rather than a multi-swarm strategy. Also, the idea of locating disjoint feasible regions in a COP was further discussed in Bonyadi and Michalewicz (2014b).

A PSO algorithm for solving COPs called "Automatic Particle Injection PSO," API-PSO, was proposed (Elsayed et al., 2013) in which the balance between exploration and exploitation was controlled during the run by using LPSO or SPSO in each iteration. The choice between the two methods was made based on a random value, which favored SPSO at the beginning and LPSO at the end of the run. The idea was to emphasize exploration in the earlier iterations and exploitation in the later iterations. Also, a mutation operator and an automatic injection of new particles were added to prevent the algorithm from getting stuck in a local optimum. Further, the inertia weight and acceleration coefficients were considered as variables and they were fed into the position vector of particles ($\vec{x}$) so that they could be set during the run by the algorithm (a self-adaptive approach). In order to handle constraints, API-PSO was extended by ELCH. Results (based on the average values) showed that the algorithm is comparable with other methods to deal with COPs.

In many real-world COPs it is highly probable that some constraints are active at optimum points (Schoenauer and Michalewicz, 1996); that is, some optimum points are on the boundary of the feasible area of the search space. The reason is that constraints in real-world problems often represent limitations of resources, such as rate, number of resources, etc. Thus, it is usually beneficial to make use of some resources as much as possible, which means that some constraints are active at high quality solutions. Such resources that limit the optimization algorithm to achieve better solutions (i.e., separate the feasible and infeasible spaces) are called bottlenecks (Bonyadi, Michalewicz, and Wagner, 2014). Bottlenecks are usually important to be found in a system because the best investment from companies should be concentrated on the bottlenecks to maximize profit (see Bonyadi, Michalewicz, and Wagner, 2014 and Chatterjee and Mukherjee, 2006 for a complete discussion on bottlenecks, investment, and their relations to optimization and constraints). The presence of active constraints at the optimum points causes difficulty for many optimization algorithms in locating optimal solutions (Schoenauer and Michalewicz, 1997). In order to address this issue, Bonyadi and Michalewicz (2014c) proposed a function (called "Constraints Boundary Narrower," CBN) and proved that

it can transform the feasible region of any COP into the edges of the constraints of that COP. Hence, if the optimization method is applied to the transformed COP, the solutions that the method finds are at the edges of feasibility of the original COP. Also, that article provided a more flexible definition for the edge and boundaries of feasibility and active constraints. The proposed definitions enabled the study to assign a degree of activity to constraints rather than categorizing them only as active and inactive. In that study, SPSO was used as an optimizer and CBN (together with some of its derivatives) was tested on seven COPs where their optimum solution was on the boundary of the feasible area. Results (based on the average of the final solutions) showed that this combination is effective in finding optima when they are close to the edges of feasible area.

Summary: There have been a few attempts to extend PSO to deal with COPs, such as the usage of penalty functions (Parsopoulos and Vrahatis, 2002a; Takahama and Sakai, 2005) or preservation of feasibility (Coath and Halgamuge, 2003). Also, Liang et al. (2010) used a cooperative approach together with comprehensive learning to deal with COPs. This approach was further enhanced with a local search to find better solutions. However, as many different components were introduced in that method, it is not possible to find which component is really responsible for the final results. One interesting field of research related to COPs is the idea of searching the edges of feasible regions proposed by Schoenauer and Michalewicz (1996). This was studied by Bonyadi and Michalewicz (2014c) for PSO; however, the results reported were not tested properly through statistical tests. Another interesting area related to COPs is locating disjoint feasible areas (Bonyadi, Li, and Michalewicz, 2014; Bonyadi and Michalewicz, 2014b; Smith et al., 2013). In fact, the feasible space might be of an irregular shape and might include many disjoint areas in real-world applications, hence, it is worthwhile to locate these areas first and then determine which one has a higher potential to include the optimal solution. This was studied in Bonyadi and Michalewicz (2014b) for PSO on a limited number of test problems.

