Swarm Intelligence manuscript No.
(will be inserted by the editor)

Self-Adaptive Particle Swarm Optimization: A Review and Analysis of Convergence

Kyle Robert Harrison · Andries P. Engelbrecht · Beatrice M. Ombuki-Berman

Received: date / Accepted: date

Abstract Particle swarm optimization (PSO) is a population-based, stochastic search algorithm inspired by the flocking behaviour of birds. The PSO algorithm has been shown to be rather sensitive to its control parameters and thus performance may be greatly improved by employing appropriately tuned parameters. However, parameter tuning is typically a time-intensive empirical process. Furthermore, *a priori* parameter tuning makes the implicit assumption that the optimal parameters of the PSO algorithm are not time-dependent. To address these issues, self-adaptive particle swarm optimization (SAPSO) algorithms adapt their control parameters throughout execution. While there is a wide variety of such SAPSO algorithms in the literature, their behaviours are not well understood. Specifically, it is unknown whether these SAPSO algorithms will even exhibit convergent behaviour. This paper addresses this lack of understanding by investigating the convergence behaviours of 18 SAPSO algorithms both analytically and empirically. This paper also empirically examines whether the adapted parameters reach a stable point and whether the final parameter values adhere to a well-known convergence criterion. The results depict a grim state for SAPSO algorithms; over half of the SAPSO algorithms exhibit divergent behaviour while many others prematurely converge.

Keywords Particle swarm optimization, self-adaptive, parameter control, convergence

Kyle Robert Harrison
Department of Computer Science, University of Pretoria, Pretoria, South Africa
E-mail: kharrison@outlook.com

Andries P. Engelbrecht
Department of Computer Science, University of Pretoria, Pretoria, South Africa
E-mail: engel@cs.up.ac.za

Beatrice M. Ombuki-Berman
Department of Computer Science, Brock University, St. Catharines, Canada
E-mail: bombuki@brocku.ca

1 Introduction

The particle swarm optimization (PSO) algorithm (Kennedy and Eberhart, 1995) is a population-based, stochastic search algorithm inspired by the flocking behaviour of birds. The searching capability of the algorithm, and by extension the exploration/exploitation balance, is directly influenced by the three main control parameters, namely the inertia weight (ω), the cognitive acceleration coefficient (c_1), and the social acceleration coefficient (c_2) which are described in greater detail in Section 2. The PSO algorithm has been shown to be rather sensitive to these control parameters (Carlisle and Dozier, 2001; Trelea, 2003; van den Bergh and Engelbrecht, 2006) and thus *a priori* tuning of the control parameters may lead to improved performance. However, the tuning of parameters is often a time-intensive process whereby a researcher must empirically analyse a wide variety of parameter configurations to decide on the best choice. While automated parameter configuration tools have been proposed (for example, the F-Race algorithm by Birattari et al (2002)), such tools have two obvious drawbacks. Firstly, such tools simply automate the process of parameter selection and do not necessarily reduce the amount of time required to effectively tune the parameters. Secondly, there is an implicit assumption in *a priori* parameter tuning that the optimal parameter configuration does not change over time. The time-sensitivity of the control parameters is evidenced by Leonard and Engelbrecht (2013) where it was empirically found that parameters well-suited for exploration were not well-suited for exploitation, and *vice versa*. Moreover, the time-sensitivity of control parameter values is further evidenced by heterogeneous PSO algorithms which have shown that the most suitable velocity update scheme to employ varies during the search (Montes de Oca et al, 2009; Wang et al, 2011; Changhe Li et al, 2012; Nepomuceno and Engelbrecht, 2013).

To alleviate the issue of *a priori* parameter tuning, various self-adaptive particle swarm optimization (SAPSO) algorithms which adapt their control parameters throughout execution have been proposed. SAPSO algorithms typically make use of introspective observation to refine the control parameters based on their current and/or past performance. Table 1 lists 18 SAPSO algorithms, along with the control parameters they tune and the net change in the total number of control parameters relative to the canonical PSO (described in Section 2). Values in the 'Net Change' column of Table 1 should be interpreted as the overall change in the number of control parameters relative to the canonical PSO algorithm. Thus, positive values denote situations where an algorithm introduces more control parameters than it tunes while negative values denote situations where an algorithm tunes one or more control parameters without the introduction of further control parameters, leading to an overall reduction in the number of control parameters relative to the canonical PSO. In cases where there are multiple entries for the net change in parameters, the entry in parenthesis indicates the net change if the constants defined by that algorithm are treated as control parameters. For the case of grey particle swarm optimization (GPSO), n_d is the number of problem dimensions.

Examining Table 1, it is clear that a significant amount of effort has been devoted to solely adapting the inertia weight parameter. A further key observation is regarding the net change in the number of parameters. Of the 18 examined SAPSO algorithms, only five depict a net reduction in the number of parameters. Removal of the algorithm which simply generates random, convergent parameters each iteration (see Section 4.13) leaves four out of 17 algorithms which actually reduce the number of parameters relative to the standard PSO. If the constants defined by the various algorithms are treated as control parameters, then only one of the algorithms leads to a reduction of parameters while one further algorithm leads to no change in the number of parameters. Given that one of the primary objectives of an adaptive variant is to eliminate the need to specify values for the control parameters, proposing a variant which increases the number of parameters is likely counter-productive, unless insensitivity to the new parameters has been illustrated. Furthermore, newly introduced parameters do not benefit from the plethora of theoretical and empirical results readily available for the traditional PSO control parameters. Therefore, it may be more difficult to adequately tune the parameters of a SAPSO relative to tuning the control parameters of a traditional PSO. Nonetheless, it is entirely possible that an algorithm with a greater number of control parameters is more robust and easier to adequately tune. Newly introduced parameters should thus be accompanied by an appropriate sensitivity analysis to ascertain their robustness.

Table 1: Net change in the number of control parameters of various SAPSO algorithms relative to the canonical PSO. Entries in parenthesis indicate the net change if the constants defined by the algorithm are treated as control parameters. Note that n_d is the number of problem dimensions.

Optimizer	**Parameters Tuned**	**Net Change**
PSO-TVIW (Shi and Eberhart, 1998, 1999)	ω	+1
PSO-AIWF (Liu et al, 2005)	ω	+1
DAPSO (Yang et al, 2007)	ω	+2
IPSO-LT (Li and Tan, 2008)	ω	+1
SAPSO-LFZ (Li et al, 2008)	ω	-1 (0)
SAPSO-DWCY (Dong et al, 2008)	ω	-1 (+2)
PSO-RBI (Panigrahi et al, 2008)	ω	+1
IPSO-CLL (Chen et al, 2009)	ω	-1
AIWPSO (Nickabadi et al, 2011)	ω	+1
APSO-VI (Xu, 2013)	ω	+2
SRPSO (Tanweer et al, 2015)	ω	+2
PSO-SAIC (Wu and Zhou, 2007)	ω, c_2	+2 (+4)
PSO-RAC	ω, c_1, c_2	-3
PSO-TVAC (Ratnaweera et al, 2004)	ω, c_1, c_2	+3
PSO-ICSA (Jun and Jian, 2009)	ω, c_1, c_2	+3 (+31)
APSO-ZZLC (Zhan et al, 2009)	ω, c_1, c_2	-3 (+35)
UAPSO-A (Hashemi and Meybodi, 2011)	ω, c_1, c_2	+6
GPSO (Leu and Yeh, 2012)	ω, c_1, c_2	+3 (+(n_d + 3))

While there are a large number of SAPSO algorithms which have been proposed in the literature, their behaviour is still not well understood. Specifically, it is unknown whether these algorithms will even exhibit convergent behaviour. Convergence, in this context, refers to the attainment of an equilibrium state such that the variance of the particle step sizes is zero. An algorithm designed to adapt its control parameters can be reasonably expected to prevent divergent behaviour given that parameters which lead to divergent behaviours should be avoided by the adaptation mechanism. However, as previous works have identified (van Zyl and Engelbrecht, 2014; Harrison et al, 2016a,b), this is not always the case. Recently, evidence has been provided to suggest that parameter configurations which adhere to a well-known convergence criterion will generally lead to better performance than parameter configurations that violate the criterion (Cleghorn and Engelbrecht, 2016; Harrison et al, 2017). Furthermore, it has been shown that many of the parameter configurations which violate the convergence criterion lead to worse performance than random search (Cleghorn and Engelbrecht, 2016). As will be shown in this paper, many of the SAPSO algorithms, as they were proposed in the literature, have serious flaws which cause them to exhibit divergent behaviour. As such, this paper provides an in-depth review of 18 SAPSO algorithms. Their respective parameter adaptation mechanisms are analytically dissected to determine if and when they will lead to convergent behaviour. Finally, to support and complement the analytical findings, the convergent behaviour is empirically examined on a specially-formulated benchmark problem which isolates the convergence behaviour. Note that the purpose is not to empirically analyse and compare algorithms to determine which performs best, but only to analyse their convergence behaviour. Furthermore, this study does not attempt to prove or disprove the optimality of the control parameter values that are produced by the algorithms at any given time throughout the search. In fact, the question of whether the control parameter values adopted by SAPSO algorithms at any particular time are well-suited for the current environment is, to the best of the authors' knowledge, still unanswered. Finally, the purpose of this work is not to conduct a sensitivity analysis of the parameters introduced by each algorithm as such an endeavour would warrant an entire study by itself.

It should be noted that a recent study by the authors (Harrison et al, 2016a) provided a detailed analysis of inertia weight control strategies which shares some commonalities with this paper. As part of the study on inertia weight strategies, a theoretical analysis of convergence was performed using the same methodology as this paper. Specifically, the derivation technique used to determine when an algorithm will exhibit convergent behaviour is shared between the studies. This paper provides an extension of that study to include a variety of SAPSO algorithms, not strictly inertia weight adaptation strategies. There are, however, six algorithms which are common to both studies, four of which have the same parametrizations. While the empirical analysis of convergence from the inertia weight study was provided through a suite of 60 benchmark problems, this paper uses a benchmark problem which is for-

mulated explicitly for analysing convergence behaviour to provide generalized empirical support for the theoretical findings. In the study of inertia weight strategies, the average particle movement was used as an empirical measure of convergence. This paper provides, in addition to the average particle movement, three further measurements as part of the empirical analysis. Two of the additional measures examine the convergence in parameter space while the final additional measure examines the extent to which particles exit the feasible region. The previous study by Harrison et al (2016a) provided no analysis of convergence in parameter space, nor did it explicitly examine whether particles remained within the search space. This paper thus provides a much more thorough investigation of the convergence behaviour of the algorithms.

The remainder of this paper is structured as follows. Section 2 provides an overview of the canonical PSO algorithm. Section 3 describes the theoretical conditions necessary for the PSO algorithm to exhibit convergent behaviour and how such behaviour can be empirically captured. Section 4 serves as an in-depth review of SAPSO algorithms and presents the results of both the analytical and empirical investigation of their convergence. Finally, concluding remarks and avenues of future research are presented in Section 5.

2 Particle swarm optimization

The PSO algorithm is inspired by the social behaviours of a flock of birds. The algorithm consists of a swarm of agents, referred to as particles, where each particle represents a candidate solution to the optimization problem at hand. The basic steps of the algorithm are a repeated calculation and subsequent application of a discrete velocity (or step size) associated with each particle, thereby providing movement. A particle's velocity is based largely on an explicit attraction towards two promising locations in the search space, namely the best position found by the particle and the best position found by any particle within the particle's neighbourhood. The neighbourhood of a particle refers to the set of other particles within the swarm from which it may take influence. The original PSO algorithm employed one of two neighbourhood topologies, either a star topology where the neighbourhood is the entire swarm, or a ring topology where the neighbourhood consists of the immediate neighbours when the particles are arranged in a ring. PSO variants employing the star and ring neighbourhood topologies are commonly referred to as global-best PSO and local-best PSO, respectively. Differences in performance between the global-best and local-best PSO variants are largely problem-dependent and neither topology can, in general, be regarded as strictly superior to the other for many classes of optimization problems (Engelbrecht, 2013a). Furthermore, if the objective of a study is to determine if a specific algorithmic change leads to improved performance, then the choice of a global-best or local-best topology is arbitrary (Engelbrecht, 2013a).

For the purposes of this study, a global-best topology is employed. The velocity is calculated for particle i according to the inertia weight model of

Shi and Eberhart (1998) as

$$\begin{aligned} v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) \\ + c_2 r_{2ij}(t)(\hat{y}_j(t) - x_{ij}(t)) \end{aligned} \tag{1}$$

where $v_{ij}(t)$ and $x_{ij}(t)$ are the velocity and position in dimension j at time t, respectively. The inertia weight is given by ω while c_1 and c_2 represent the cognitive and social coefficients, respectively. The stochastic component of the algorithm is provided by the random numbers, $r_{1ij}(t), r_{2ij}(t) \sim U(0,1)$, which are independently sampled each iteration for all components of the velocity of each particle. Finally, $y_{ij}(t)$ and $\hat{y}_j(t)$ denote the personal and neighbourhood best positions in dimension j, respectively. The inertia weight PSO model thus contains three primary control parameters, namely ω, c_1, and c_2. Particle positions are then updated according to

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{2}$$

3 Convergence analysis of particle swarm optimization

There has been a significant amount of research effort devoted to the theoretical study of PSO convergence (Trelea, 2003; van den Bergh and Engelbrecht, 2006; Kadirkamanathan et al, 2006; Poli and Broomhead, 2007; Poli, 2009; Gazi, 2012; Cleghorn and Engelbrecht, 2014a, 2017). Specifically, researchers are interested in analytically determining the values of PSO parameters which lead to convergent behaviour. While convergent behaviour does not necessarily translate to superior performance, it keeps particle step sizes within reason and will generally cause the step sizes to tend towards zero. Conversely, divergent behaviour can cause particle step sizes to be orders of magnitude larger than the size of the initial search space (Cleghorn and Engelbrecht, 2014b), thereby causing a large portion of the search effort to be wasted on infeasible solutions and particles leaving the search space Engelbrecht (2013b).

According to the theoretical analyses of Poli and Broomhead (2007) and Poli (2009), control parameters which adhere to

$$c_1 + c_2 < \frac{24(1-\omega^2)}{7-5\omega} \tag{3}$$

will lead to convergent behaviour in the PSO algorithm. Alternatively, Eq. (3) can be rewritten as

$$\frac{5C - g(C)}{48} < \omega < \frac{5C + g(C)}{48} \tag{4a}$$

with

$$C = c_1 + c_2 \text{ and } g(C) = \sqrt{25C^2 - 672C + 2304} \tag{4b}$$

to provide a condition on ω. The region defined by Eq. (3), depicted in Figure 1, has been empirically demonstrated to be the most accurate of the various

proposed convergence criteria (Cleghorn and Engelbrecht, 2014b, 2015) and is thus the convergence criterion used in this study. Further studies by Liu (2015) and Cleghorn and Engelbrecht (2015) have also suggested that the region defined by Eq. (3) is not dependent upon the neighbourhood topology employed by PSO. Moreover, recent studies have provided evidence that parameter configurations which adhere to the criterion of Eq. (3) will generally lead to better performance than parameter configurations which violate the criterion (Cleghorn and Engelbrecht, 2016; Harrison et al, 2017). Specifically, it was shown that a majority of theoretically unstable parameter configurations cause the PSO algorithm to perform worse than random search and that selecting theoretically convergent parameters drastically increases the likelihood of PSO outperforming random search (Cleghorn and Engelbrecht, 2016). Therefore, a SAPSO algorithm should strive to ensure that the control parameter values it adapts adhere to the theoretical convergence criterion.

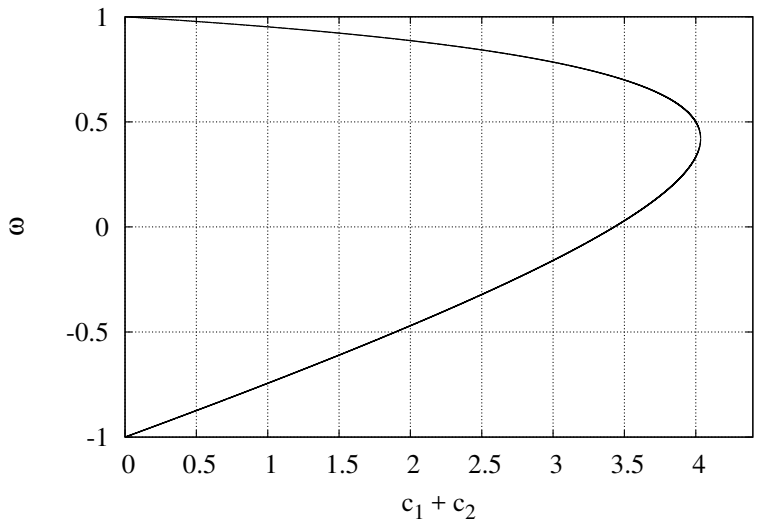


Fig. 1: Visualization of Poli's convergent region for PSO parameters. Parameters which lie within the parabolic region lead to convergent behaviour.

