An Analysis of Parameter Control Mechanisms for the Particle Swarm Optimization Algorithm

by

Kyle Robert Harrison

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

July 2018

An Analysis of Parameter Control Mechanisms for the Particle Swarm Optimization Algorithm

by

Kyle Robert Harrison

Abstract

The particle swarm optimization (PSO) algorithm is a stochastic, population-based optimization technique influenced by social dynamics. It has been shown that the performance of the PSO algorithm can be greatly improved if the control parameters are appropriately tuned. However, the tuning of control parameter values has traditionally been a time-consuming, empirical process followed by statistical analysis. Furthermore, ideal values for the control parameters may be time-dependent; parameter values that lead to good performance in an exploratory phase may not be ideal for an exploitative phase. Self-adaptive algorithms eliminate the need to tune parameters in advance, while also providing real-time behaviour adaptation based on the current problem.

This thesis first provides an in-depth review of existing self-adaptive particle swarm optimization (SAPSO) techniques. Their ability to attain order-2 stability is examined and it is shown that a majority of the existing SAPSO algorithms are guaranteed to exhibit either premature convergence or rapid divergence. A further investigation focusing on inertia weight control strategies demonstrates that none of the examined techniques outperform a static value. This thesis then investigates the performance of a wide variety of PSO parameter configurations, thereby discovering regions in parameter space that lead to good performance. This investigation provides strong empirical evidence that the best values to employ for the PSO control parameters change over time. Finally, this thesis proposes novel PSO variants inspired by results of the aforementioned studies.

Supervisors	**:**	Prof. A. P. Engelbrecht
		Prof. B. M. Ombuki-Berman
Department	**:**	Department of Computer Science
Degree	**:**	Philosophiae Doctor

“The measure of intelligence is the ability to change.”

Albert Einstein

“Life is neither static nor unchanging. With no individuality, there can be no change, no adaptation and, in an inherently changing world, any species unable to adapt is also doomed.”

Jean M. Auel

Acknowledgements

I would like to acknowledge the following people, without whom this thesis would not have been possible:

- My supervisors, Prof. Andries Engelbrecht and Prof. Beatrice Ombuki-Berman, who both provided invaluable support and guidance throughout this research.
- My family for their constant love and support as I pursue my academic dreams.
- My colleagues and friends of CIRG for their valuable feedback and assistance.

Additionally, I would like to acknowledge the following organizations for their financial support of this research:

- The University of Pretoria.
- The Natural Sciences and Engineering Research Council of Canada (NSERC).
- The Council for Scientific and Industrial Research of South Africa (CSIR).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of computational intelligence (CI) is premised on the study of techniques to facilitate intelligent behaviour in complex environments. Within the field of CI, a large body of research examines the study of social organisms, whereby the collective intelligence is leveraged in some capacity. This sub-field of CI is referred to as swarm intelligence (SI). As a prominent example of SI, the particle swarm optimization (PSO) algorithm [59] is a population-based, stochastic search technique inspired by the flocking behaviour of birds.

It is well known that an effective search technique must strike a balance between exploring new regions in the search space and exploiting known, promising regions. In the context of CI algorithms, exploration refers to the identification of new, previously undiscovered areas in the search space while exploitation refers to refinement of previously found, promising solutions. In the PSO algorithm, exploration and exploitation can be controlled by the values of the control parameters [4, 9, 10, 63, 105, 107]. Moreover, the performance of the PSO algorithm can be improved by appropriately tuning the values of the control parameters to the current problem [12, 52, 71, 106], given that the particle movement patterns are heavily influenced by these control parameters [9, 106]. The searching capability of the algorithm, and by extension the exploration/exploitation balance, is thus directly influenced by the values of the three main control parameters, namely the inertia weight (ω), the cognitive acceleration coefficient (c_1), and the social acceleration coefficient (c_2). Additionally, the PSO algorithm has been shown to be rather sensitive to the values of these control parameters [12, 105, 107] and thus *a priori* tuning of the control parameter values may lead to improved performance. However, the tuning of control parameter values has traditionally been a time-consuming, empirical process followed by an arduous statistical analysis. To alleviate the issue of *a priori* parameter tuning, this thesis investigates self-adaptive particle swarm optimization (SAPSO) techniques that do not rely on *a priori* specification of values for the conventional PSO control parameters.

The remainder of this chapter is organized as follows. Section 1.1 provides further motivation for the investigation of SAPSO strategies. The primary objectives of this thesis are then provided in Section 1.2. Finally, Section 1.3 provides a detailed outline of the remainder of this thesis.