There are several potential future directions related to COPs. For example, there have not been many attempts to extend PSO to search the edge of the feasible region, which can be considered as a future research direction. Another area that has been recently investigated is to locate disjoint feasible regions in a COP. In addition to the edges of feasibility and locating disjoint feasible regions, dealing with dynamic COPs (Nguyen and Yao, 2012) can be also investigated. As constraints formulate limitations of resources and these limitations might change in time because of random events in the real world, dynamic constraints are regularly found in real-world optimization problems. Another potential area of research is to investigate the performance of PSO variants to deal with COPs through theoretical analysis, for example, first hitting time, convergence, and stability. This track of research had been started by Baba (1981), but to the best of our knowledge, remained untouched for optimization algorithms in the field of evolutionary computation.

6 Conclusions

New advances in the PSO algorithm related to its convergence properties, transformation invariance, modification of components, coefficients adaptation, population sizing, topology, hybridization, and dealing with constraint optimization problems, were reviewed in this survey. For each of these topics, several articles were cited and, at the end of each subsection, a summary of the reviewed articles was given where the potential challenges and research directions related to that area were discussed.

The first part of this article concentrated on limitations of PSO that have been theoretically investigated in the past years. These limitations were categorized into two main groups: convergence and transformation invariance. Regarding the convergence properties of PSO, it was pointed out that analyzing different behaviors of particles before convergence and convergence analysis (including convergence to a fixed point, local convergence, and stagnation) of other PSO variants (e.g., SPSO2011 and RPSO) constitute potential research directions (Section 3.1). Also, there are not many studies on the first hitting time analysis of PSO variants, which is another potential research direction (Section 3.1.4). Additional important area of study in PSO is the transformation invariance property of the algorithm. Indeed, there are not many PSO variants that are transformation invariant, which makes this topic an open area for new ideas and analyses (Section 3.2). Note that, as has been already mentioned in some articles (Auger et al., 2009; Hansen et al., 2008), these properties (convergence and transformation invariance) are important for an algorithm since without these properties it is impossible to scale its good performance achieved on a test set to a wide variety of problems. Moreover, many other derivative-free optimization methods (e.g., CMA-ES) have addressed these issues successfully (Auger et al., 2009).

Apart from theoretical studies, many experimental studies related to modifications of components and parameters of PSO were also reviewed. Topology of the swarm (Section 4.1.1), setting coefficients (Section 4.1.2), and population size (Section 4.1.3) are parameters of PSO that were investigated in this review. Also, articles that modified the velocity update rule of PSO were analyzed in Section 4.2. Some other studies that hybridized the algorithm with other optimization methods to improve the overall performance of the designed optimizer were discussed in Section 4.3. Finally, applications of PSO to deal with COPs were investigated (Section 5). Some of the potentially promising research directions related to experimental studies were discussed. One of these was the usage of landscape characteristics to determine the current state of the search and then, based on this information, decide how to update the topology, coefficients, or velocity itself. Another potential direction was to investigate the theoretical aspects of the current experimental approaches to see if the good performance reported in those articles can be supported more formally (see summaries of these sections for additional possible directions). We also reported the computational complexity of some of the methods included in the experimental part of this survey. It seems that some modifications of the algorithm, such as hybridization or adaptive parameter control, may introduce a computational overhead that may significantly increase the computational complexity of the algorithm. For example, some methods determine the search state (e.g., exploration vs. exploitation) using the distances among all particles in the swarm which slows down the algorithm significantly. Similar issues are present when comparing methods to handle constraints, as some of these methods (e.g., repair methods) are computationally expensive. However, such computational overhead is usually ignored when comparing algorithms as the number of function evaluations remains the main criterion.

Despite the fact that the articles reviewed in this survey were selected from high-ranked conferences and journals, many of them (especially those included in the experimental part of this survey) presented only experimental results and did not provide adequate discussion (neither from theoretical perspective, nor general discussions) on the merits of the proposed approach. In fact, most articles aimed only at outperforming other methods on a set of benchmark functions, missing the point that the main aim in black-box optimization is to design a method that has a good *scalable* performance.