3.1 Empirical convergence analysis of self-adaptive particle swarm optimizers

The empirical results in this study are obtained via a specially-formulated benchmark function. The function used in this study is a vertically-shifted version of the function proposed by Cleghorn and Engelbrecht (2014b, 2015) and provides the algorithms with an environment in which complete stagnation

is highly unlikely, thereby isolating the convergence behaviour. The function is given by

$$F(\mathbf{x}) \sim U(0, 2000) \tag{5a}$$

such that

$$F(\mathbf{x}_1) = F(\mathbf{x}_2) \text{ if } \mathbf{x}_1 = \mathbf{x}_2. \tag{5b}$$

Equation (5a) thus defines the fitness value of each position in the search space as a randomly-sampled real value within the range [0, 2000]. Equation (5b) stipulates that subsequent evaluations of the same position during a simulation will always result in the same fitness. Therefore, the fitness value of each unique position is randomly determined, but remains fixed throughout each independent simulation. Note that the function was shifted to produce strictly non-negative fitness values as some of the examined algorithms require such a condition. Also note that it is not strictly necessary for all particles to converge toward the same point in the search space for Eq. (5) to correctly identify scenarios in which the particle's behaviour is convergent (Cleghorn and Engelbrecht, 2014b). Rather, convergent behaviour, in this context, denotes a situation in which the particle positions have become stable, but not necessarily to a single point.

To empirically measure convergence, the average Euclidean distance of the particles' movement (Cleghorn and Engelbrecht, 2014b, 2015) given by

$$\Delta(t+1) = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i(t+1) - \mathbf{x}_i(t)|| \tag{6}$$

is employed. This measure will reflect a scenario in which even a single particle is divergent. A sensible upper threshold, Δ_{max}, is defined as the maximal distance between any two points in the feasible search space, according to

$$\Delta_{max} = \sqrt{k(l-u)^2} \tag{7}$$

where $[l, u]^k$ is the (feasible) domain of the objective function (Cleghorn and Engelbrecht, 2015). The use of Δ_{max} as the criterion to determine convergence has been empirically demonstrated to be 98.79% accurate when correlated with the convergence criterion of Poli (Cleghorn and Engelbrecht, 2015). Due to the well-known phenomenon of velocity explosion (Engelbrecht, 2013b), it is not uncommon for convergent particles to have step sizes which initially exceed Δ_{max} and decrease over time. However, if particle movements are consistently exceeding this value, convergent behaviour is clearly not being exhibited. In this study, the domain of the problem is fixed at $[-100, 100]^{50}$ and, therefore, $\Delta_{max} = 1414.214$. Additionally, the movement values are capped at 2000 to prevent excessively divergent behaviour from muddling the results.

An analogous version of Eq. (6) is used to measure the average Euclidean distance of the parameter configurations between successive iterations. This measure indicates the stability of control parameter values over time. Furthermore, the percentage of particles with convergent values for their control

parameters (i.e., values which adhere to Eq. (3)) and the percentage of particles which are outside the feasible region (i.e., have a bound violation in at least one dimension) are measured at each iteration.

4 Critical analysis of self-adaptive particle swarm optimizers

SAPSO algorithms fall broadly into two categories, namely time-variant approaches and (true) self-adaptive approaches. Time-variant approaches alter the control parameters based solely on the number of iterations which have passed. Given that such algorithms make no introspective adaptations, they cannot be considered self-adaptive in the proper sense. Nonetheless, time-variant approaches encourage varied exploration and exploitation during the search. In contrast to time-variant approaches, truly self-adaptive PSO variants use introspective observation to adapt their control parameters. Refinement of control parameters is done at either the global (swarm) level or the local (particle) level based on the state of the algorithm.

This section presents a number of self-adaptive PSO algorithms from both categories. It should be noted that this section focuses solely on the adaptation mechanisms and does not account for any additional algorithmic components (e.g., mutations) that may be present in the algorithm. A theoretical analysis of the convergent behaviour was performed, ultimately describing the algorithmic conditions necessary for the respective algorithm to exhibit convergent behaviour. Furthermore, Eq. (5) was employed to empirically investigate the convergence of both the particles as well as the control parameters. This empirical investigation provides an individualized profile of the convergence behaviour of each algorithm and primarily serves to support and complement the analytical results, rather than provide a direct comparison between algorithms.

Four performance measures were employed to evaluate the behaviour of each algorithm as follows:

1. The **average particle movement** was calculated using Equation (6) and quantifies the average particle step size. If particle steps sizes do not decrease, particles will not converge. Furthermore, large step sizes may prevent particles from exploiting promising locations in the search space.
2. The **percentage of particles with convergent control parameters** measures the proportion of particles that had parameter settings which adhered to Poli's convergence criterion. This measure provides an indication of an algorithm's ability to generate convergent parameters.
3. The **average parameter movement** measures the average step size in parameter space. This measure provides an indication of the stability of the employed control parameter values. The average parameter movement was calculated using Equation (6), using the control parameter values rather than the particle position.
4. The **percentage of particles with a bound violation** measures the proportion of particles which violated the boundary constraints in at least

one dimension. This measure provides an indication of the search effort which is wasted on infeasible solutions.

Empirical results depict the measured values of each of the four performance measures averaged over 50 independent runs, each consisting of 5000 iterations. Each algorithm made use of a global best (star) topology, as used in the original PSO algorithm (Kennedy and Eberhart, 1995), and a synchronous iteration strategy (Carlisle and Dozier, 2001). Particle positions were randomly initialized within the search space and their initial velocity was set to $\mathbf{0}$ (Engelbrecht, 2012). To prevent invalid attractors, a particle's personal best position was only updated if the new position had a better objective function value and was within the (feasible) search space. Algorithmic parameters, as described below and summarized in Table 3, were employed based on the guidelines of the respective authors.

It should be explicitly stated that there was no attempt to tune any of the respective control parameter values and that the observations made herein may change if the values of the control parameters are altered. While it can be argued that different values for the various control parameters may lead to an unfair comparison, the parameters employed in this study are those offered by the respective authors and, therefore, this study examines the convergence behaviour of the algorithms as they were published. Furthermore, given that a self-adaptive algorithm is explicitly designed to remove the need for parameter tuning, it is reasonable to assume that the parameters of such algorithms should not need to be tuned.

4.1 Particle swarm optimization with time-varying inertia weight

The particle swarm optimization with time varying inertia weight (PSO-TVIW) (Shi and Eberhart, 1998, 1999) was proposed as a method to linearly decrease the inertia weight over time. This inertia weight strategy was based on the general consensus that exploration is favoured early in the search process while exploitation is favoured later. The inertia weight is calculated at each iteration according to

$$\omega(t) = \omega_s + (\omega_f - \omega_s)\frac{t}{T} \tag{8}$$

where ω_s and ω_f are the initial and final inertia weights, respectively, and T is the maximum number of iterations. The social and cognitive coefficients remain static over the course of execution. Thus, the PSO-TVIW algorithm tunes the ω parameter at the expense of adding two new control parameters, namely ω_s and ω_f.

As employed by Shi and Eberhart (1999), the parameters for the PSO-TVIW algorithm are set as $\omega_s = 0.9$ and $\omega_f = 0.4$, with $c_1 = c_2 = 1.49618$. It was shown by Harrison et al (2016a)[1] that using this parametrization, the

[1] In the study of Harrison et al (2016a), PSO-TVIW was referred to as PSO-LDIW.

PSO-TVIW algorithm will only exhibit convergent behaviour after 22.9% of the search has completed.

As Figure 8a depicts, the average particle movement for PSO-TVIW drops below Δ_{max} after roughly 1000 iterations (i.e., 20%). However, as Figure 9a shows, the resulting control parameters were not convergent before the expected 22.9% of the search was completed, indicating that the particle step sizes dropped below Δ_{max} slightly before the parameters adhered to the convergence criterion. Given the aforementioned parametrization, the inertia weight value changes by 0.0001 each iteration leading to the constant, near-zero parameter movement values shown in Figure 10a. The PSO-TVIW algorithm depicts a gradual decrease in particle bound violations over time, as shown in Figure 11a. This is likely a result of the decreasing inertia weight gradually altering the behaviour of the algorithm from exploration to exploitation as the search progresses.

4.2 Particle swarm optimization with adaptive inertia weight factor

The particle swarm optimization with adaptive inertia weight factor (PSO-AIWF) algorithm (Liu et al, 2005) adapts the inertia weight based on a particle's fitness relative to the average fitness. The authors posit that particles with good fitness values should be 'protected' through the use of low inertia values while particles with inferior fitnesses should be 'disrupted' via larger inertia weights. Using this premise, the inertia weight of each particle is given by

$$\omega_i(t) = \begin{cases} \omega_{min} + \frac{(\omega_{max}-\omega_{min})(f_i(t)-f_{min}(t))}{\overline{f(t)}-f_{min}(t)} & \text{if } f_i(t) \leq \overline{f(t)} \\ \omega_{max} & \text{if } f_i(t) > \overline{f(t)} \end{cases} \tag{9}$$

where $\overline{f(t))}$ and $f_{min}(t)$ are the average and minimum fitness values at time t and ω_{min} and ω_{max} are the user-supplied minimum and maximum inertia weights. The PSO-AIWF algorithm thus tunes the value of the ω parameter but introduces two additional parameters, namely ω_{min} and ω_{max}.

Liu et al (2005) used parameters $\omega_{min} = 0.2, \omega_{max} = 1.2$, and $c_1 = c_2 = 2$, which leads to a range of $\frac{1}{3} < \omega_i(t) < \frac{1}{2}$ for convergent behaviour to be exhibited. Examining Eq. (9), it is noted that any particle which has a fitness value worse than the average will have an inertia value of 1.2 and therefore will not demonstrate convergent behaviour. Furthermore, the only scenario in which convergent behaviour will occur is when

$$f_i(t) \leq \overline{f(t)}$$

and

$$\frac{2}{15} < \frac{f_i(t) - f_{min}(t)}{\overline{f(t)} - f_{min}(t)} < 0.3.$$

Given that roughly half of the particles will exhibit divergent behaviour on any given iteration, the PSO-AIWF algorithm is expected to demonstrate rapid divergence.

The rapid divergence of the PSO-AIWF algorithm is exemplified by the average particle movement immediately hitting the maximal value of 2000 and never decreasing, as shown in Figure 8b. Moreover, Figure 9b depicts that no particles employ convergent parameter configurations indicating that Eq. (9) is unsuccessful at generating inertia values within the convergent range. In fact, Figure 10b depicts that the average change in parameter space is immediately near zero, suggesting that the adaptation mechanism was failing to adapt the inertia weight altogether. As a result, particles immediately exit the feasible region and never return; Figure 11b indicates that 100% of the particles are infeasible throughout the entirety of the search.

4.3 Dynamic adaptation particle swarm optimization

The dynamic adaptation particle swarm optimization (DAPSO) algorithm (Yang et al, 2007) is a PSO variant whereby two calculated values are used to describe the state of the algorithm. Adapted from those originally introduced by Xuanping et al (2005), the evolutionary[2] speed factor and aggregation degree are used by the DAPSO algorithm to dynamically adapt the individualized inertia weights.

The evolutionary speed factor of particle i at time t considers the history of the particle according to

$$h_i(t) = \left| \frac{\min\{f(\mathbf{y}_i(t-1)), f(\mathbf{y}_i(t))\}}{\max\{f(\mathbf{y}_i(t-1)), f(\mathbf{y}_i(t))\}} \right|. \tag{10}$$

Note that $0 \leq h \leq 1$, and smaller values for h correspond to faster 'evolution'; a high value for h implies that a major improvement to the personal best position has been achieved.

The aggregation degree measures the similarity between the average fitness and the best fitness from iteration t according to

$$s(t) = \left| \frac{\min\{f(\mathbf{y}^*(t)), f_{avg}(t)\}}{\max\{f(\mathbf{y}^*(t)), f_{avg}(t)\}} \right| \tag{11}$$

where $\mathbf{y}_i^*(t)$ denotes the best solution found during iteration t and $f_{avg}(t)$ is the average fitness of the entire swarm.

The inertia weight of particle i at time t is the calculated as

$$\omega_i(t) = \omega_s - \alpha(1 - h_i(t)) + \beta s(t) \tag{12}$$

where α and β are user-supplied values in the range $[0, 1]$. Due to both $h_i(t)$ and $s(t)$ being within the range $[0, 1]$, it can be shown that

$$\forall t : 1 - \alpha \leq \omega_i(t) \leq 1 + \beta. \tag{13}$$

[2] PSO does not actually exhibit evolution. Nonetheless, the original terminology is used in this work.

The DAPSO algorithm thus tunes the value of the ω parameter at the expense of introducing three additional parameters, namely ω_s, α, and β. Yang et al (2007) concluded that the parametrization of DAPSO, specifically the α and β parameters, does not have a significant impact on performance. Therefore, the parameters used in this work, namely $\alpha = 1.0, \beta = 0.1, \omega_s = 1.0$, and $c_1 = c_2 = 1.496180$, are taken from previous usage (Yang et al, 2007; van Zyl and Engelbrecht, 2014).

From Eq. (13), it follows that

$$\forall t : 0.0 \leq \omega_i(t) \leq 1.1,$$

and, given the aforementioned social and cognitive control parameters, the DAPSO algorithm thus requires $\omega_i(t) < 0.78540$ to be convergent. Eq. (12) can then be simplified to

$$\omega_i(t) = h_i(t) + 0.1s(t), \tag{14}$$

demonstrating that the inertia, and thereby the convergence, is predominantly influenced by $h_i(t)$. Furthermore, this indicates that

$$h_i(t) + 0.1s(t) < 0.78450$$

is necessary for convergent behaviour. Given that the maximum value for $s_i(t)$ is 1.0, then

$$h_i(t) < 0.68450$$

becomes the predominant requirement for convergence. Therefore, only when the ratio of a particle's personal best fitness to its previous personal best fitness is below 0.68450 (i.e., improves by more than 31.6%) will the DAPSO algorithm exhibit convergent behaviour. However, sustaining such a high degree of fitness improvement is infeasible, thus the DAPSO algorithm is expected to exhibit divergent behaviour.

Given the infeasibility of continual fitness improvements, the DAPSO algorithm depicts immediately divergent behaviour, as shown in Figure 8c. Furthermore, as Figures 9c and 10c demonstrate, the parameters generated via the adaptation mechanism are never within the convergent region and are wildly unstable. In fact, the adaptation mechanism fails to generate reasonable parameters as a result of the explosive divergence, causing fitness values to deteriorate such that they become too large to represent as a standard 32-bit floating point value. As a result, Figure 11c shows that particles immediately exit the feasible region and never return. Thus, the adaptation mechanism of the DAPSO algorithm is wildly deficient.

4.4 Improved particle swarm optimization I

The improved particle swarm optimization I (IPSO-LT) algorithm (Li and Tan, 2008) is premised on the assumption that the inertia weight should be in direct relation to the convergence factor,

$$c_i(t) = \frac{|f(\mathbf{y}_i(t-1)) - f(\mathbf{y}_i(t))|}{f(\mathbf{y}_i(t-1)) + f(\mathbf{y}_i(t))}, \tag{15}$$

as well as the diffusion factor,

$$d_i(t) = \frac{|f(\mathbf{y}_i(t)) - f(\hat{\mathbf{y}}_i(t))|}{f(\mathbf{y}_i(t)) + f(\hat{\mathbf{y}}_i(t))}, \tag{16}$$

which characterize the state of the algorithm. The inertia weight of each particle is controlled by

$$\omega_i(t) = 1 - \left| \frac{\alpha(1 - c_i(t))}{(1 + d_i(t))(1 + \beta)} \right| \tag{17}$$

where $\alpha, \beta \in [0, 1]$ are user-supplied constants. The IPSO-LT algorithm thus tunes the value of the ω parameter but introduces two additional parameters, namely α and β.