1.1 Motivation

There has been a number of studies that have empirically examined the performance of various PSO parameter configurations [9, 17, 43, 52, 71, 106]. However, there is no general consensus as to which parameter configurations lead to the best performance. Most of the previous empirical studies examined only a limited set of parameter configurations over a small number of benchmark problems, and for only specific problem dimensionalities. The result of these studies was a set of recommended parametrizations for the PSO algorithm. In a more comprehensive study, Cleghorn and Engelbrecht [17] examined 1264 parameter configurations over 28 benchmark problems, and clearly identified regions of the parameter space that lead the PSO algorithm to perform worse than a random search. Each of these studies have led to an enhanced understanding of the general region in parameter space where good parameter configurations lie. However, none have answered an important question, namely whether the best parameters to employ are in fact time-dependent. Similarly, many studies implicitly assumed that the cognitive and social control parameters should have equal values. Thus, an additional equally important question remains unanswered: in what regions of parameter space do the best parameters reside when the values of these two parameters are not equal? This thesis investigates both these questions.

While there is no doubt that parameter tuning in the PSO algorithm is important, especially when more complex optimization problems are considered, the task of effectively tuning the parameters is computationally expensive. Parameter tuning is often an arduous manual process, whereby a large number of candidate parametrizations must be examined and analysed. While there have been various automated parameter configuration tools proposed [5, 49, 50, 87], such tools have two clear drawbacks. Firstly, automated methods simply automate the process of parameter tuning and do not necessarily reduce the time complexity of the parameter search. Although various optimizations can be made, such as removing a particular parameter configuration from consideration if enough evidence is gathered to deduce that the configuration is poorly performing, the original argument still stands in that this does not necessarily reduce the overall complexity of the control parameter tuning problem. Rather, it simply translates a manual process into an automated process. However, it is worth mentioning that the *a priori* parameter tuning mechanisms can, in some instances, produce well-performing, robust parameter configurations that are reusable across a set of similar problems. This is especially true if the parameter configurations are tuned using a large set of problems.

Secondly, there is an implicit, often-overlooked assumption in *a priori* parameter tuning that the optimal parameter configuration does not change over time. With an automated parameter tuning strategy (and, for that matter, manual tuning techniques), the tuned parameters will be statically used throughout the course of the search. This is likely not an optimal scenario given that there exists a well-established ideology that the best parameter values change over time. For instance, the linearly-decreasing inertia weight PSO by Shi and Eberhart [98, 99] was premised on the idea of reducing the value of the inertia weight over time. Leonard and Engelbrecht [63] empirically found that parameters well-suited for exploration were not well-suited for exploitation, and *vice versa*. Moreover, heterogeneous PSO algorithms have evidenced that the most suitable velocity update scheme to employ varies during the search [65, 79, 81, 110]. To address the time sensitivity of control parameter values, this thesis investigates the performance of various PSO parameters at different points throughout the search process. Furthermore, this thesis examines whether the short-term performance of PSO parameter configurations is indicative of their long-term performance.

To alleviate the issues associated with *a priori* parameter tuning, various SAPSO algorithms that adapt their control parameters throughout execution have been proposed. SAPSO algorithms typically make use of introspective observation to refine the values of the control parameters based on their current and/or past performance. Despite the well-known result of the No Free Lunch theorem stating that no optimization algorithm will outperform any other optimization algorithm, including random search, across all optimization problems [111], SAPSO algorithms are nonetheless an attempt at providing superior performance on limited subsets of such problems.

While there are a large number of SAPSO algorithms that have been proposed in the literature, their behaviour is still not well understood. Specifically, it is unknown whether these algorithms will even exhibit stable behaviour. Stability, in this context, refers to the attainment of order-1 and order-2 stability, i.e., an equilibrium state such that the mean and variance of the particle step sizes tends to a constant value (see Section 2.3). An algorithm designed to adapt its control parameters can be reasonably expected to prevent unstable behaviour given that parameters leading to instability should be avoided by the adaptation mechanism. However, as previous works have identified [41, 42, 109], this is not always the case. Recently, evidence has been provided to suggest that parameter configurations that adhere to a well-known stability criterion will generally lead to better performance than parameter configurations that violate the criterion [17, 43]. Furthermore, it has been shown that many of the parameter configurations that violate the stability criterion lead to worse performance than random search [17]. To this end, this thesis investigates the stability behaviour of a wide variety of SAPSO strategies. Finally, this thesis proposes three novel PSO techniques that do not rely on the conventional PSO control parameters.

1.2 Objectives

The primary objectives of this thesis are summarized as follows:

- provide a thorough review of existing SAPSO techniques.
- investigate the behaviour of existing SAPSO techniques, specifically with regards to theoretical stability criteria.