Therefore, there is a need for a discussion in these experimental papers to clarify why the authors think that the new method is going to work better than previous ones before they apply it to a set of benchmarks and if there are any particular situations that the method might have merits/drawbacks and why. The lack of such discussion becomes more severe when appropriate statistical analyses of the results are not presented, arbitrary benchmarks are used, or only a small number of test cases are used. Further, in some cases, the set of test cases were biased (see Sections 4.1.1 and 4.1.2 for some examples) that prevented readers from drawing generic conclusions. This in fact invalidates many claims by the authors and may mislead other researchers, although the ideas themselves may have been original and potentially beneficial. In addition, in some articles, many ideas have been experimented with and many components have been added/modified that may confuse the reader, as it is not clear which component is responsible for a good performance of the algorithm and why (see Sections 4 and 5 for some examples). Such an issue is especially visible in hybrid approaches where the result method needs to be shown (through proper statistical tests or theoretical analyses) to outperform each of its constituent methods.

Acknowledgments

The authors would like to extend their appreciation to Professor James Kennedy who provided detailed comments on earlier versions of this article. We would also like to thank Dr. Bradley Alexander for his assistance on improving readability of the article. This work was partially funded by the ARC Discovery Grant DP130104395.

References

Agrafiotis, D. K., and Cedeno, W. (2002). Feature selection for structure-activity correlation using binary particle swarms. *Journal of Medicinal Chemistry*, 45(5):1098–1107.

Alatas, B., Akin, E., and Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solutions and Fractals*, 40(4):1715–1734.

Albert, R., and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Angeline, P. J. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *Evolutionary Programming VII*, pp. 601–610. Berlin: Springer.

Arabas, J., Michalewicz, Z., and Mulawka, J. (1994). GAVaPS-a genetic algorithm with varying population size. In *IEEE World Congress on Computational Intelligence*, pp. 73–78.

Arumugam, M. S., Rao, M., and Chandramohan, A. (2008). A new and improved version of particle swarm optimization algorithm with global–local best parameters. *Knowledge and Information Systems*, 16(3):331–357.

Arumugam, M. S., Rao, M. V. C., and Tan, A. W. C. (2009). A novel and effective particle swarm optimization like algorithm with extrapolation technique. *Applied Soft Computing*, 9(1):308–320.

Auger, A., Hansen, N., Zerpa, J. P., Ros, R., and Schoenauer, M. (2009). Empirical comparisons of several derivative free optimization algorithms. In J. Vahrenhold (Ed.), *8th International Symposium on Experimental Algorithms*, volume 5526 of LNCS, pp. 3–15. Berlin: Springer.

Baba, N. (1981). Convergence of a random optimization method for constrained optimization problems. *Journal of Optimization Theory and Applications*, 33(4):451–461.

Banks, A., Vincent, J., and Anyakoha, C. (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing*, 6(4):467–484.

Banks, A., Vincent, J., and Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1):109–124.

Beyer, H.-G., and Schwefel, H.-P. (2002). Evolution strategies—A comprehensive introduction. *Natural Computing*, 1(1):3–52.

Birbil, Ş. İ., Fang, S.-C., and Sheu, R.-L. (2004). On the convergence of a population-based global optimization algorithm. *Journal of Global Optimization*, 30(2–3):301–318.

Blackwell, T. (2012). A study of collapse in bare bones particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 16(3):354–372.

Bonyadi, M. R., Li, X., and Michalewicz, Z. (2013). A hybrid particle swarm with velocity mutation for constraint optimization problems. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1–8.

Bonyadi, M. R., Li, X., and Michalewicz, Z. (2014). A hybrid particle swarm with a time-adaptive topology for constrained optimization. *Swarm and Evolutionary Computation*, 18:22–37.

Bonyadi, M. R., and Michalewicz, Z. (2012). A fast particle swarm optimization algorithm for the multidimensional knapsack problem. In *IEEE Congress on Evolutionary Computation*, pp. 1–8.

Bonyadi, M. R., and Michalewicz, Z. (2014a). A locally convergent rotationally invariant particle swarm optimization algorithm. *Swarm Intelligence*, 8(3):159–198.

Bonyadi, M. R., and Michalewicz, Z. (2014b). Locating potentially disjoint feasible regions of a search space with a particle swarm optimizer. In K. Deb and R. Datta (Eds.), *Evolutionary constrained optimization*, pp. 205–230. Berlin: Springer.

Bonyadi, M. R., and Michalewicz, Z. (2014c). On the edge of feasibility: A case study of the particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, pp. 3059–3066.