Li and Tan (2008) employed parameters of $c_1 = c_2 = 2.0$ but did not specify the values used for α and β. Therefore, the mid-point of the allowable range was used for both these parameters, namely $\alpha = \beta = 0.5$. Using acceleration coefficients of $c_1 = c_2 = 1.496180$, Harrison et al (2016a) demonstrated that

$$0.64380 < \left| \frac{1 - c_i(t)}{d_i(t) + 1} \right|$$

was necessary for convergence. Following the same process with $c_1 = c_2 = 2.0$,

$$\frac{2}{3} < \left| \frac{1 - c_i(t)}{d_i(t) + 1} \right| < 2 \tag{18}$$

is necessary for convergence to be exhibited during any given iteration.

As evidenced by the average particle movement in Figure 8d, the IPSO-LT algorithm demonstrates immediate divergent behaviour. Thus, it can be concluded that the condition in Eq. (18) is never satisfied, thereby leading to all particles having divergent behaviours. This is further evidenced by Figure 9d depicting that no particles employ convergent parameter configurations. Moreover, Figure 10d shows that parameters are not changing over time, indicating that the adaptation mechanisms is inherently flawed and is incapable of tuning the inertia weight control parameter. However, Figure 11d shows that not all particles are outside of the search space. Specifically, the particle corresponding to the best position remains within the feasible region given that it will have a zero velocity as a result.

4.5 Self-adaptive particle swarm optimization I

The self-adaptive particle swarm optimization I (SAPSO-LFZ) algorithm (Li et al, 2008) adapts the inertia weight of each particle based on its personal best fitness in relation to the average personal best fitness. At each iteration, the inertia weight is calculated as

$$\omega_i(t) = 0.15 + \frac{1}{1 + e^{F_i(t)}} \tag{19a}$$

with

$$F_i(t) = \overline{f(\mathbf{y}(t))} - f(\mathbf{y}_i(t)) \tag{19b}$$

where $\overline{f(\mathbf{y}(t))}$ is the average personal best fitness. Figure 2 visualizes the result of Eq. (19a) based on the value of $F_i(t)$. The SAPSO-LFZ algorithm thus tunes the value of the ω parameter and does not introduce any additional parameters. However, the value of 0.15 in Eq. (19a) may require tuning and could be considered a control parameter.

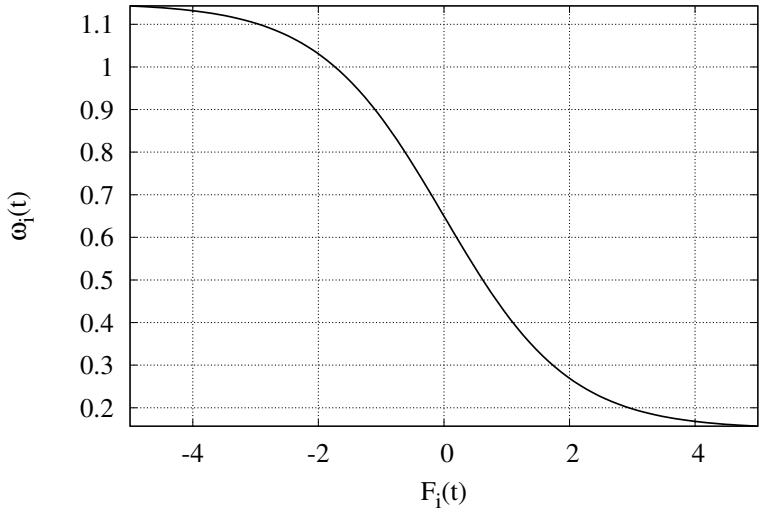


Fig. 2: The inertia value of the SAPSO-LFZ algorithm based on the value of $F_i(t)$.

Li et al (2008) used parameters $c_1 = c_2 = 1.496180$, which simplifies the convergence criterion to

$$\omega_i(t) < 0.78540.$$

From (19a),

$$\forall t : 0.15 < \omega_i(t) < 1.15,$$

and furthermore, $\omega_i(t) < 0.78540$ occurs when

$$f(\mathbf{y}_i(t)) < \overline{f(\mathbf{y}_i(t))} + 0.55545. \tag{20}$$

Thus, only particles which have a personal best fitness which is no greater than 0.55545 above the average will exhibit convergent behaviour. Given that particles tend to roam and exit the feasible search space early in the search (Engelbrecht, 2012, 2013b), it is expected that not all particles will improve their personal best positions initially. When a particle does not improve its personal best position, the corresponding fitness does not improve and the particle is assigned a divergent trajectory. This effectively prevents the particle from improving its personal best fitness in subsequent iterations. Therefore, divergent behaviour is expected for the SAPSO-LFZ algorithm.

Figure 8e depicts movement values which immediately hit the maximum value of 2000 and never decrease, thereby validating the theoretical expectations. Given that particles which attain a fitness value no greater than the average swarm fitness plus 0.55545 will lead to convergent behaviour, it is expected that roughly 50% of the particles will exhibit convergent behaviour. As evidenced by Figure 9e, just under 50% of the particles show convergent behaviour at any given iteration. To further support this observation, Figure 11e shows that just over 50% of particles (i.e., those with non-convergent behaviours) are outside the search space after only a few iterations. However, as Figure 10e shows, there is little to no adaptation of parameters occurring which is indicative of stagnating personal best fitness values. Thus, as expected, initially divergent behaviours cause no improvements in fitness, thereby causing the adaptation mechanism to have virtually no effect.

4.6 Self-adaptive particle swarm optimization II

The self-adaptive particle swarm optimization II (SAPSO-DWCY) algorithm (Dong et al, 2008) is based on an assumed relationship among various characteristics of the search. The authors propose that there exists a relationship between the inertia weight, fitness, swarm size, and problem dimension. The authors also posit that problems in higher dimensions can benefit from an increased inertia weight as this will help to escape local optima. Similarly, the authors claim that higher inertia weight values, and thereby enhanced exploration, can be used to compensate for smaller swarm sizes. To this end, the inertia weight for a particle i at time t is calculated as

$$\omega_i(t) = \frac{1}{\alpha - e^{-n_s/\beta} + \left(\frac{R_i(t)}{\gamma * n_d}\right)^2} \tag{21}$$

where $R_i(t)$ denotes the fitness rank of the particle, and α, β, and γ are empirically determined constants with values 3, 200, and 8, respectively. The SAPSO-DWCY algorithm thus tunes the value of the ω parameter without introducing any additional parameters. However, the values of α, β, and γ in

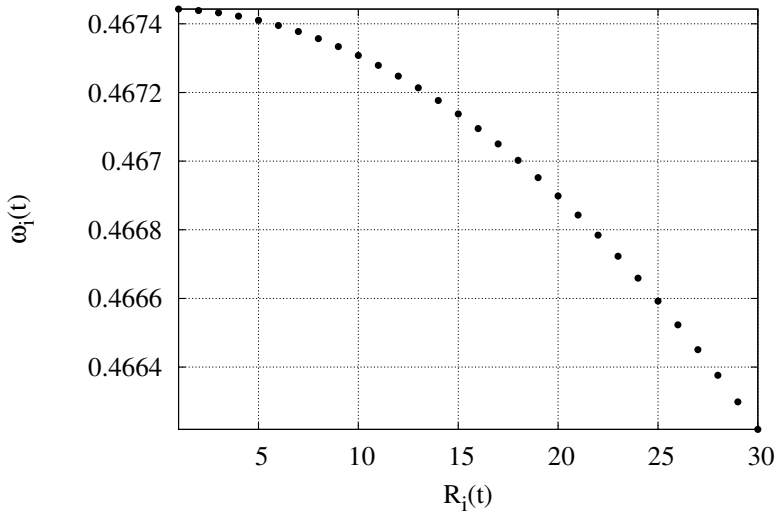


Fig. 3: The inertia value of the SAPSO-DWCY algorithm based on the fitness rank of the particle, assuming 30 particles in 50 dimensions.

Eq. (21) may require tuning and could be considered as control parameters. It should be noted that particles with better fitnesses are assigned higher ranks – the best-fit particle is assigned a rank of n_s while the worst-fit particle is assigned a rank of 1. This update strategy increases the inertia weight for low ranking particles (i.e., particles with relatively bad fitnesses) to enhance their exploration while high ranked (i.e., more fit) particles have their inertia decreased to encourage exploitation. Figure 3 demonstrates the inertia weight values assuming a swarm size of 30 particles and a 50-dimensional problem.

Dong et al (2008) employed control parameters of $c_1 = c_2 = 2.0$, leading to the necessary condition of

$$\frac{1}{3} < \omega_i(t) < \frac{1}{2}$$

for convergent behaviour to be depicted. By substituting minimum and maximum ranks of 1 and 30 into Eq. (21), the range of possible inertia values exemplified by the SAPSO-DWCY algorithm is then given by

$$0.46622 < \omega_i(t) < 0.46744.$$

Thus, all possible values for the inertia weight fall within the convergent range, and therefore the SAPSO-DWCY algorithm will exhibit convergent behaviour. However, the strikingly small range for the inertia weight values (i.e., 0.00122) will cause all configurations of the control parameters to lie relatively close to

the boundaries of the convergent region. As a result, the SAPSO-DWCY algorithm is expected to exhibit unreasonably slow declines in particle movement (Cleghorn and Engelbrecht, 2014b).

While Figure 8f does depict the average particle movement value is below the threshold for most iterations, there is no decline in particle movement over time; the average particle movement value hovers around 1400. Moreover, as Figures 9f and 10f depict, the parameters are always within the convergent region and are undergoing virtually no change over time. While the SAPSO-DWCY algorithm could thus be considered convergent, it cannot be said that the particles are stagnating nor that the swarm is converging to a stable point. Finally, despite the categorically convergent behaviour, Figure 11f demonstrates that no solutions are retained within the feasible region; the particles immediately exit the feasible region and never return.

4.7 Particle swarm optimization with rank-based inertia

The particle swarm optimization with rank-based inertia (PSO-RBI) algorithm (Panigrahi et al, 2008) is based on a claim that the movement of the swarm should be controlled by the objective function. Thus, the PSO-RBI algorithm adapts the inertia weight of each particle based on the rank of its fitness relative to the remainder of the swarm. The inertia weight of each particle is given by

$$\omega_i(t) = \omega_{min} + \frac{R_i(t)(\omega_{max} - \omega_{min})}{n_s} \tag{22}$$

where $R_i(t)$ is the fitness rank of particle i at time t and ω_{min} and ω_{max} are the user-supplied minimum and maximum values of the inertia weight. The PSO-RBI algorithm thus tunes the value of the ω parameter but introduces two additional parameters, namely ω_{min} and ω_{max}. According to Eq. (22), the best-fit particle (i.e., rank 1) will be assigned the lowest inertia weight while the worst fit particle will be assigned the highest inertia weight as an attempt to enhance exploitation for the best particles and exploration for the worst.

Parameters of $\omega_{min} = 0.4$ and $\omega_{max} = 0.9$ were employed by Panigrahi et al (2008). However, there was no mention of the social and cognitive acceleration coefficients employed, and thus sensible defaults of $c_1 = c_2 = 1.496180$ (van den Bergh and Engelbrecht, 2006) are used for the purposes of this study. Assuming a swarm size of 30 particles, Harrison et al (2016a) showed that the worst 22.9% of the swarm will always exhibit non-convergent behaviour, leading to overall divergent behaviour being exhibited by the PSO-RBI algorithm.

The divergent behaviour of the PSO-RBI algorithm is evidenced by the particle movement over time in Figure 8g, which depicts a rapid increase in particle movement until the value reaches the maximal value of 2000, from which it never decreases. The empirical results in Figure 9g suggest that 20% of the particles employ divergent parameters, which is directly in line with the theoretical prediction. Figure 10g demonstrates a non-zero particle movement is maintained throughout the search, indicating the adaptive mechanism does

not stagnate and thereby suggesting that the rank of particles is constantly changing. Despite the overall divergent behaviour depicted by the PSO-RBI algorithm, Figure 11g demonstrates that the number of infeasible particles is constantly decreasing, finalizing at roughly 40% infeasible solutions after 5000 iterations. Thus, despite overall divergent behaviour, a majority of the particles in the PSO-RBI algorithm remain within the bounds of the feasible region.

4.8 Improved particle swarm optimization II

The improved particle swarm optimization II (IPSO-CLL) algorithm (Chen et al, 2009) uses an adaptive inertia weight aimed at accelerating the convergence speed of the PSO algorithm. To this end, the inertia weight at time t is given by

$$\omega(t) = e^{-\lambda(t)} \tag{23a}$$

with

$$\lambda(t) = \frac{\alpha(t)}{\alpha(t-1)} \tag{23b}$$

and

$$\alpha(t) = \frac{1}{n_s} \sum_{i=1}^{n_s} |f(\mathbf{x}_i(t)) - f(\mathbf{y}^*(t))| \tag{23c}$$

where $\mathbf{y}^*(t)$ is the best particle found during iteration t. In Eq. (23a), the use of the exponential function has not been empirically determined, but rather was introduced based on its presence in engineering calculations (Chen et al, 2009). In this approach, $\alpha(t)$ is used to identify the smoothness of the fitness values and allows the inertia weight to vary according to the convergence of particles. Chen et al (2009) further claim that the convergence state of the algorithm is dependent upon the value of $\lambda(t)$[3]; when $\lambda(t) < 1$, the algorithm demonstrates convergent behaviours while $\lambda(t) > 1$ leads to globally divergent behaviour. Similarly, the value of $\lambda(t)$ directly affects the exploration of particles as a smaller value for $\lambda(t)$ implies a larger $\omega(t)$, as seen in Figure 4, and thus increased exploration. Note that the IPSO-CLL algorithm tunes the ω control parameter while introducing no additional parameters.

Chen et al (2009) employed control parameters of $c_1 = c_2 = 2.0$ in their initial proposal of the IPSO-CLL algorithm. Substitution of these values into Eq. (4) reduces the convergence criterion to

$$\frac{1}{3} < \omega(t) < \frac{1}{2}$$

for the IPSO-CLL algorithm. It then follows from Eq. (23a) that

$$\log(2) < \lambda(t) < \log(3)$$

[3] No indication of the convergence when $\lambda(t) = 1$ was provided by Chen et al (2009).

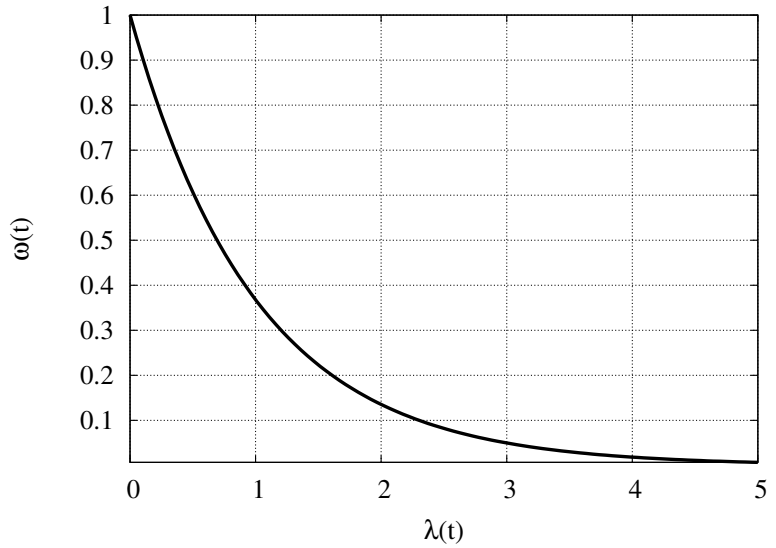


Fig. 4: The inertia value of the IPSO-CLL algorithm relative to the smoothness function $\lambda(t)$.

is necessary for convergent behaviour to occur. Moreover, convergent behaviour at time t will only be exhibited when

$$\log(2) < \frac{\sum_{i=1}^{n_s} |f(\mathbf{x}_i(t)) - f(\mathbf{y}^*(t))|}{\sum_{i=1}^{n_s} |f(\mathbf{x}_i(t-1)) - f(\mathbf{y}^*(t-1))|} < \log(3).$$

In other words, convergent behaviour will only be exhibited when the average difference in fitness between all particles and the iteration best deviates between 30.1% and 47.8%. Given that it would be unreasonable to attain a high level of deviation between fitnesses during successive iterations, it is therefore expected that the IPSO-CLL algorithm will lead to divergent behaviour. Moreover, given the large values for the cognitive and social control parameters, the divergence of the IPSO-CLL algorithm is expected to be rapid.