- investigate the parametrization of PSO to identify regions of parameter space that lead to good performance.
- provide further evidence in support of SAPSO strategies.
- propose novel PSO techniques that do not rely on *a priori* values for the conventional PSO parameters.

1.3 Thesis Outline

The section provides a brief outline of the remainder of this thesis.

- **Chapter 2** provides background information on the PSO algorithm that is necessary for the remainder of this thesis.
- **Chapter 3** provides an extensive review of existing SAPSO techniques.
- **Chapter 4** examines the ability of existing SAPSO techniques to attain order-2 stability. Conditions for stability to be exhibited are derived analytically. Empirical behaviour is also examined in support of the analytical findings.
- **Chapter 5** provides an extensive analysis of inertia weight control strategies. A detailed empirical investigation is carried out, ultimately concluding that none of the examined strategies outperform a constant inertia weight.
- **Chapter 6** presents an empirical investigation of 1012 PSO parameter configurations. The focus of this chapter is on the identification of regions in parameter space that lead to good performance.
- **Chapter 7** extends the study of Chapter 6 to include imbalanced configurations and, furthermore, addresses the question of whether the best values for PSO parameters are time-dependent.
- **Chapter 8** proposes a novel SAPSO technique built upon the results from Chapters 6 and 7.

- **Chapter 9** proposes a novel PSO variant that employs machine learning models, which are trained using the data from Chapter 7, to predict the success of parameter configurations for PSO.
- **Chapter 10** proposes a probabilistic PSO technique that has no dependency on the conventional PSO control parameters.
- **Chapter 11** provides concluding remarks and presents avenues of future work.

In addition, the following appendices are provided.

- **Appendix A** describes the suite of benchmark problems.
- **Appendix B** provides a listing of the acronyms defined in this thesis.
- **Appendix C** provides a listing of symbols used throughout this thesis.
- **Appendix D** lists the publications that were derived from this thesis.

Chapter 2

Particle Swarm Optimization

This chapter discusses the standard PSO algorithm. Section 2.1 provides a detailed description of the PSO algorithm. Section 2.2 discusses the PSO neighbourhood topology. Section 2.3 presents a brief overview of theoretical stability results for the PSO algorithm, while Section 2.4 discusses velocity clamping. Finally, a summary of this chapter is given in Section 2.5.

2.1 The Particle Swarm Optimization Algorithm

The PSO algorithm, developed by Kennedy and Eberhart [59], was inspired by a simulation of the complex flight patterns of a flock of birds. Their initial simulations evolved into a simple optimization algorithm that exhibits complex behaviour. The PSO algorithm consists of a collection of agents, referred to as particles, where each particle represents a candidate solution to the current optimization problem. Each particle retains three pieces of information, namely its current position, current velocity, and the best position it has found within the search space. Movement of particles is then governed by the iterative calculation of a velocity vector. The calculation of each particle's velocity is based on its 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. A particle also has a tendency to retain on its current trajectory via an inertia component. The neighbourhood of a particle refers to the other particles within the swarm from which it may take influence. For example, the original PSO algorithm employed one of two neighbourhood strategies, either a star topology where the neighbourhood was the entire swarm, or a ring topology where the neighbourhood consisted of the immediate neighbours, determined by particle index, when the particles were arranged in a ring [60].

The velocity is then calculated for dimension j of particle i at time t as

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2ij}(t)(\hat{y}_{ij}(t) - x_{ij}(t)) \tag{2.1}$$

and particle positions are updated according to

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{2.2}$$

The calculation of the velocity vector can be dissected into three distinct components as follows:

- **The inertia component**, $\omega \mathbf{v}_i(t)$, which constitutes a particle's tendency to remain on its current trajectory by applying a portion of the previous velocity to the current velocity. The portion of the previous velocity applied is known as the inertia weight, ω, and is typically a positive value within the range of $[0, 1]$.

- **The cognitive component**, $c_1 \mathbf{r}_{1i}(t) \otimes (\mathbf{y}_i(t) - \mathbf{x}_i(t))$, which governs the particle's self-influence. The $\otimes$ operator is used to indicate component-wise multiplication of two vectors. In the cognitive component, the difference vector between the particle's best found position, $\mathbf{y}_i(t)$, and its current position is first calculated. This difference vector is then component-wise multiplied with a vector of uniform random components, where each $r_{1ij}(t) \sim U(0, 1)$, and the cognitive acceleration coefficient c_1. Typically, the cognitive acceleration coefficient is within the range $[0, 2]$.