Bonyadi, M. R., and Michalewicz, Z. (2014d). SPSO2011—Analysis of stability, local convergence, and rotation sensitivity. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 9–16.

Bonyadi, M. R., Michalewicz, Z., and Li, X. (2014). An analysis of the velocity updating rule of the particle swarm optimization algorithm. *Journal of Heuristics*, 20(4):417–452.

Bonyadi, M., and Michalewicz, Z. (2016). Analysis of stability, local convergence, and transformation sensitivity of a variant of particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 20(3):370–385.

Bonyadi, M., and Michalewicz, Z. (In Press). Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE Transactions on Evolutionary Computation*, In Press.

Bonyadi, M. R., Michalewicz, Z., and Wagner, M. (2014). Beyond the edge of feasibility: Analysis of bottlenecks. In G. Dick, W. Browne, P. Whigham, M. Zhang, L. Bui, H. Ishibuchi, Y. Jin, et al. (Eds.), *Simulated evolution and learning*, pp. 431–442. Berlin: Springer.

Bratton, D., and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pp. 120–127.

Caamao, P., Prieto, A., Becerra, J., Bellas, F., and Duro, R. (2010). Real-valued multimodal fitness landscape characterization for evolution. In K. Wong, B. Mendis, and A. Bouzerdoum (Eds.), *Neural information processing: Theory and algorithms*, pp. 567–574. Berlin: Springer.

Campana, E., Fasano, G., and Pinto, A. (2010). Dynamic analysis for the selection of parameters and initial population, in particle swarm optimization. *Journal of Global Optimization*, 48(3):347–397.

Carlisle, A., and Dozier, G. (2001). An off-the-shelf PSO. In *Workshop on Particle Swarm Optimization*, pp. 1–6. Indianapolis, Indiana: Purdue School of Engineering and Technology.

Chang-Huang, C., and Jia-Shing, S. (2009). Simple particle swarm optimization. In *International Conference on Machine Learning and Cybernetics*, pp. 460–466.

Changhe, L., Shengxiang, Y., and Trung Thanh, N. (2012). A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3):627–646.

Chatterjee, A., and Mukherjee, S. (2006). Unified concept of bottleneck. Technical Report W.P. No. 2006-05-01, Indian Institute of Management Ahmedabad, Research and Publication Department.

Chatterjee, A., and Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers and Operations Research*, 33(3):859–871.

Chen, W. N., Zhang, J., Chung, H. S. H., Zhong, W. L., Wu, W. G., and Shi, Y. H. (2010). A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300.

Chen, D. B., and Zhao, C. X. (2009). Particle swarm optimization with adaptive population size and its application. *Applied Soft Computing*, 9(1):39–48.

Cleghorn, C. W., and Engelbrecht, A. P. (2014a). A generalized theoretical deterministic particle swarm model. *Swarm intelligence*, 8(1):35–59.

Cleghorn, C. W., and Engelbrecht, A. P. (2014b). Particle swarm convergence: An empirical investigation. In *IEEE Congress on Evolutionary Computation*, pp. 2524–2530.

Cleghorn, C. W., and Engelbrecht, A. P. (2014c). Particle swarm convergence: Standardized analysis and topological influence. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle (Eds.), *Swarm intelligence*, pp. 134–145. Berlin: Springer.

Cleghorn, C. W., and Engelbrecht, A. P. (2015). Particle swarm variants: Standardized convergence analysis. *Swarm Intelligence*, 9(2):177–203.

Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pp. 1951–1957.

Clerc, M. (2006). *Particle swarm optimization*. Newport Beach, CA: Wiley-ISTE.

Clerc, M. (2011). Standard particle swarm optimisation from 2006 to 2011. Technical Report hal-00764996.

Clerc, M., and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

Coath, G., and Halgamuge, S. K. (2003). A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In *IEEE Congress on Evolutionary Computation*, pp. 2419–2425.

Coello Coello, C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127.

Cooren, Y., Clerc, M., and Siarry, P. (2009). Performance evaluation of tribes: An adaptive particle swarm optimization algorithm. *Swarm Intelligence*, 3(2):149–178.