Figure 8h depicts that the average particle movement exhibited by the IPSO-CLL algorithm was immediately above the maximal value of 2000. Despite the divergent behaviour, the IPSO-CLL algorithm always employed convergent parameters, as evidenced by Figure 9h. The only reasonable explanation for this anomalous event was that the parameter values were generated sufficiently close to the boundaries of the convergent region such that the movement values were divergent; this is likely an example of the slight inaccuracy associated with using Δ to classify convergence (see Section 3.1). As Figure 10h indicates, the adaptation mechanism completely fails to adapt the param-

eters, leading to no change in parameters over time. Finally, as indicated in Figure 11h, particles immediately exit feasible space and never return. Therefore, the adaptation mechanism of the IPSO-CLL algorithm is ineffective at adapting the control parameters.

4.9 Adaptive inertia weight particle swarm optimization

The adaptive inertia weight particle swarm optimization (AIWPSO) algorithm (Nickabadi et al, 2011) uses the success rate of the swarm as feedback to adapt the inertia weight. The success rate of the swarm at time t is defined as the proportion of particles which improved their personal best position during iteration t. The inertia weight is then adapted according to

$$\omega(t) = (\omega_{max} - \omega_{min})P_s(t) + \omega_{min} \tag{24a}$$

with

$$P_s(t) = \frac{\sum_{i=1}^{n_s} S_i(t)}{n_s} \tag{24b}$$

and

$$S_i(t) = \begin{cases} 1 & \text{if } f(\mathbf{y}_i(t)) < f(\mathbf{y}_i(t-1)) \\ 0 & \text{otherwise.} \end{cases} \tag{24c}$$

The justification for this behaviour is that the algorithm increases the inertia weight when particle successes are high to heighten exploration and to decrease the inertia weight when particle successes are low to enhance exploitation. Typically, higher success rates are attained early in a search when the fitness values of particles are rapidly improving. Therefore, the inertia weight of the AIWPSO algorithm is expected to be relatively large initially. However, the inertia weight is also expected to decrease over time as fitness improvements become more difficult to attain. The AIWPSO algorithm thus tunes the value of the ω parameter but introduces two additional parameters, namely ω_{min} and ω_{max}.

Nickabadi et al (2011) used parameters $\omega_{min} = 0.0$ and $\omega_{max} = 1.0$ while no indication of the social and cognitive control parameters was given. Therefore, the commonly employed parameters of $c_1 = c_2 = 1.496180$ (van den Bergh and Engelbrecht, 2006) are assumed. As demonstrated in Harrison et al (2016a), the AIWPSO algorithm will exhibit convergent behaviour when $P_s(t) < 0.78540$. Due to the stochastic, exploratory nature of the PSO algorithm, it would be unrealistic to sustain a success rate greater than 78.54% for an extended period of time. Thus, the AIWPSO algorithm is expected to exhibit convergent behaviour.

As expected, Figure 8i demonstrates the average particle movement is well below Δ_{max}, while Figure 9i depicts that the generated parameter configurations are always within the convergent region. These two observations provide conclusive evidence that the high rate of improvement necessary for

non-convergent behaviour is unreasonable and will likely not be achieved in practice. However, despite the convergent behaviour, the particle movement in the AIWPSO is strikingly low which suggests that the algorithm suffers from premature convergence. This is further supported by the parameter movement immediately stagnating, as shown in Figure 10i. Thus, the adaptation mechanism of AIWPSO struggles to effectively adapt the parameters during the search. The average percentage of particle bound violations, shown in Figure 11i, is by far the lowest among all the algorithms. Over 80% of the particles are within the search space after only a few iterations while only four of the remaining algorithms even attain 80% feasible solutions within 5000 iterations.

4.10 Adaptive particle swarm optimization based on velocity information

The adaptive parameter tuning of particle swarm optimization based on velocity information (APSO-VI) algorithm (Xu, 2013) adapts the inertia weight based on the current velocities of the particles, with the intention of pushing the velocity closer to an "ideal" velocity. The concept of a decreasing target velocity in the APSO-VI algorithm is borrowed from earlier work by Yasuda et al (2008), which proposed adapting the inertia weight in a fully-informed particle swarm to control exploration and exploitation. In the APSO-VI algorithm, the average velocity of the swarm is calculated as

$$\overline{v(t)} = \frac{1}{n_d n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n_d} |v_{ij}(t)| \tag{25}$$

where n_d and n_s represent the number of problem dimensions and the size of the swarm, respectively. An equation defining the ideal (average) velocity was proposed whereby the ideal velocity decreases over time, leading to heightened exploitation nearer to the end of the search. Furthermore, Xu (2013) posits that such an ideal velocity will be non-linear, having long exploratory and exploitative phases with a minimal transition period. The ideal velocity at time t is defined according to

$$v_{ideal}(t) = v_s \left(\frac{1 + \cos\left(\pi \frac{t}{T_{0.95}}\right)}{2} \right) \tag{26}$$

where v_s is the initial ideal velocity, given by $\frac{x_{max} - x_{min}}{2}$, and $T_{0.95}$ is the point in which 95% of the search is complete. This ideal velocity function is visualized in Figure 5.

The APSO-VI algorithm then dynamically adapts the inertia weight each iteration based on the average velocity in relation to the ideal velocity as

$$\omega(t+1) = \begin{cases} \max\{\omega(t) - \Delta\omega, \omega_{min}\} & \text{if } \overline{v(t)} \geq v_{ideal}(t+1) \\ \min\{\omega(t) + \Delta\omega, \omega_{max}\} & \text{if } \overline{v(t)} < v_{ideal}(t+1) \end{cases} \tag{27}$$

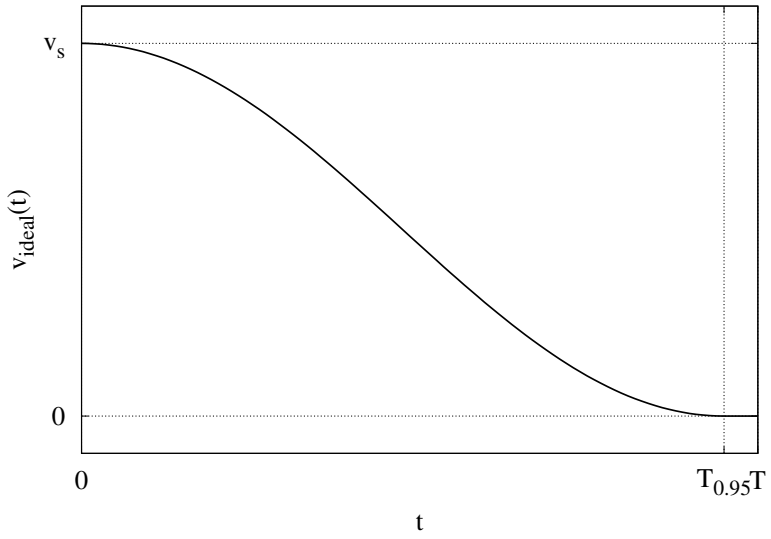


Fig. 5: The ideal velocity of APSO-VI as a function of time.

where ω_{min} and ω_{max} are the minimum and maximum inertia weights, respectively, and $\Delta\omega$ is the step size of the inertia weight. The APSO-VI algorithm thus tunes the value of the ω parameter but introduces three additional parameters, namely ω_{min}, ω_{max}, and $\Delta\omega$.

Xu (2013) used parameters $\omega_{min} = 0.3$, $\omega_{max} = 0.9$, and $\Delta\omega = 0.1$ with $c_1 = c_2 = 1.496180$. As demonstrated by Harrison et al (2016a), the decreasing ideal velocity will influence the inertia weight value to decrease such that it is within the convergent range, thereby causing the APSO-VI algorithm to demonstrate convergent behaviour.

Figure 8j depicts a gradual, smooth decline in particle movement over time. This is to be expected given that the APSO-VI algorithm is premised on explicitly controlling the average velocity and, by extension, the particle movement sizes. The percentage of convergent particles over time, shown in Figure 9j, depicts an inverse relationship with the particle movement. That is, the number of particles with convergent parameters shows a gradual increasing trend over time such that after approximately 4000 (80%) iterations, the entire swarm has convergent parameters. It is noted that around the same time that all particles employ convergent parameter configurations, i.e., around 4000 iterations, the parameter movement also stabilizes. This is evidenced by Figure 10j which shows the parameter adaptation abruptly ceases just after 4000 iterations. Finally, the percentage of particles outside the feasible region over time is shown in Figure 11j and indicates that almost all particles immediately exit the feasible region and do not return until nearly half of the search has completed. However, during the second half of the search, particle

bound violations steadily decrease, eventually stabilizing around 6%. Thus, the adaptation mechanism of APSO-VI is successfully controlling the inertia weight.

4.11 Self-regulating particle swarm optimization

The self-regulating particle swarm optimization (SRPSO) algorithm (Tanweer et al, 2015) controls the inertia weight of each particle such that the inertia weight value is increased for the best particle and decreased for all other particles. This adaptation scheme is premised on the idea that the best particle of the swarm should have a high level of confidence in its direction and thus accelerate quicker. Additionally, the remainder of particles should follow a linearly decreasing inertia weight strategy similar in principle to the PSO-TVIW algorithm. The inertia weight in the SRPSO algorithm is given by

$$\omega_i(t) = \begin{cases} \omega_i(t-1) + \eta\Delta\omega & \text{for the best particle} \\ \omega_i(t-1) - \Delta\omega & \text{for all other particles} \end{cases} \tag{28a}$$

with

$$\Delta\omega = \frac{\omega_s - \omega_f}{T} \tag{28b}$$

where η is a constant to control the rate of acceleration, ω_s and ω_f are the initial and final values of the inertia weight, and T is the maximum number of iterations. The AIWPSO algorithm thus tunes the value of the ω parameter but introduces three additional parameters, namely ω_s, ω_f, and η.

Tanweer et al (2015) employed parameters of $\omega_s = 1.05, \omega_f = 0.5$, $\eta = 1$, and $c_1 = c_2 = 1.49445$ in their study. Using a slightly different parametrization (namely, $\omega_s = 0.9, \omega_f = 0.4, \eta = 1$, and $c_1 = c_2 = 1.496180$), Harrison et al (2016a) demonstrated that the SRPSO algorithm can only lead to convergent behaviour when $1227.86 < t < 11378.50$. The aforementioned condition was derived with the following assumptions: a swarm size of 30, 5000 iterations were to be executed, and each particle was equally likely to be have the best fitness at any iteration. Using the same assumptions, it follows that the parametrization used in this paper can only lead to convergent behaviour when

$$2572.69 < t < 11816.70.$$

Therefore, the SRPSO algorithm will only exhibit convergence after 2573 iterations under ideal conditions, which are likely problem-dependent due to the assumption that all particles are equally likely to be the global best.

Figure 8k depicts the particle movement over time for the SRPSO algorithm. As expected, the algorithm is divergent for the first half of the search. However, it takes slightly longer than the (ideal) theoretically-predicted 2572 iterations to begin exhibiting convergent behaviour, suggesting that the observed probability of particles attaining the best position was not uniform. Furthermore, the movement values only briefly fall below the threshold before

gradually increasing back to the maximum value of 2000. Figure 9k shows that after roughly half of the search has completed, only 3.3% of the particles (i.e., a single particle) employ non-convergent control parameters. This further evidences that the best position is predominantly obtained by a single particle, thereby causing this particle to exhibit divergent behaviour. As shown in Figure 10k, no parameter changes occur after 5000 iterations. However, as shown in Figure 11k, the percentage of particles with a bound violation decreases near the end of the search. Despite nearly all of the particles initially exiting the search space, particles begin re-entering the feasible region after approximately 3000 iterations.

4.12 Particle swarm optimization with individual coefficients adjustment

The self-adaptive particle swarm optimization with individual coefficients adjustment (PSO-SAIC) algorithm (Wu and Zhou, 2007) adapts the inertia and social acceleration coefficients of each particle based on its fitness in relation to the global best fitness. This technique facilitates diversity injection when particles are near the global best position.

Wu and Zhou (2007) first define the related distance for particle i as

$$\xi_i(t) = \begin{cases} 0 & \text{if } f(\mathbf{x}_i(t-1)) = 0 \\ \frac{f(\mathbf{x}_i(t-1)) - f(\hat{\mathbf{y}}(t-1))}{f(\mathbf{x}_i(t-1))} & \text{otherwise} \end{cases} \tag{29}$$

which quantifies the efficacy of particle i at time t. Clearly, $\xi_i(t) = 1$ when particle i is far from the global best position and, conversely, $\xi_i(t) = 0$ when particle i is near the global best. Note that the definition of related distance, as defined in Eq. (29), presupposes a minimization problem with a positive global minimum.

The inertia weight is then adapted according to

$$\omega_i(t) = \omega_a F(\xi_i(t)) + \omega_b \tag{30a}$$

with

$$F(\xi_i(t)) = 2\left(1 - \cos\left(\frac{\pi\xi_i(t)}{2}\right)\right) \tag{30b}$$

while the social acceleration coefficient is calculated as

$$c_{2i}(t) = c_{2a} G(\xi_i(t)) + c_{2b} \tag{31a}$$

with

$$G(\xi_i(t)) = 2.5\left(1 - \cos\left(\frac{\pi\xi_i(t)}{2}\right)\right). \tag{31b}$$

The PSO-SAIC algorithm thus tunes the value of the ω and c_2 parameters but introduces four additional parameters, namely ω_a, ω_b, c_{2a}, and c_{2b}. Note that the constants 2 and 2.5 in Eqs. (30b) and (31b) may also be treated as

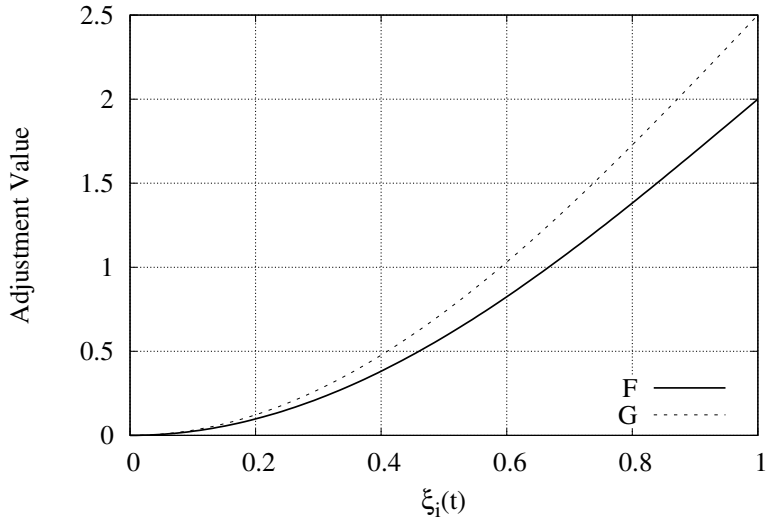


Fig. 6: The adjustment values described by Eqs. (30b) and (31b) for the PSO-SAIC algorithm.

control parameters. Figure 6 presents the adjustment values as calculated by Eqs. (30b) and (31b).

Wu and Zhou (2007) employed parameters of $\omega_a = 0.9$, $\omega_b = 0.45$, $c_{2a} = 0.5$, $c_{2b} = 2.5$, and $c_1 = 2.05$. It follows from Eq. (30) that

$$\forall i, t : 0.45 \leq \omega_i(t) \leq 2.25,$$

and from Eq. (31) that

$$\forall i, t : 2.5 \leq c_{2i}(t) \leq 3.75.$$

Therefore,

$$\forall i, t : c_1 + c_{2i}(t) \geq 4.55,$$

which indicates that the PSO-SAIC can strictly never lead to convergent behaviour.

In line with the theoretical expectations, the average particle movement over time, presented in Figure 8l, immediately reaches the maximum value and never decreases. This is a result of the inability to generate convergent parameter configurations which is empirically evidenced in Figure 9l. Due to the rapid divergence, invalid fitness values (which exceeded the limitation of a 32-bit floating point value) were obtained, thereby causing Eq. (29) to fail at producing feasible values for the related distance. Moreover, this resulted in invalid values calculated for the average change in parameters, causing Figure 10l to be empty. As should be expected, all particles immediately exited the

feasible region and never returned, as shown in Figure 11l. Thus, the adaptation mechanism of the PSO-SAIC algorithm is flawed in such an extreme manner that the entire algorithm fails to even complete the search.