- **The social component**, $c_2 \mathbf{r}_{2ij}(t) \otimes (\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t))$, which dictates the degree to which the neighbourhood best influences a particle's movement direction. As with the cognitive component, the operator $\otimes$ is used to indicate component-wise multiplication of two vectors. The social component is calculated as the difference between a particle's position and its neighbourhood best position, $\hat{\mathbf{y}}_i(t)$, component-wise multiplied by a vector with uniform random components, where each $r_{2ij}(t) \sim U(0,1)$, and the social acceleration coefficient c_2. Typically, the social acceleration coefficient is within the range $[0,2]$.

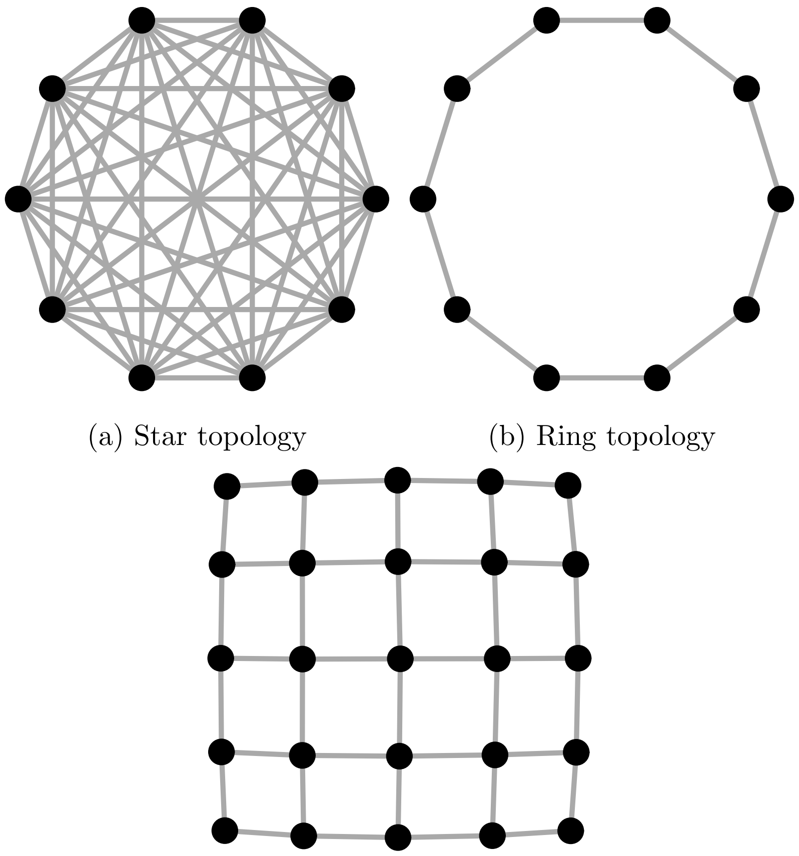
The formulation of PSO using the velocity calculation given in Equation (2.1) is known as the inertia weight model of Shi and Eberhart [98], and is hereafter referred to as the standard PSO. Alternative formulations, which use different velocity update equations, have also been proposed. The most notable alternative formulation is the constriction factor PSO proposed by Clerc [22]. However, it can easily be shown that the velocity update mechanism of the inertia weight and constriction factor variants are equivalent [2]. Another noteworthy alternative is known as Standard PSO 2011, commonly referred to as SPSO-2011, which is promoted as a standardized variant of PSO that includes rotational invariance and an adaptive random neighbourhood topology [118].

2.2 Neighbourhood Topologies

The strength of the PSO algorithm lies in the social communication, which is realized via the neighbourhood topology. The neighbourhood of a particle refers to the other particles within the swarm where a direct avenue of communication exists. It is well known that the best topology to employ is dependent upon both the optimization problem and computational budget [31, 60, 72].

A common technique for visualizing neighbourhood topologies is through the use of graphs, whereby nodes represent particles and edges depict the avenues of communication. Thus, the neighbourhood of a particle i can be more formally defined as the sub-graph induced by all particles adjacent to i within the social graph. While there are many potential neighbourhood topologies [60, 72, 74], each with their own advantages and disadvantages, most PSO literature employs one of three common topologies:

- **Star**: each particle is connected to every other particle via a fully-connected network. The star topology is visualized in Figure 2.1a. A PSO algorithm employing the star topology is commonly referred to as a global-best (gbest) PSO.



(a) Star topology (b) Ring topology

(c) 2D von Neumann topology

Figure 2.1: Common PSO neighbourhood topologies.