Dasgupta, S., Das, S., Biswas, A., and Abraham, A. (2009). On stability and convergence of the population-dynamics in differential evolution. *AI Communications*, 22(1):1–20.

Deng, J.-L. (1989). Introduction to grey system theory. *The Journal of Grey System*, 1(1):1–24.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Dorea, C. (1983). Expected number of steps of a random optimization method. *Journal of Optimization Theory and Applications*, 39(2):165–171.

Eberhart, R., and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *International Symposium on Micro Machine and Human Science*, pp. 39–43.

Eberhart, R., and Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *IEEE Congress on Evolutionary Computation*, pp. 94–97.

Eberhart, R. C., and Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pp. 84–88.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141.

Eiben, A. E., and Rudolph, G. (1999). Theory of evolutionary algorithms: A bird's eye view. *Theoretical Computer Science*, 229(1):3–9.

Elsayed, S. M., Sarker, R. A., and Essam, D. L. (2012). Memetic multi-topology particle swarm optimizer for constrained optimization. In *IEEE Congress on Evolutionary Computation*, pp. 1–8.

Elsayed, S. M., Sarker, R. A., and Mezura-Montes, E. (2013). Particle swarm optimizer for constrained optimization. In *IEEE Congress on Evolutionary Computation*, pp. 2703–2711.

Emara, H. M. (2009). Adaptive clubs-based particle swarm optimization. In *American Control Conference*, pp. 5628–5634.

Engelbrecht, A. (2005). *Fundamentals of computational swarm intelligence*. Hoboken, NJ: Wiley.

Engelbrecht, A. (2012). Particle swarm optimization: Velocity initialization. In *IEEE Congress on Evolutionary Computation*, pp. 1–8.

Engelbrecht, A. (2013). Particle swarm optimization: Global best or local best. In *IEEE Congress on Computational Intelligence*, pp. 124–135.

García-Gonzalo, E., and Fernández-Martínez, J. L. (2014). Convergence and stochastic stability analysis of particle swarm optimization variants with generic parameter distributions. *Applied Mathematics and Computation*, 249:286–302.

Ghosh, S., Das, S., Kundu, D., Suresh, K., Panigrahi, B. K., and Cui, Z. (2010). An inertia-adaptive particle swarm system with particle mobility factor for improved global optimization. *Neural Computing and Applications*, 21(2):237–250.

Gong, Y. J., and Zhang, J. (2013). Small-world particle swarm optimization with topology adaptation. In *Genetic and Evolutionary Computation Conference*, pp. 25–32.

Hansen, N. (2000). Invariance, self-adaptation and correlated mutations in evolution strategies. In *Parallel Problem Solving from Nature*, pp. 355–364.

Hansen, N. (2006). The CMA evolution strategy: A comparing review. In J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea (Eds.), *Towards a new evolutionary computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pp. 75–102.

Hansen, N., and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE Congress on Evolutionary Computation*, pp. 312–317.

Hansen, N., Ros, R., Mauny, N., Schoenauer, M., and Auger, A. (2008). PSO facing non-separable and ill-conditioned problems. Technical Report 6447, INRIA, France.

Hansen, N., Ros, R., Mauny, N., Schoenauer, M., and Auger, A. (2011). Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing*, 11(8):5755–5769.

He, J., and Yao, X. (2002). From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495–511.

He, Q., and Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20(1):89–99.

Helwig, S., Branke, J., and Mostaghim, S. (2013). Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17(2):259–271.

Helwig, S., and Wanka, R. (2007). Particle swarm optimization in high-dimensional bounded search spaces. In *Swarm Intelligence Symposium*, pp. 198–205.

Helwig, S., and Wanka, R. (2008). Theoretical analysis of initial particle swarm behavior. In *Parallel Problem Solving from Nature*, pp. 889–898.

Hsieh, S. T., Sun, T. Y., Liu, C. C., and Tsai, S. J. (2009). Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):444–456.

Hu, M., Wu, T., and Weir, J. D. (2012). An intelligent augmentation of particle swarm optimization with multiple adaptive methods. *Information Sciences*, 213:68–83.

Hu, X., and Eberhart, R. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. In *World Multiconference on Systemics, Cybernetics and Informatics*, pp. 203–206.