4.13 Particle swarm optimization with random acceleration coefficients

As a baseline, the particle swarm optimization with random acceleration coefficients (PSO-RAC) algorithm employs randomly-generated, convergent control parameters. Specifically, a new set of control parameters, which explicitly adhere to the convergence criterion outlined in Eq. (3), is (randomly) generated for each particle at each iteration. Thus, the PSO-RAC algorithm tunes each of the ω, c_1, and c_2 parameters and requires no additional control parameters. The PSO-RAC algorithm is therefore used as a control to ascertain how the various SAPSO algorithms compare to a solely random parameter maintenance strategy. Note that by definition, the PSO-RAC algorithm will always exhibit convergent behaviour but should, in theory, demonstrate relatively large parameter changes.

Figures 8m and 9m, which plot the average particle movement and percentage of particles with convergent parameters over time, depict that the PSO-RAC algorithm exhibits non-divergent behaviour. Examining Figure 10m, it is noted that the average parameter movement is approximately 1.6 at each iteration. Given that this strategy randomly selects the parameters each iteration, a reasonable SAPSO strategy should (ideally) exhibit average parameter movement values below 1.6. Finally, Figure 11m plots the number of particles with a bound violation over time which demonstrates that after 5000 iterations, more than 40% of the particles are infeasible. However, it can be argued that the high proportion of particles outside the search space is a result of the PSO-RAC algorithm making no attempt at exploitation near the end of the search. Rather, the algorithm attempts to maintain an equal balance between exploration and exploitation throughout the entire search. While the relatively large number of particles outside the search space seems problematic, it should be noted that the search space defined by Eq. (5) is highly irregular and is largely meant to examine particle convergence by preventing complete stagnation. Therefore, it is not unreasonable to assume particles will exit and remain outside the feasible region to a larger extent than cases when a less irregular search space is employed.

4.14 Particle swarm optimization with time-varying acceleration coefficients

The particle swarm optimization with time varying acceleration coefficients (PSO-TVAC) algorithm (Ratnaweera et al, 2004) linearly varies the cognitive and social coefficients over time. The cognitive coefficient is decreased while the social coefficient is increased over time to hopefully provide a smooth transition from exploration to exploitation as the search progresses. The cognitive and

social coefficients at time t are linearly scaled each iteration according to

$$c_1(t) = c_{1s} + (c_{1f} - c_{1s})\frac{t}{T} \tag{32a}$$

$$c_2(t) = c_{2s} + (c_{2f} - c_{2s})\frac{t}{T} \tag{32b}$$

where the subscripts s and f represent the initial and final values, respectively. PSO-TVAC also employs the linearly decreasing inertia weight of PSO-TVIW described in Eq. (8). The PSO-TVAC algorithm thus tunes the values for each of the ω, c_1, and c_2 parameters but introduces six additional control parameters, namely ω_i, ω_f, c_{1s}, c_{1f}, c_{2s}, and c_{2f}.

Parameters for the PSO-TVAC algorithm were set at $\omega_i = 0.9, \omega_f = 0.4$, $c_{1s} = 2.5, c_{1f} = 0.5$, and $c_{2s} = 0.5, c_{2f} = 2.5$. Given that $c_1(t)$ and $c_2(t)$ have an inverse relationship, it is trivial to see that

$$\forall t : c_1(t) + c_2(t) = 3.$$

It follows from Eq. (3) that

$$-0.15934 < \omega(t) < 0.78436$$

is the necessary condition for convergent behaviour and that this condition is satisfied when $\frac{t}{T} = 0.23128$. Thus, the PSO-TVAC algorithm will only demonstrate convergent behaviour after approximately 23% of the search has completed.

As Figure 8n depicts, the PSO-TVAC algorithm exhibits initial divergent behaviour but demonstrates a rapid decrease in particle movement slightly after 1000 iterations (20%) have passed. The rapid decrease is caused by an immediate switch from all particles divergent parameter configurations to convergent configurations, as evidenced in Figure 9n. Given the linear nature of the parameter adaptation mechanism, the average change in parameters, shown in Figure 10n, is constant at 5.74E-4. Finally, the percentage of particles with bound violations, presented in Figure 11n, demonstrates that the initial explosive movements cause particles to initially exit the feasible search space. However, after roughly 1000 iterations (and coinciding with the switch from divergent to convergent behaviour), the PSO-TVAC algorithm depicts a smooth decline in bound violations over the remainder of the search. After 5000 iterations, only 6.5% of the particles (i.e., 2 particles) are outside the feasible region, on average. Thus, despite not being a truly self-adaptive algorithm, the PSO-TVAC algorithm demonstrates relative good search behaviours in relation to the other algorithms.

4.15 Particle swarm optimization with simulated annealing

The particle swarm optimization with simulated annealing (PSO-ICSA) algorithm (Jun and Jian, 2009) adapts both the inertia weight and social acceleration coefficients. Firstly, the "adaptive coefficient" of particle i at time t,

given by

$$\eta_i(t) = \frac{f(\hat{\mathbf{y}}(t-1))}{f(\mathbf{x_i}(t-1))}, \tag{33}$$

quantifies the performance of the particle. The adaptive coefficient measures the similarity of a particle's fitness relative the global best fitness; $\eta_i(t) \approx 0$ denotes that a particle's fitness is far from the global best while $\eta_i(t) = 1$ denotes the particle's fitness is equal to the global best fitness.

The inertia weight of a particle is then given by

$$\omega_i(t) = \omega_a F(\eta_i(t)) + \omega_b \tag{34a}$$

with

$$F(\eta_i(t)) = \begin{cases} 2 & \text{if } \eta_i(t) < 0.0001 \\ 1 & \text{if } 0.0001 \leq \eta_i(t) < 0.01 \\ 0.3 & \text{if } 0.01 \leq \eta_i(t) < 0.1 \\ -0.8 & \text{if } 0.1 \leq \eta_i(t) < 0.9 \\ -5.5 & \text{if } 0.9 \leq \eta_i(t) \leq 1 \end{cases} \tag{34b}$$

where ω_a and ω_b are user-supplied, positive constants. Note that when $\eta_i(t)$ is low, the inertia weight is increased to enhance exploration while high values of $\eta_i(t)$ lead to decreased inertia, thereby enhancing exploitation.

The social acceleration coefficient of a particle is given by

$$c_{2i}(t) = c_{2a} G(\eta_i(t)) + c_{2b} \tag{35a}$$

with

$$G(\eta_i(t)) = \begin{cases} 2.5 & \text{if } \eta_i(t) < 0.0001 \\ 1.2 & \text{if } 0.0001 \leq \eta_i(t) < 0.01 \\ 0.5 & \text{if } 0.01 \leq \eta_i(t) < 0.1 \\ 0.2 & \text{if } 0.1 \leq \eta_i(t) < 0.9 \\ 0.1 & \text{if } 0.9 \leq \eta_i(t) \leq 1 \end{cases} \tag{35b}$$

where c_{2a} and c_{2b} are user-supplied, positive constants. When $\eta_i(t)$ is low, the social acceleration coefficient is increased as an attempt to attract the particle towards the global best. This is an explicit attempt to increase the speed of convergence. When $\eta_i(t)$ is high, the social acceleration coefficient is decreased to discourage crowding around the global best position. The cognitive coefficient, while not truly adapted, is decreased linearly according to Eq. (32a). The PSO-ICSA algorithm thus tunes the values for each of the ω, c_1, and c_2 parameters but introduces six additional control parameters, namely ω_a, ω_b, c_{1s}, c_{1f}, c_{2a}, and c_{2b}. Furthermore, the function values and the corresponding piece-wise boundaries in Eqs. (34b) and (35b) may require tuning, in which case the PSO-ICSA algorithm introduces a further 18 parameters.

Jun and Jian (2009) used the parameters $\omega_a = 0.9, \omega_b = 0.45$, $c_{2a} = 0.5, c_{2b} = 2.5$, and $c_{1s} = 2.5, c_{1f} = 0.5$. From Eqs. (34b) and (35b), it is clear that there are five distinct scenarios which can occur:

1. $\eta_i(t) < 0.0001 \mapsto \omega = 2.25, c_2 = 6.75$
2. $0.0001 \leq \eta_i(t) < 0.0100 \mapsto \omega = 1.35, c_2 = 3.50$
3. $0.0100 \leq \eta_i(t) < 0.1000 \mapsto \omega = 0.72, c_2 = 1.75$
4. $0.1000 \leq \eta_i(t) < 0.9000 \mapsto \omega = -0.27, c_2 = 1.00$
5. $0.9000 \leq \eta_i(t) \leq 1.0000 \mapsto \omega = -4.50, c_2 = 0.75$

Cases (1), (2), and (5) – $\eta_i(t) < 0.01$ or $\eta_i(t) \geq 0.9$: None of the respective inertia weight values of 2.25, 1.35, and -4.5, resulting from each of these scenarios, satisfy the convergence criterion outlined in Eq. (3). Therefore, convergent behaviour will never be exhibited.

Case (3) – $0.01 \leq \eta_i(t) < 0.1$: With an inertia weight value of 0.72 and a social coefficient of 1.75, it follows from Eq. (3) that $c_1(t) < 1.64853$ is necessary for convergent behaviour. This can only occur when $\frac{t}{T} = 0.42574$, i.e., after roughly 42.6% of the search is completed. Therefore, scenario (3) can only lead to convergent behaviour after 42.6% of the search is completed.

Case (4) – $0.1 \leq \eta_i(t) < 0.9$: With an inertia weight value of -0.27 and a social coefficient of 1, it follows from Eq. (3) that $c_1(t) < 2.08378$ is necessary for convergent behaviour. This can only occur when $\frac{t}{T} = 0.20811$, or roughly 20.8% of the search has completed. Therefore, scenario (4) can only lead to convergent behaviour after 20.8% of the search is completed.

Summary: During the first 20.8% of the search process, the PSO-ICSA strictly cannot satisfy the convergence criterion. Moreover, there are only two (unlikely) scenarios which can lead to the convergence criterion to be satisfied. The first scenario requires roughly 20.8% of the search process to be completed, while the second scenario requires roughly 42.6% of the search to be completed. In either scenario, all particles must have a fitness value which is neither too close nor too far from the fitness of the global best position. Given the extremely strict requirements for convergence, the PSO-ICSA algorithm is expected to exhibit overall divergent behaviour.

Figure 8o presents the average particle movement over time for the PSO-ICSA algorithm and clearly demonstrates that convergent behaviour is never exhibited. Furthermore, the percentage of particles with convergent parameters shown in Figure 9o is never above 0%, indicating that the adaptation mechanism completely fails to produce convergent parameters. Moreover, the average change in parameters over time, shown in Figure 10o, is constant at a value of 4.00E-4, indicating that the only change in parameters is due to the linearly decreasing cognitive coefficient. Given the observed divergent behaviour, it is not unexpected that the bound violation percentage, shown in Figure 11o, immediately reaches 100% and never decreases. Thus, the adaptation mechanism of the PSO-ICSA is extremely flawed and causes immediate divergence.

4.16 Adaptive particle swarm optimization

The adaptive particle swarm optimization (APSO-ZZLC) algorithm (Zhan et al, 2009) adapts each of the three control parameters through the use of a

fuzzy classification system. The classification system classifies the current behaviour of the algorithm into one of four states: exploration (S_1), exploitation (S_2), convergence (S_3), or jumping out (S_4). To perform the classification, the authors first propose an evolutionary factor[4] based on the spread of particles in the search space given by

$$f_e(t) = \frac{d_g(t) - d_{min}(t)}{d_{max}(t) - d_{min}(t)} \tag{36a}$$

with

$$d_i(t) = \frac{1}{n_s - 1} \sum_{j=1, j\neq i}^{n_s} \sqrt{\sum_{k=1}^{n_d} (x_{ik}(t) - x_{jk}(t))^2} \tag{36b}$$

where $d_g(t)$ is the value of Eq. (36b) for the global best position and $d_{min}(t)$ and $d_{max}(t)$ are the minimum and maximum observed values of $d_i(t)$. Note that $d_i(t)$ is the average Euclidean distance of particle i to all other particles in the swarm. The fuzzy membership value for each of the four algorithmic states, depicted in Figure 7, are thus dependent upon the value of f_e given by the piece-wise functions as follows:

Exploration

$$\mu_{S_1}(f_e(t)) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.4 \\ 5f_e(t) - 2 & \text{if } 0.4 < f_e(t) \leq 0.6 \\ 1 & \text{if } 0.6 < f_e(t) \leq 0.7 \\ -10f_e(t) + 8 & \text{if } 0.7 < f_e(t) \leq 0.8 \\ 0 & \text{if } 0.8 < f_e(t) \leq 1.0 \end{cases} \tag{37a}$$

Exploitation

$$\mu_{S_2}(f_e) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.2 \\ 10f_e(t) - 2 & \text{if } 0.2 < f_e(t) \leq 0.3 \\ 1 & \text{if } 0.3 < f_e(t) \leq 0.4 \\ -5f_e(t) + 3 & \text{if } 0.4 < f_e(t) \leq 0.6 \\ 0 & \text{if } 0.6 < f_e(t) \leq 1.0 \end{cases} \tag{37b}$$

Convergence

$$\mu_{S_3}(f_e) = \begin{cases} 1 & \text{if } 0.0 \leq f_e(t) \leq 0.1 \\ -5f_e(t) + 1.5 & \text{if } 0.1 < f_e(t) \leq 0.3 \\ 0 & \text{if } 0.3 < f_e(t) \leq 1.0 \end{cases} \tag{37c}$$

Jumping-out

$$\mu_{S_4}(f_e) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.7 \\ 5f_e(t) - 3.5 & \text{if } 0.7 < f_e(t) \leq 0.9 \\ 1 & \text{if } 0.9 < f_e(t) \leq 1.0 \end{cases} \tag{37d}$$

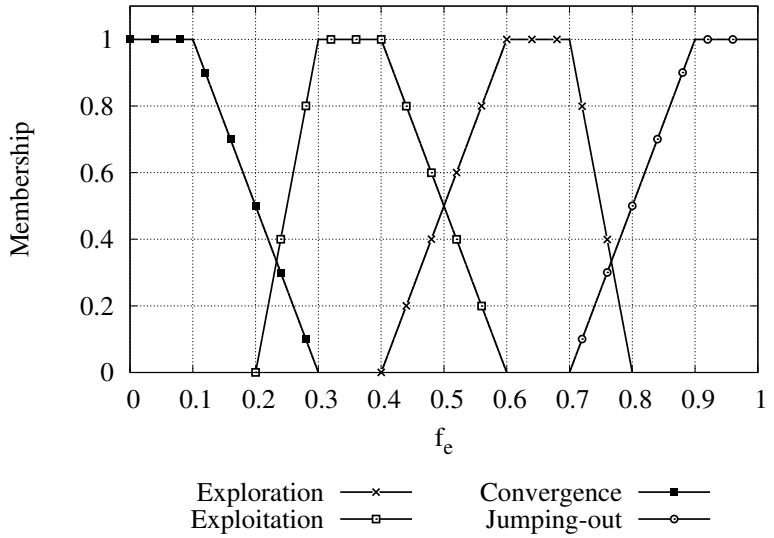


Fig. 7: The fuzzy membership functions of the APSO-ZZLC algorithm for the four evolutionary states.

Due to the possibility of having a degree of membership to multiple states simultaneously, a defuzzification process must be employed to provide a singular classification. Zhan et al (2009) expect that the algorithm will transition according to the following sequence $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_1 ...$ and therefore the defuzzification procedure must account for this (see Rule 2 below). The defuzzification process consists of three rules, in decreasing order of priority, as follows.

1. If the degree of membership to the current state is non-zero, there is no change in state. This provides classification stability by preventing excessive changes.
2. If the degree of membership to the next state in the sequence is non-zero, the state transitions to the next state in the sequence (i.e., $S_1 \rightarrow S_2$ or $S_3 \rightarrow S_4$).
3. The current state is selected as the state with the highest degree of membership.

Once the defuzzification process has determined a singular classification, the control parameters can be calculated. The inertia weight is provided by

$$\omega(f_e) = \frac{1}{1 + 1.5e^{-2.6f_e}} \in [0.4, 0.9] \tag{38}$$

[4] Again, the original terminology is retained despite the PSO algorithm not exhibiting evolution.

Table 2: Parameter control strategies for the cognitive and social coefficients in the APSO-ZZLC algorithm.

Algorithmic State	**Cognitive Coefficient**	**Social Coefficient**
S_1 – Exploration	Increase	Decrease
S_2 – Exploitation	Increase slightly	Decrease slightly
S_3 – Convergence	Increase slightly	Increase slightly
S_4 – Jumping-out	Decrease	Increase

while the cognitive and social coefficients are either increased or decreased based on the algorithmic state, as described in Table 2. An entry which indicates an "Increase" denotes that the corresponding control parameter is increased by $\delta(t)$ and an entry marked "Decrease" denotes the corresponding parameter is reduced by $\delta(t)$, where $\delta(t) \sim U(0.05, 0.1)$. Entries marked as "Increase slightly" or "Decrease slightly" are incremented or decremented by $0.5\delta(t)$, respectively. Both the social and cognitive coefficients are clamped to the range [1.5, 2.5]. Furthermore, if the sum of the cognitive and social coefficients is greater than 4.0, the coefficients are each normalized to

$$c_i = 4.0\frac{c_i}{c_1 + c_2}, \;\; i = 1, 2$$

to effectively bound the range of $c_1 + c_2$ to [3.0, 4.0].

Note that, while the APSO-ZZLC algorithm can be employed without any additional parameters, there are numerous constants which may be treated as control parameters. Specifically, for each of the membership functions in Eq. (37), there are a total of nine constant values (five for the function values and four for the piece-wise boundaries), which leads to an additional 36 possible parameters. Furthermore, Eq. (38) defines two constant values, namely 1.5 and -2.6, which can be seen as additional control parameters. Thus, the APSO-ZZLC can be considered to have 38 additional parameters if these values are taken to be tunable.

Zhan et al (2009) used $\omega = 0.9$ and $c_1 = c_2 = 2$ as initial values for the control parameters, leading to initially divergent behaviour. Based on their respective adaptation mechanisms, both the cognitive and social coefficients are expected to tend towards 2 (Zhan et al, 2009), which leads to the condition of

$$\frac{1}{3} < \omega(t) < \frac{1}{2}$$

for convergent behaviour to be exhibited. Assuming that $c_1 + c_2 = 4$ (note the conditions for convergence become easier to satisfy if $c_1 + c_2 < 4$), then $\omega(t)$ will be within the convergent range when $-0.11065 < f_e(t) < 0.15595$. Revisiting Eq. (36), it is expected that the global best should, in general, be near the centre of the swarm, thereby causing $d_g(t) \simeq d_{min}(t)$. When $d_g(t) \simeq d_{min}(t)$, the value of $f_e(t)$ should tend towards 0. Given that in the strictest case, $f_e(t) < 0.15595$ is necessary for convergent behaviour, the APSO-ZZLC algorithm should exhibit convergent behaviour. However, given that $c_1 + c_2$

will tend towards 4, the particle movement sizes are expected to be rather large (Cleghorn and Engelbrecht, 2014b).

Directly in line with the theoretical expectations, Figure 8p depicts an average particle movement which is just slightly below Δ_{max} over the entirety of the search. Furthermore, Figure 9p depicts that the generated parameter configurations always adhere to the convergence criterion of Poli. As Figure 10p depicts, the parameter configurations are extremely stable and after 5000 iterations, are only changing by an average of 4.51E-5. As such, the APSO-ZZLC algorithm could be naively classified as convergent. However, as Figure 11p depicts, nearly all particles exit the feasible region and never return. Thus, despite having theoretically convergent parameters at all times, the APSO-ZZLC algorithm exhibits such large particle movement values that it is incapable of retaining particles within the feasible search space.

4.17 Adventurous unified particle swarm optimization

The adventurous unified adaptive particle swarm optimization (UAPSO-A) algorithm (Hashemi and Meybodi, 2009, 2011) adapts each control parameter using an independent learning automaton. Using a set of learning automata, the UAPSO-A algorithm takes the performance of the current parameters as feedback to control their probability of selection in the future.

Learning automata are a type of machine learning algorithm used to probabilistically select an action from a set. At each step, an action is selected and applied to the given environment. Immediately after the application of the action, the environment evaluates the action and returns a reinforcement signal back to the automaton, which then interprets this signal and uses it to improve the selection probabilities. When an action was successful, the learning automaton will increase the probability of selecting this action again, while an unsuccessful action will have the selection probability decreased.

In the context of parameter selection, the UAPSO-A algorithm employs three learning automata – one for each of the PSO control parameters. The set of actions (i.e., parameter values) in each automaton are given by a user-supplied number of discrete values from the allowable range, namely n_ω equidistant values from $[\omega_{min}, \omega_{max}]$ are used for the inertia automaton while the cognitive and social automata are both provided n_c independent, equidistant values from the range $[c_{min}, c_{max}]$. At each iteration, one control parameter value is selected from each automaton, thereby providing values for each of $\omega, c_1,$ and c_2. Use of the selected parameters constitutes applying the action to the environment and their performance is used as the reinforcement signal. Initially, the probability of selection is equal for all parameter values. The success of the selected parameters is then determined based on the proportion of particles which have improved their fitness during the current iteration. If the proportion of particles which improved their fitness is greater than τ, the parameters are considered to be successful and each of the automata must be updated accordingly.

If a successful iteration was observed employing the parameter at index i, the probabilities are updated according to

$$p_j(t+1) = \begin{cases} p_j(t) + a(1 - p_j(t)) & \text{if } i = j \\ p_j(t)(1-a) & \text{otherwise} \end{cases} \tag{39}$$

where a is the reward step size. Note that the probability of selection is increased for the parameter which was successfully employed while the remainder of the parameters have their probabilities decreased slightly. Conversely, for an unsuccessful iteration employing parameter i, the probabilities are updated according to

$$p_j(t+1) = \begin{cases} p_j(t)(1-b) & \text{if } i = j \\ \frac{b}{|A|-1} + p_j(t)(1-b) & \text{otherwise} \end{cases} \tag{40}$$

where b is the penalty step size and $|A|$ is the number of actions in the automaton. Thus, the probability of selecting the unsuccessful parameter value is decreased after an unsuccessful iteration is observed. The UAPSO-A algorithm thus tunes the values for each of the ω, c_1, and c_2 parameters but introduces nine additional control parameters, namely $n_\omega, \omega_{min}, \omega_{max}, n_c, c_{min}, c_{max}, a, b$, and τ.

Hashemi and Meybodi (2009, 2011) used $\omega_{min} = 0$, $\omega_{max} = 1$, $n_\omega = 20$, $c_{min} = 0$, $c_{max} = 2$, $n_c = 10$, $a = b = 0.01$, and $\tau = 0.5$. Through the use of Eqs. (39) and (40), the respective automata will learn the values of each parameter which lead to successful behaviours, thereby improving the selection of parameters over time. Note that, while the UAPSO-A algorithm will guide the swarm towards successful parameters, there is no guarantee of convergent behaviour given that each parameter is selected independently. However, parameters which exhibit divergent behaviour will likely not demonstrate successful behaviour and thus will eventually be given smaller probabilities of selection, thereby promoting convergent behaviour. Thus, the UAPSO-A is expected to exhibit convergent behaviour.

Figure 8q shows the average particle movement over time for the UAPSO-A algorithm. This figure depicts relatively small initial movement values of approximately 500 which gradually decrease over time. The average percentage of particles with convergent parameter configurations, as shown in Figure 9q, fluctuates wildly between 60% and 80%. These results indicate that the UAPSO-A algorithm is able to retain overall convergent behaviour despite having between 20% and 40% of the particles exhibiting divergent tendencies at any given iteration. This suggests that even when particles do exhibit divergent behaviour, parameter configurations are subsequently adapted such that the divergent behaviour does not persist for an extended period of time. However, the significant portion of particles with divergent behaviours at any given time suggests that the divergent parameter configurations are never completely eliminated from consideration. Given the probabilistic nature of parameter selection, the relatively high average change in parameters, shown in Figure 10q,

is to be expected. Finally, the average percentage of bound violations is presented in Figure 11q, where the UAPSO-A demonstrates a smooth decrease in violations over time. After 5000 iterations, the UAPSO-A algorithm has only 38.1% of the particles outside the search space, on average, which is relatively low compared to the other examined algorithms.

4.18 Grey particle swarm optimization

The GPSO algorithm (Leu and Yeh, 2012) uses a measure of similarity for finite sequences under incomplete information, namely grey relational analysis (Ju-Long, 1982), to aid with control parameter adaptation. The grey relational analysis is used to modify the inertia weight and social coefficient based on the grey relational grade. To calculate the grey relational grade, the relational coefficient of particle i is first calculated according to

$$r_{ij}(t) = \frac{\Delta_{min}(t) + \xi\Delta_{max}(t)}{\Delta_{ij}(t) + \xi\Delta_{max}(t)} \tag{41a}$$

with

$$\Delta_{ij}(t) = |\hat{y}_j(t) - x_{ij}(t)| \tag{41b}$$

where j is the current dimension, $\Delta_{min}(t)$ and $\Delta_{max}(t)$ are the minimum and maximum values of $\Delta_{ij}(t)$, respectively, and $\xi \in (0, 1]$ controls the resolution between Δ_{max} and Δ_{min}. The grey relational coefficient of particle i is then given by

$$g_i(t) = \sum_{j=1}^{n_d} (\alpha_j r_{ij}(t)) \tag{42a}$$

such that

$$\sum_{j=1}^{n_d} \alpha_j = 1 \tag{42b}$$

where α_j is the weighting factor of the relational coefficient for dimension j and n_d is the number of dimensions. In general, it is acceptable to set $\alpha_j = \frac{1}{n_d}$ for all dimensions j (Leu and Yeh, 2012). However, the values of α_j can be taken as an additional n_d control parameters.

The relational grade is then used to adapt the inertia weight of each particle according to

$$\omega_i(t) = \frac{\omega_{min} - \omega_{max}}{g_{max}(t) - g_{min}(t)} g_i(t) + \frac{\omega_{max} g_{max}(t) - \omega_{min} g_{min}(t)}{g_{max}(t) - g_{min}(t)} \tag{43}$$

where $g_{min}(t)$ and $g_{max}(t)$ are the minimum and maximum relational grades at time t and ω_{min} and ω_{max} are the minimum and maximum inertia weights, respectively. Furthermore, the relational grade controls the social control parameter according to

$$c_{2i}(t) = \frac{c_{max}(t) - c_{min}(t)}{g_{max}(t) - g_{min}(t)} g_i(t) + \frac{c_{min}(t) g_{max}(t) - c_{max}(t) g_{min}(t)}{g_{max}(t) - g_{min}(t)} \tag{44}$$

where $c_{min}(t)$ and $c_{max}(t)$ are the linearly-varying minimum and maximum values for the social coefficient at time t governed by

$$c_{min}(t) = (C_{final} - C_{min})\frac{t}{T} + C_{min} \tag{45a}$$

$$c_{max}(t) = (C_{final} - C_{max})\frac{t}{T} + C_{max} \tag{45b}$$

where C_{min}, C_{max}, and C_{final} are user-supplied parameters such that

$$C_{min} \leq C_{final} \leq C_{max}.$$

Finally, the cognitive control parameter is given by

$$c_{1i} = 4.0 - c_{2i} \tag{46}$$

such that $\forall i : c_{1i} + c_{2i} = 4.0$. Note that, while the GPSO algorithm adapts all three control parameters, the cognitive coefficient is based solely on the social coefficient and therefore is not truly an adaptive parameter in itself. The GPSO algorithm thus tunes the values for each of the ω, c_1, and c_2 parameters but introduces six additional control parameters, namely ω_{min}, ω_{max}, C_{min}, C_{max}, C_{final}, and $\xi = 1.0$. Additionally, the values for α_j can also be viewed as control parameters, leading to an additional n_d parameters.

Leu and Yeh (2012) employed parameters of $\omega_{min} = 0.4, \omega_{max} = 0.9, C_{min} = 1.5, C_{max} = C_{final} = 2.5$, and $\xi = 1.0$. Note that the sum of the cognitive and social coefficients will always be 4.0, which leaves a very slim window for the inertia weight, specifically $\frac{1}{3} \leq \omega_i(t) \leq \frac{1}{2}$, to lead to convergent behaviour. By substituting the aforementioned parameters into Eq. (43) the condition for convergent behaviour simplifies to

$$\frac{1}{3} \leq \frac{0.9g_{max}(t) - 0.4g_{min}(t) - 0.5g_i(t)}{g_{max}(t) - g_{min}(t)} \leq \frac{1}{2},$$

which will always lead to $\omega_i(t) = 0.9$, and thereby divergent behaviour in at least one particle, namely the particle which exhibits $g_i(t) = g_{min}(t)$. Moreover, note that the only convergent inertia values that can be produced by Eq. (43) are those within the range $[0.4, 0.5]$ and that this range accounts for only 20% of the possible inertia values that Eq. (43) can produce. Thus, if the inertia weights calculated by Eq. (43) follow a uniform distribution over the range of $[0.4, 0.9]$, it is expected that only approximately 20% of the particles will exhibit convergent behaviour during each iteration. Therefore, the GPSO algorithm is expected to exhibit purely divergent behaviour.

Figure 8r presents the average particle movement over time for the GPSO algorithm and empirically confirms that the algorithm demonstrates purely divergent behaviour. Considering Figure 9r, it is observed that slightly fewer than 20% of the particles exhibit convergent behaviour during each iteration, which is directly in line with the theoretical expectation under the assumption that Eq. (43) uniformly distributes the inertia value. As Figure 10r depicts,

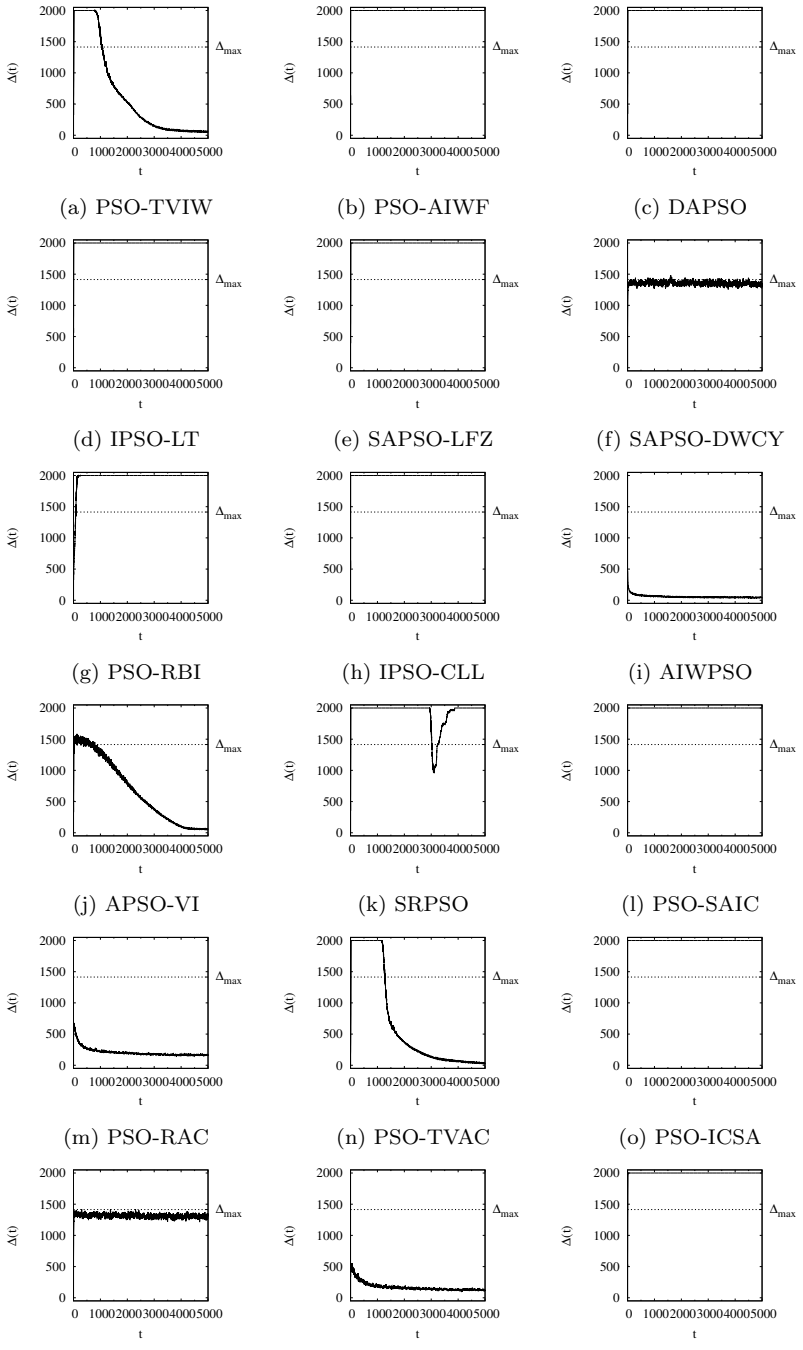
the average change in parameters is approximately 0.25 initially, but gradually decreases over time to a value of 0.835E-2, which is still relatively large in comparison to the other algorithms. Nonetheless, the parameter changes demonstrated by the GPSO are continually decreasing and therefore may lead to a stable set of parameters if given a greater number of iterations. However, a stable set of parameters will likely not be of benefit to the algorithm given that the entire swarm exits the feasible region and never returns, as evidenced by Figure 11r.