Hu, X., Shi, Y., and Eberhart, R. (2004). Recent advances in particle swarm. In *IEEE Congress on Evolutionary Computation*, pp. 90–97.

Huang, H., Qin, H., Hao, Z., and Lim, A. (2010). Example-based learning particle swarm optimization for continuous optimization. *Information Sciences*, 182(1):125–138.

Jägersküpper, J. (2008). Lower bounds for randomized direct search with isotropic sampling. *Operations Research Letters*, 36(3):327–332.

Janson, S., and Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(6):1272–1282.

Jiang, M., Luo, Y., and Yang, S. (2007a). Particle swarm optimization-stochastic trajectory analysis and parameter selection. In F. Chan and M. KumarTiwari (Eds.), *Swarm intelligence, Focus on ant and particle swarm optimization*, pp. 179–198. Wien: I-TECH Education and Publishing.

Jiang, M., Luo, Y. P., and Yang, S. Y. (2007b). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102(1):8–16.

Kao, Y.-T., and Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied Soft Computing*, 8(2):849–857.

Kaucic, M. (2013). A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. *Journal of Global Optimization*, 55(1):165–188.

Kelley, C. T. (1999). *Iterative methods for optimization*. Philadelphia: Society for Industrial and Applied Mathematics.

Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *IEEE Congress on Evolutionary Computation*, pp. 1931–1938.

Kennedy, J. (2003). Bare bones particle swarms. In *Swarm Intelligence Symposium*, pp. 80–87.

Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. In *International Conference on Neural Networks*, pp. 1942–1948.

Kennedy, J., and Mendes, R. (2002). Population structure and particle swarm performance. In *IEEE Congress on Evolutionary Computation*, pp. 1671–1676.

Krink, T., VesterstrOm, J. S., and Riget, J. (2002). Particle swarm optimisation with spatial particle extension. In *IEEE Congress on Evolutionary Computation*, pp. 1474–1479.

Lehre, P. K., and Witt, C. (2013). Finite first hitting time versus stochastic convergence in particle swarm optimisation. In *Advances in Metaheuristics*, pp. 1–20.

Leu, M.-S., and Yeh, M.-F. (2012). Grey particle swarm optimization. *Applied Soft Computing*, 12(9):2985–2996.

Liang, J., Zhigang, S., and Zhihui, L. (2010). Coevolutionary comprehensive learning particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, pp. 1–8.

Liang, J. J., Qin, A. K., Suganthan, P. N., and Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295.

Liu, Q. (2014). Order-2 stability analysis of particle swarm optimization. *Evolutionary Computation*, 23(2):187–216.

Mahor, A., Prasad, V., and Rangnekar, S. (2009). Economic dispatch using particle swarm optimization: A review. *Renewable and Sustainable Energy Reviews*, 13(8):2134–2141.

Malan, K., and Engelbrecht, A. (2009). Quantifying ruggedness of continuous landscapes using entropy. In *IEEE Congress on Evolutionary Computation*, pp. 1440–1447.

Mallipeddi, R., and Suganthan, P. (2010). Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore.

Matyas, J. (1965). Random optimization. *Automation and Remote Control*, 26(2):246–253.

Mendes, R. (2004). Population topologies and their influence in particle swarm performance. PhD thesis, Departmento de Informática, Universidade do Minho.

Mendes, R., Kennedy, J., and Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. In *Annual Conference on Evolutionary Programming*, pp. 135–155.

Michalewicz, Z., and Janikow, C. (1996). GENOCOP: A genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM*, 39(12):223–240.

Michalewicz, Z., and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

Moler, C., and Van Loan, C. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49.

Montes de Oca, M. A., and Stützle, T. (2008). Towards incremental social learning in optimization and multiagent systems. In *Genetic and Evolutionary Computation Conference*, pp. 1939–1944.

Montes de Oca, M. A., Stützle, T., Birattari, M., and Dorigo, M. (2009). Frankenstein's PSO: A composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 13(5):1120–1132.

Montes de Oca, M. A., Stützle, T., Van den Enden, K., and Dorigo, M. (2011). Incremental social learning in particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(2):368–384.

Montgomery, D. C. (1984). *Design and analysis of experiments*. New York: Wiley.