4.19 Summary of empirical results

This section provides a summary of the empirical results. Table 3 summarizes the algorithmic parameters employed while Table 4 provides an overview of the final measure values obtained by the examined algorithms in contrast to the canonical PSO. Examining Table 4, 10 of the 18 examined algorithms (not accounting for the canonical PSO) attained average particle movements at the maximum, capped value of 2000 suggesting that the true average movement values were even larger. A further two algorithms, namely SAPSO-DWCY and APSO-ZZLC, had average movement values within 10% of Δ_{max}, suggesting that these two algorithms are unlikely to result in a stabilizing swarm within a reasonable amount of time.

Of the 18 examined algorithms, five contained no particles with convergent parameters, indicating a complete failure of their respective adaptation mechanisms. Furthermore, only an additional two algorithms had fewer than 50% of the particles with convergent parameters. Of the 18 algorithms, only eight managed to result in convergent parameters being employed by all particles. Regarding the average parameter movement, eight of the algorithms depicted no change in parameters during the final iteration, while a majority depicted relatively small changes. With the exception of two algorithms, namely DAPSO and PSO-SAIC, which resulted in invalid parameter movement sizes, all algorithms exhibited parameter movement sizes below that of the baseline PSO-RAC. This suggests that, despite their respective failures, the adaptation mechanisms did achieve overall stability with respect to the resulting parameters. Finally, when considering the average percentage of particles which are outside the feasible search space, nine of the algorithms had greater than 95% of the particles outside the feasible region. However, due to the chaotic nature of the search space, it is not unreasonable for the algorithms to struggle with retaining particles within the search space, as evidenced by the canonical PSO having approximately 70% of the particles outside the feasible region after 5000 iterations. Nonetheless, five of the algorithms were able to retain greater than 90% of the particles within the search space, although none of which attained 100% feasibility.

A further noteworthy observation was that care must be taken with regards to the classification of algorithms as convergent or divergent based solely upon the particle movement sizes. As evidenced by the SAPSO-DWCY algorithm in



(a) PSO-TVIW (b) PSO-AIWF (c) DAPSO

(d) IPSO-LT (e) SAPSO-LFZ (f) SAPSO-DWCY

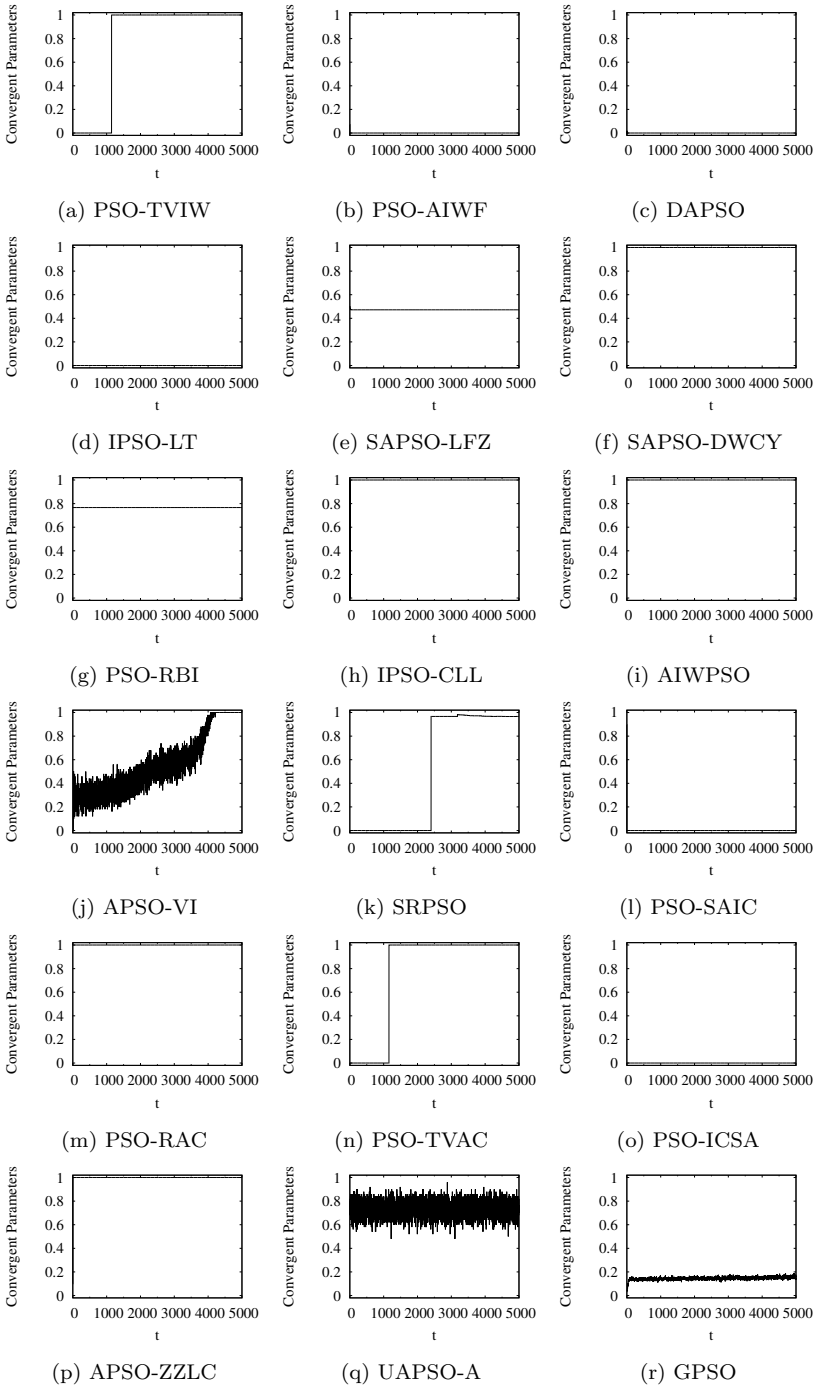
(g) PSO-RBI (h) IPSO-CLL (i) AIWPSO

(j) APSO-VI (k) SRPSO (l) PSO-SAIC

(m) PSO-RAC (n) PSO-TVAC (o) PSO-ICSA

(p) APSO-ZZLC (q) UAPSO-A (r) GPSO

Fig. 8: Average particle movement over time.



(a) PSO-TVIW (b) PSO-AIWF (c) DAPSO

(d) IPSO-LT (e) SAPSO-LFZ (f) SAPSO-DWCY

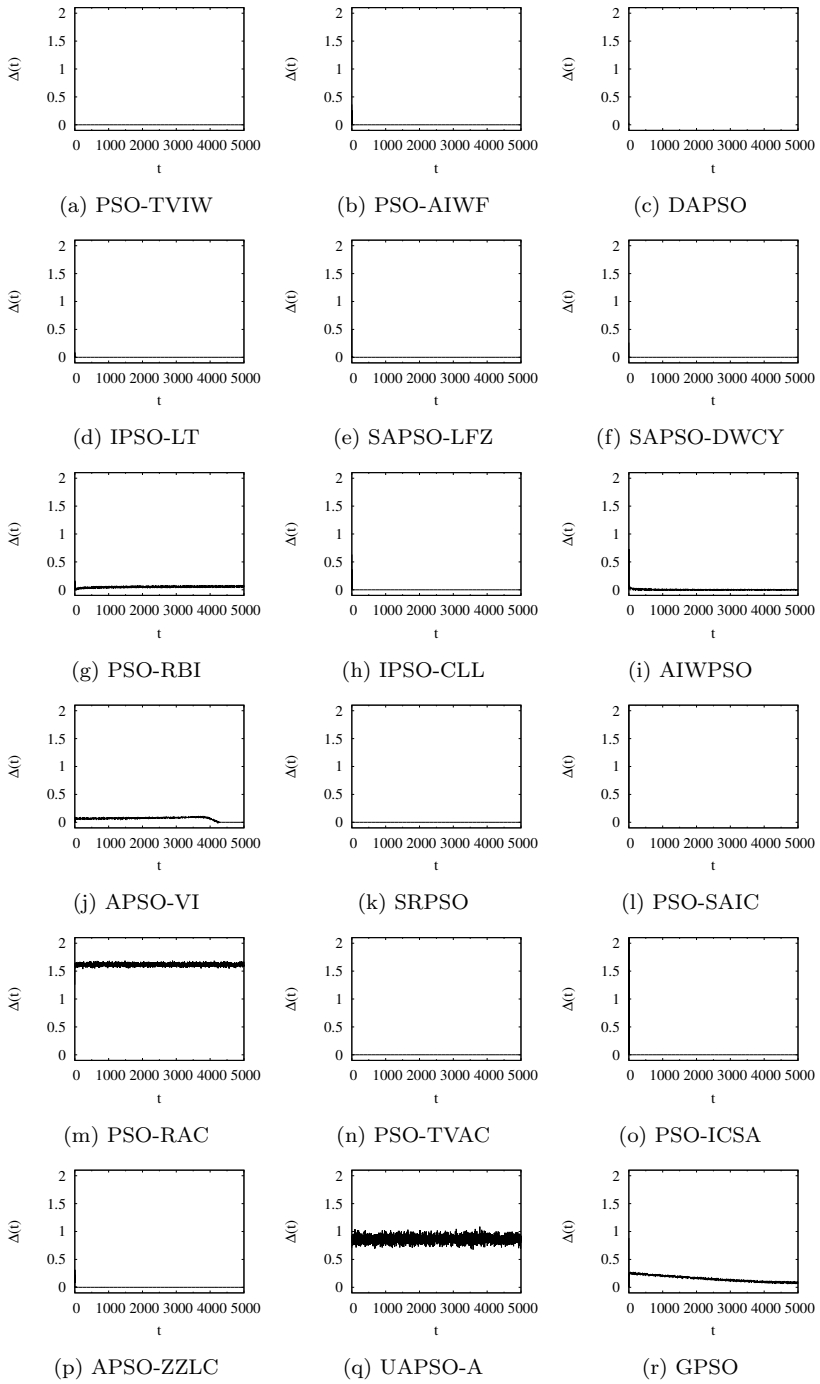
(g) PSO-RBI (h) IPSO-CLL (i) AIWPSO

(j) APSO-VI (k) SRPSO (l) PSO-SAIC

(m) PSO-RAC (n) PSO-TVAC (o) PSO-ICSA

(p) APSO-ZZLC (q) UAPSO-A (r) GPSO

Fig. 9: Percentage of particles with convergent control parameters.



(a) PSO-TVIW (b) PSO-AIWF (c) DAPSO

(d) IPSO-LT (e) SAPSO-LFZ (f) SAPSO-DWCY

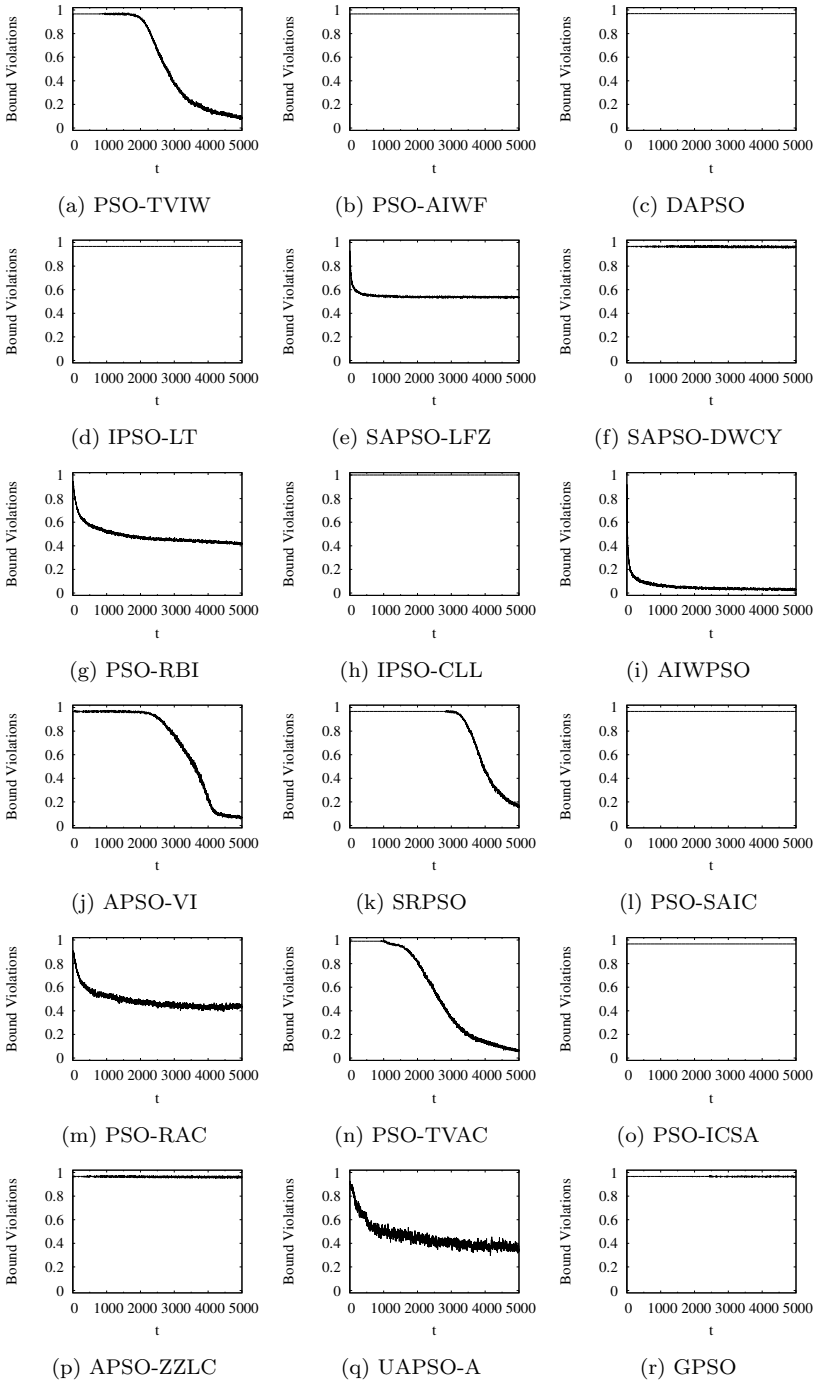
(g) PSO-RBI (h) IPSO-CLL (i) AIWPSO

(j) APSO-VI (k) SRPSO (l) PSO-SAIC

(m) PSO-RAC (n) PSO-TVAC (o) PSO-ICSA

(p) APSO-ZZLC (q) UAPSO-A (r) GPSO

Fig. 10: Average parameter movement over time.



(a) PSO-TVIW (b) PSO-AIWF (c) DAPSO

(d) IPSO-LT (e) SAPSO-LFZ (f) SAPSO-DWCY

(g) PSO-RBI (h) IPSO-CLL (i) AIWPSO

(j) APSO-VI (k) SRPSO (l) PSO-SAIC

(m) PSO-RAC (n) PSO-TVAC (o) PSO-ICSA

(p) APSO-ZZLC (q) UAPSO-A (r) GPSO

Fig. 11: Percentage of particles with a bound violation in at least one dimension.

Table 3: Control parameter values employed by the SAPSO algorithms.

Algorithm	Parameters
PSO	$\omega = 0.729844, c_1 = c_2 = 1.496180$
PSO-TVIW	$\omega_s = 0.9, \omega_f = 0.4, c_1 = c_2 = 1.496180$
PSO-AIWF	$\omega_{min} = 0.2, \omega_{max} = 1.2, c_1 = c_2 = 2.0$
DAPSO	$\alpha = 1.0, \beta = 0.1, \omega_s = 1.0, c_1 = c_2 = 1.496180$
IPSO-LT	$\alpha = 0.5, \beta = 0.5, c_1 = c_2 = 2.0$
SAPSO-LFZ	$c_1 = c_2 = 1.496180$
SAPSO-DWCY	$\alpha = 3, \beta = 200, \gamma = 8, c_1 = c_2 = 2.0$
PSO-RBI	$\omega_{min} = 0.4, \omega_{max} = 0.9, c_1 = c_2 = 1.496180$
IPSO-CLL	$c_1 = c_2 = 2.0$
AIWPSO	$\omega_{min} = 0.0, \omega_{max} = 1.0, c_1 = c_2 = 1.496180$
APSO-VI	$\omega_{min} = 0.3, \omega_{max} = 0.9, \Delta\omega = 0.1, c_1 = c_2 = 1.496180$
SRPSO	$\Delta\omega = 0.00011, \eta = 1, \lambda = 0.5, \omega_s = 1.05, \omega_f = 0.5, c_1 = c_2 = 1.49445$
PSO-SAIC	$\omega_a = 0.9, \omega_b = 0.45, c_{2a} = 0.5, c_{2b} = 2.5, c_1 = 2.05$
PSO-RAC	–
PSO-TVAC	$\omega_s = 0.9, \omega_f = 0.4, c_{1s} = 2.5, c_{1f} = 0.5, c_{2s} = 0.5, c_{2f} = 2.5$
PSO-ICSA	$\omega_a = 0.9, \omega_b = 0.45, c_{1s} = 2.5, c_{1f} = 0.5, c_{2a} = 0.5, c_{2b} = 2.5$
APSO-ZZLC	–
UAPSO-A	$n_\omega = 20, n_c = 10, \omega_{min} = 0, \omega_{max} = 1, C_{min} = 0, C_{max} = 2, \tau = 0.5, a = b = 0.01$
GPSO	$\omega_{min} = 0.4, \omega_{max} = 0.9, C_{min} = 1.5, C_{max} = C_{final} = 2.5, \xi = 1.0$

Section 4.6, simply maintaining particle movement sizes below Δ_{max} does not guarantee that good search behaviour will be exhibited. This is further supported by the findings of Cleghorn and Engelbrecht (2014b), which indicated that parameters which lie near the boundaries of the region defined by Eq. (3) may exhibit large particle movement values coupled with unreasonably slow convergence despite being classified as convergent.

5 Conclusions

The primary purpose of this study was to provide a critical analysis of the current state of self-adaptive particle swarm optimization algorithms with a focus on whether these algorithms will exhibit convergent behaviour. The analysis was performed from both a theoretical and empirical standpoint and encompassed 18 algorithms. Firstly, the theoretical aspect was focused on determining the algorithmic conditions necessary for convergent behaviour to be exhibited according to the best-known convergence criterion. The empirical aspect employed a specially-formulated benchmark designed to isolate convergence behaviour as a means to examine whether the theoretical results held in practice. Furthermore, the empirical experiments examined the ability of the adaptation mechanisms to generate convergent parameter configurations, the ability to continually modify the values of their control parameters, and the ability to retain particles within the feasible region. All of the examined algorithms employed the parameters suggested by their respective authors and, therefore, no attempt was made to tune the parameters.

Table 4: Average measure values after 5000 iterations. Δ = average particle movement, CP = convergent parameters, Δ_p = average parameter movement, IP = invalid particles.

Algorithm	Δ	CP	Δ_p	IP
PSO	415.125	100%	0.0	70.7%
PSO-TVIW	56.489	100%	1.00e-4	9.6%
PSO-AIWF	2000.000	0%	0.0	96.7%
DAPSO	2000.000	0%	NaN	96.9%
IPSO-LT	2000.000	0%	0.0	96.7%
SAPSO-LFZ	2000.000	47.2%	0.0	53.5%
SAPSO-DWCY	1324.322	100%	0.0	96.2%
PSO-RBI	2000.000	76.7%	6.01e-2	41.5%
IPSO-CLL	2000.000	100%	0.0	100%
AIWPSO	45.521	100%	0.0	3.3%
APSO-VI	55.940	100%	0.0	6.1%
SRPSO	2000.000	96.7%	0.0	3.3%
PSO-SAIC	2000.000	0%	NaN	96.7%
PSO-RAC	165.544	100%	1.60e+0	44.2%
PSO-TVAC	32.354	100%	5.74e-4	6.5%
PSO-ICSA	2000.000	0%	4.00e-4	96.7%
APSO-ZZLC	1318.307	100%	4.51e-5	96.1%
UAPSO-A	124.467	70%	8.47e-1	38.1%
GPSO	2000.000	16.7%	8.35e-2	96.7%

The results depicted were rather underwhelming and depicted a grim state for the study of SAPSO algorithms. A majority of the examined algorithms demonstrated immediate divergence, whereby nearly all particles exited the feasible region and never returned. Of the non-divergent algorithms, a significant portion exhibited premature convergence. Similarly, a significant portion of the algorithms were incapable of even generating parameter configurations which adhered to the convergence criterion, thereby causing their search efforts to be wasted on infeasible solutions. However, there were a few self-adaptive algorithms which demonstrated exemplary search behaviour, namely the APSO-VI and UAPSO-A algorithms. Additionally, the purely random parameter selection (PSO-RAC) and the linearly-varying algorithms (PSO-TVIW and PSO-TVAC) depicted relatively good search behaviour. Therefore, it can be concluded that a vast majority of the SAPSO algorithms examined have fundamental flaws which hinder their ability to perform an effective search.

Given that this study only examined the search behaviour on a specially-formulated benchmark, an immediate avenue of future work is to analyse the performance on a wide variety of benchmark functions. However, given that many of the algorithms fail to converge on a benchmark specifically designed to examine convergence, performance is expected to be underwhelming for a majority of the algorithms. Furthermore, this study made no attempt to tune the parameters of the algorithms. It is entirely possible that the convergence behaviour of the examined algorithms will change when the parameters are appropriately tuned, thus future work will examine the sensitivity of the examined algorithms to their respective control parameters. Finally, further re-

search should investigate whether the values of the control parameters adopted at various phases of the search are well-suited for the current environment.

Acknowledgement

This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 46712). The opinions, findings and conclusions or recommendations expressed in this article is that of the author(s) alone, and not that of the NRF. The NRF accepts no liability whatsoever in this regard. This work was also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

References

van den Bergh F, Engelbrecht AP (2006) A study of particle swarm optimization particle trajectories. Information Sciences 176(8):937–971, DOI 10.1016/j.ins.2005.02.003

Birattari M, Stützle T, Paquete L, Varrentrapp K (2002) A racing algorithm for configuring metaheuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, pp 11–18, DOI 10.1.1.20.1464

Carlisle A, Dozier G (2001) An Off-The-Shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, Purdue School of Engineering and Technology, pp 1–6

Changhe Li, Shengxiang Yang, Trung Thanh Nguyen (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, DOI 10.1109/TSMCB.2011.2171946

Chen Hh, Li Gq, Liao Hl (2009) A self-adaptive improved particle swarm optimization algorithm and its application in available transfer capability calculation. In: 2009 Fifth International Conference on Natural Computation, IEEE, vol 3, pp 200–205, DOI 10.1109/ICNC.2009.214

Cleghorn CW, Engelbrecht A (2016) Particle swarm optimizer: The impact of unstable particles on performance. In: 2016 IEEE Symposium Series on Computational Intelligence, IEEE, pp 1–7, DOI 10.1109/SSCI.2016.7850265

Cleghorn CW, Engelbrecht AP (2014a) A generalized theoretical deterministic particle swarm model. Swarm Intelligence 8(1):35–59, DOI 10.1007/s11721-013-0090-y

Cleghorn CW, Engelbrecht AP (2014b) Particle swarm convergence: an empirical investigation. In: 2014 IEEE Congress on Evolutionary Computation, IEEE, pp 2524–2530, DOI 10.1109/CEC.2014.6900439

Cleghorn CW, Engelbrecht AP (2015) Particle swarm variants: standardized convergence analysis. Swarm Intelligence 9(2-3):177–203, DOI 10.1007/s11721-015-0109-7

Cleghorn CW, Engelbrecht AP (2017) Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. Swarm Intelligence pp 1–22, DOI 10.1007/s11721-017-0141-x

Dong C, Wang G, Chen Z, Yu Z (2008) A method of self-adaptive inertia weight for PSO. In: 2008 International Conference on Computer Science and Software Engineering, IEEE, vol 1, pp 1195–1198, DOI 10.1109/CSSE.2008.295

Engelbrecht A (2012) Particle swarm optimization: velocity initialization. In: 2012 IEEE Congress on Evolutionary Computation, IEEE, pp 1–8, DOI 10.1109/CEC.2012.6256112

Engelbrecht A (2013a) Particle swarm optimization: global best or local best? In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, IEEE, pp 124–135, DOI 10.1109/BRICS-CCI-CBIC.2013.31

Engelbrecht A (2013b) Roaming behavior of unconstrained particles. In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, IEEE, pp 104–111, DOI 10.1109/BRICS-CCI-CBIC.2013.28

Gazi V (2012) Stochastic stability analysis of the particle dynamics in the PSO algorithm. In: 2012 IEEE International Symposium on Intelligent Control, IEEE, pp 708–713, DOI 10.1109/ISIC.2012.6398264

Harrison KR, Engelbrecht AP, Ombuki-Berman BM (2016a) Inertia weight control strategies for particle swarm optimization. Swarm Intelligence 10(4):267–305, DOI 10.1007/s11721-016-0128-z

Harrison KR, Engelbrecht AP, Ombuki-Berman BM (2016b) The sad state of self-adaptive particle swarm optimizers. In: 2016 IEEE Congress on Evolutionary Computation, IEEE, pp 431–439, DOI 10.1109/CEC.2016.7743826

Harrison KR, Ombuki-Berman BM, Engelbrecht AP (2017) Optimal parameter regions for particle swarm optimization algorithms. In: 2017 IEEE Congress on Evolutionary Computation, IEEE, pp 349–356, DOI 10.1109/CEC.2017.7969333

Hashemi A, Meybodi M (2011) A note on the learning automata based algorithms for adaptive parameter selection in PSO. Applied Soft Computing 11(1):689–705, DOI 10.1016/j.asoc.2009.12.030

Hashemi AB, Meybodi MR (2009) Adaptive parameter selection scheme for PSO: A learning automata approach. In: 2009 14th International CSI Computer Conference, IEEE, pp 403–411, DOI 10.1109/CSICC.2009.5349614

Ju-Long D (1982) Control problems of grey systems. Systems & Control Letters 1(5):288–294, DOI 10.1016/S0167-6911(82)80025-X

Jun S, Jian L (2009) An improved self-adaptive particle swarm optimization algorithm with simulated annealing. In: 2009 Third International Symposium on Intelligent Information Technology Application, IEEE, vol 3, pp 396–399, DOI 10.1109/IITA.2009.476

Kadirkamanathan V, Selvarajah K, Fleming P (2006) Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Transactions on Evolutionary Computation 10(3):245–255, DOI 10.1109/TEVC.2005.857077

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, vol 4, pp 1942–1948, DOI 10.1109/ICNN.1995.488968

Leonard BJ, Engelbrecht AP (2013) On the optimality of particle swarm parameters in dynamic environments. In: 2013 IEEE Congress on Evolutionary Computation, IEEE, pp 1564–1569, DOI 10.1109/CEC.2013.6557748

Leu MS, Yeh MF (2012) Grey particle swarm optimization. Applied Soft Computing 12(9):2985–2996, DOI 10.1016/j.asoc.2012.04.030

Li X, Fu H, Zhang C (2008) A self-adaptive particle swarm optimization algorithm. In: 2008 International Conference on Computer Science and Software Engineering, IEEE, vol 5, pp 186–189, DOI 10.1109/CSSE.2008.142

Li Z, Tan G (2008) A self-adaptive mutation-particle swarm optimization algorithm. In: 2008 Fourth International Conference on Natural Computation, IEEE, vol 1, pp 30–34, DOI 10.1109/ICNC.2008.633

Liu B, Wang L, Jin YH, Tang F, Huang DX (2005) Improved particle swarm optimization combined with chaos. Chaos, Solitons & Fractals 25(5):1261–1271, DOI 10.1016/j.chaos.2004.11.095

Liu Q (2015) Order-2 stability analysis of particle swarm optimization. Evolutionary Computation 23(2):187–216, DOI 10.1162/EVCO_a_00129

Montes de Oca MA, Peña J, Stützle T, Pinciroli C, Dorigo M (2009) Heterogeneous particle swarm optimizers. In: 2009 IEEE Congress on Evolutionary Computation, IEEE, pp 698–705, DOI 10.1109/CEC.2009.4983013

Nepomuceno FV, Engelbrecht AP (2013) A self-adaptive heterogeneous pso for real-parameter optimization. In: 2013 IEEE Congress on Evolutionary Computation, IEEE, pp 361–368, DOI 10.1109/CEC.2013.6557592

Nickabadi A, Ebadzadeh MM, Safabakhsh R (2011) A novel particle swarm optimization algorithm with adaptive inertia weight. Applied Soft Computing 11(4):3658–3670, DOI 10.1016/j.asoc.2011.01.037

Panigrahi B, Ravikumar Pandi V, Das S (2008) Adaptive particle swarm optimization approach for static and dynamic economic load dispatch. Energy Conversion and Management 49(6):1407–1415, DOI 10.1016/j.enconman.2007.12.023

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, DOI 10.1109/TEVC.2008.2011744

Poli R, Broomhead D (2007) Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press, p 134, DOI 10.1145/1276958.1276977

Ratnaweera A, Halgamuge S, Watson H (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation 8(3):240–255, DOI 10.1109/TEVC.2004.826071

Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings., IEEE, pp 69–73, DOI 10.1109/ICEC.1998.699146

Shi Y, Eberhart R (1999) Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, IEEE, vol 3, pp 1945–1950, DOI 10.1109/CEC.1999.785511

Tanweer M, Suresh S, Sundararajan N (2015) Self regulating particle swarm optimization algorithm. Information Sciences 294:182–202, DOI 10.1016/j.ins.2014.09.053

Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6):317–325, DOI 10.1016/S0020-0190(02)00447-7

Wang Y, Li B, Weise T, Wang J, Yuan B, Tian Q (2011) Self-adaptive learning based particle swarm optimization. Information Sciences 181(20):4515–4538, DOI 10.1016/j.ins.2010.07.013

Wu Z, Zhou J (2007) A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. In: 2007 International Conference on Computational Intelligence and Security, IEEE, pp 133–136, DOI 10.1109/CIS.2007.95

Xu G (2013) An adaptive parameter tuning of particle swarm optimization algorithm. Applied Mathematics and Computation 219(9):4560–4569, DOI 10.1016/j.amc.2012.10.067

Xuanping Z, Du Yuping QG, Zheng Q (2005) Adaptive particle swarm algorithm with dynamically changing inertia weight. Journal of Xi'an Jiaotong University 10:1039–1042

Yang X, Yuan J, Yuan J, Mao H (2007) A modified particle swarm optimizer with dynamic adaptation. Applied Mathematics and Computation 189(2):1205–1213, DOI 10.1016/j.amc.2006.12.045

Yasuda K, Iwasaki N, Ueno G, Aiyoshi E (2008) Particle swarm optimization: a numerical stability analysis and parameter adjustment based on swarm activity. IEEJ Transactions on Electrical and Electronic Engineering 3(6):642–659, DOI 10.1002/tee.20326

Zhan ZH, Zhang J, Li Y, Chung HSH (2009) Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39(6):1362–1381, DOI 10.1109/TSMCB.2009.2015956

van Zyl E, Engelbrecht A (2014) Comparison of self-adaptive particle swarm optimizers. In: 2014 IEEE Symposium on Swarm Intelligence, IEEE, pp 1–9, DOI 10.1109/SIS.2014.7011775