Muller, C., Baumgartner, B., and Sbalzarini, I. F. (2009). Particle swarm CMA evolution strategy for the optimization of multi-funnel landscapes. In *IEEE Congress on Evolutionary Computation*, pp. 2685–2692.

Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., and Suganthan, P. N. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences*, 209:16–36.

Newman, M., Barabasi, A.-L., and Watts, D. J. (2006). *The structure and dynamics of networks*. Princeton: Princeton University Press.

Nguyen, T., and Yao, X. (2012). Continuous dynamic constrained optimisation—The challenges. *IEEE Transactions on Evolutionary Computation*, 16(6):769–786.

Nickabadi, A., Ebadzadeh, M. M., and Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing*, 11(4):3658–3670.

Omidvar, M., Li, X., Mei, Y., and Yao, X. (2014). Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393.

Ozcan, E., and Mohan, C. K. (1999). Particle swarm optimization: Surfing the waves. In *IEEE Congress on Evolutionary Computation*, pp. 1939–1944.

Paquet, U., and Engelbrecht, A. (2003). A new particle swarm optimiser for linearly constrained optimisation. In *IEEE Congress on Evolutionary Computation*, pp. 227–233.

Paquet, U., and Engelbrecht, A. (2007). Particle swarms for linearly constrained optimisation. *Fundamenta Informaticae*, 76(1):147–170.

Parsopoulos, K., and Vrahatis, M. (2002a). Particle swarm optimization method for constrained optimization problems. *Intelligent Technologies—Theory and Application: New Trends in Intelligent Technologies*, 76:214–220.

Parsopoulos, K., and Vrahatis, M. (2002b). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2):235–306.

Parsopoulos, K. E., and Vrahatis, M. N. (2004). On the computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):211–224.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Application*, 2008(3):1–10.

Poli, R. (2009). Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation*, 13(4):712–721.

Poli, R., Brattonx, D., Blackwell, T., and Kennedy, J. (2007). Theoretical derivation, analysis and empirical evaluation of a simpler particle swarm optimiser. In *IEEE Congress on Evolutionary Computation*, pp. 1955–1962.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*, 1(1):33–57.

PSO (2006). PSO source code (available online at PSO info). Retrieved from http://particleswarm.info/standard-PSO-2006.c

Pulido, G. T., and Coello, C. A. C. (2004). A constraint-handling mechanism for particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pp. 1396–1403.

Ratnaweera, A., Halgamuge, S. K., and Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255.

Rudolph, G. (1997). Local convergence rates of simple evolutionary algorithms with cauchy mutations. *IEEE Transactions on Evolutionary Computation*, 1(4):249–258.

Rudolph, G. (1998). Finite Markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35(1–4):67–89.

Rudolph, G. (2013). Stochastic convergence. In T. B. Rozenberg and J. Kok (Eds.), *Handbook of natural computing*, pp. 847–869. Berlin: Springer.

Salomon, R. (1995). Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions—A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39:263–278.

Schmitt, M., and Wanka, R. (2013). Particle swarm optimization almost surely finds local optima. In *Genetic and Evolutionary Computation Conference*, pp. 1629–1636.

Schoenauer, M., and Michalewicz, Z. (1996). Evolutionary computation at the edge of feasibility. *Parallel Problem Solving from Nature*, pp. 245–254.

Schoenauer, M., and Michalewicz, Z. (1997). Boundary operators for constrained parameter optimization problems. In *Conference on Genetic Algorithms*, pp. 322–329.

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. New York: CRC Press.

Shi, Y., and Eberhart, R. (1998a). A modified particle swarm optimizer. In *IEEE World Congress on Computational Intelligence*, pp. 69–73.

Shi, Y., and Eberhart, R. (1998b). Parameter selection in particle swarm optimization. In V. Porto, N. Saravanan, D. Waagen, and A. Eiben (Eds.), *Evolutionary programming VII*, volume 1447 of LNCS, pp. 591–600. Berlin: Springer.

Shi, Y., and Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pp. 101–106.

Smith, L., Chinneck, J., and Aitken, V. (2013). Constraint consensus concentration for identifying disjoint feasible regions in nonlinear programmes. *Optimization Methods and Software*, 28(2):339–363.

Solis, F. J., and Wets, R. J.-B. (1981). Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30.

Song, M.-P., and Gu, G.-C. (2004). Research on particle swarm optimization: A review. In *International Conference on Machine Learning and Cybernetics*, pp. 2236–2241.

Spears, W., Green, D., and Spears, D. (2010). Biases in particle swarm optimization. *International Journal of Swarm Intelligence Research*, 1(2):34–57.

Storn, R., and Price, K. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Sun, C.-l., Zeng, J.-c., and Pan, J.-s. (2011). An improved vector particle swarm optimization for constrained optimization problems. *Information Sciences*, 181(6):1153–1163.

Takahama, T., and Sakai, S. (2005). Constrained optimization by ϵ constrained particle swarm optimizer with ϵ-level control. In A. Abraham, Y. Dote, T. Furuhashi, M. Köppen, A. Ohuchi, and Y. Ohsawa (Eds.), *Soft computing as transdisciplinary science and technology*, volume 29 of *Advances in soft computing*, pp. 1019–1029. Berlin: Springer.

Tang, Y., Wang, Z., and Fang, J.-a. (2011). Feedback learning particle swarm optimization. *Applied Soft Computing*, 11(8):4713–4725.

Trefethen, L. N., and Bau III, D. (1997). *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325.

Van den Bergh, F. (2002). An analysis of particle swarm optimizers. PhD thesis, Department of Computer Science, University of Pretoria.

Van den Bergh, F., and Engelbrecht, A. (2002). A new locally convergent particle swarm optimiser. In *Systems, Man and Cybernetics*, pp. 96–101.

Van den Bergh, F., and Engelbrecht, A. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239.

Van den Bergh, F., and Engelbrecht, A. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971.

Van den Bergh, F., and Engelbrecht, A. (2010). A convergence proof for the particle swarm optimiser. *Fundamenta Informaticae*, 105(4):341–374.

Voss, M. S. (2005). Principal component particle swarm optimization: A step towards topological swarm intelligence. In *IEEE Congress on Evolutionary Computation*, pp. 298–305.

Vrahatis, M., Androulakis, G., and Manoussakis, G. (1996). A new unconstrained optimization method for imprecise function and gradient values. *Journal of Mathematical Analysis and Applications*, 197(2):586–607.

Wang, H., Sun, H., Li, C., Rahnamayan, S., and Pan, J.-s. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences*, 223:119–135.

Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., and Tian, Q. (2011). Self-adaptive learning based particle swarm optimization. *Information Sciences*, 181(20):4515–4538.

Wilke, D. (2005). Analysis of the particle swarm optimization algorithm. Master's thesis, Department of Mechanical and Aeronautical Engineering, University of Pretoria, South Africa.

Wilke, D., Kok, S., and Groenwold, A. (2007). Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on scale and frame invariance. *International Journal for Numerical Methods in Engineering*, 70(8):985–1008.

Witt, C. (2009). Why standard particle swarm optimisers elude a theoretical runtime analysis. In *Foundations of Genetic Algorithms*, pp. 13–20.

Xinchao, Z. (2010). A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, 10(1):119–124.

Yang, J.-M., Chen, Y.-P., Horng, J.-T., and Kao, C.-Y. (1997). Applying family competition to evolution strategies for constrained optimization. In P. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart (Eds.), *Evolutionary programming VI*, volume 1213 of *LNCS*, pp. 201–211. Berlin: Springer.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Zhan, Z., Zhang, J., Li, Y., and Chung, H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(6):1362–1381.

Zhan, Z.-H., Zhang, J., Li, Y., and Shi, Y.-h. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 15(6):832–847.

Zhang, C., and Yi, Z. (2011). Scale-free fully informed particle swarm optimization algorithm. *Information Sciences*, 181(20):4550–4568.

Zheng, Y., Ma, L., Zhang, L., and Qian, J. (2003). Empirical study of particle swarm optimizer with an increasing inertia weight. In *IEEE Congress on Evolutionary Computation*, pp. 221–226.

Zou, R., Kalivarapu, V., Winer, E., Oliver, J., and Bhattacharya, S. (2015). Particle swarm optimization-based source seeking. *IEEE Transactions on Automation Science and Engineering*, 12(3):865–875.