- **Ring**: each particle is connected to its immediate neighbours when arranged in a ring, as determined by particle index. The ring topology is visualized in Figure 2.1b. A PSO algorithm employing the ring topology is commonly referred to as a local-best (lbest) PSO. Note that the ring topology can be generalized to incorporate more than just the immediate neighbour on either side.

- **von Neumann**: each particle is connected to its neighbours when arranged in a grid-like lattice. The 2D von Neumann topology is visualized in Figure 2.1c.

2.3 Theoretical Particle Stability

There has been a significant amount of research effort devoted to the theoretical study of PSO stability [7, 18, 39, 56, 89, 90, 105, 107]. Specifically, many researchers have investigated the relationship between control parameter values and particle stability in the PSO algorithm. Note that stability does not necessary imply that the swarm has converged to a single point. Rather, stability is defined to be order-1 and order-2 stability of particle positions, as given in Definitions 2.3.1 and 2.3.2, respectively [89]. In simpler terms, order-1 and order-2 stability, when considered together, indicate that the sequence of particle positions has a fixed expected value and variance. Thus, a particle that is exhibiting both order-1 and order-2 stability is often referred to as convergent. However, it should be explicitly noted that there is no guarantee of deterministic convergence, such that the particle positions cease to move, unless order-2 stability occurs along with the personal and neighbourhood best positions being equal for all particles. This behaviour is infrequently observed in a stochastic context and there is no guarantee this condition will even occur in practice [21]. Furthermore, even if deterministic convergence does occur, there is no guarantee the particles will convergence toward an optimum. Therefore, the theoretical work on PSO focuses on providing conditions for order-1 and order-2 stability rather than on guaranteeing the deterministic convergence of particle positions.

Definition 2.3.1 (Order-1 stability) *A sequence $\mathbf{s}_t$ over $\mathbb{R}^n$ is order-1 stable if there exists an $\mathbf{s}_e \in \mathbb{R}^n$ such that*

$$\lim_{t\to\infty} E[\mathbf{s}_t] = \mathbf{s}_e \tag{2.3}$$

where $E[\mathbf{s}_t]$ is the expected value of $\mathbf{s}_t$.

Definition 2.3.2 (Order-2 stability) *A sequence $\mathbf{s}_t$ over $\mathbb{R}^n$ is order-2 stable if there exists an $\mathbf{s}_v \in \mathbb{R}^n$ such that*

$$\lim_{t\to\infty} V[\mathbf{s}_t] = \mathbf{s}_v \tag{2.4}$$

where $V[\mathbf{s}_t]$ is the variance of $\mathbf{s}_t$.

Given the stochastic nature of the PSO algorithm, various assumptions have been made to assist in deriving the region that leads to stable behaviour. One such assumption is known as the stagnation assumption, which assumes that the personal and neighbourhood best positions will eventually stagnate. However, the various assumptions employed by these theoretical studies have proven to be limiting, and therefore many of these purely theoretical studies do not capture the true region [19].

Empirical simulations (without any simplifying assumptions) by Cleghorn and Engelbrecht [19] have shown that, of the various stability criteria, the criterion given by Poli and Broomhead [90] and Poli [89] for order-2 stability,

$$c_1 + c_2 < \frac{24(1-\omega^2)}{7-5\omega}, \tag{2.5}$$

is the most accurate in practice. The criteria for order-1 stability, originally derived by Trelea [105] and later derived by Poli and Broomhead [90] and Poli [89], are given by

$$0 < c_1 + c_2 < 4(1+\omega), \quad |\omega| < 1, \tag{2.6}$$

thereby limiting consideration to only positive values for $c_1 + c_2$. While Equation (2.5) was derived using the stagnation assumption, later studies by Bonyadi and Michalewicz [6] and Cleghorn and Engelbrecht [21] have shown that equivalent, more generalized, criteria for order-1 and order-2 stability can be derived using less restrictive assumptions. The region defined by Equation (2.5) is illustrated in Figure 2.2, where parameter values that lie within the parabolic region will lead to stable behaviour.

Further studies by Liu [71] and Cleghorn and Engelbrecht [20] have found that the region defined by Equation (2.5) is not dependent upon the neighbourhood topology. Moreover, recent studies have provided evidence that parameter configurations that adhere to the criterion of Equation (2.5) will generally lead to better performance than parameter configurations that violate the criterion [17, 43]. Specifically, it was shown that a majority of theoretically unstable parameter configurations caused the PSO algorithm to perform worse than random search, and that selecting theoretically stable parameters drastically increased the likelihood of PSO outperforming random search [17].

2.4 Velocity Clamping

It is well known that haphazard selection of parameter values may lead to divergent behaviour in the PSO algorithm due to excessively large particle movements. To address

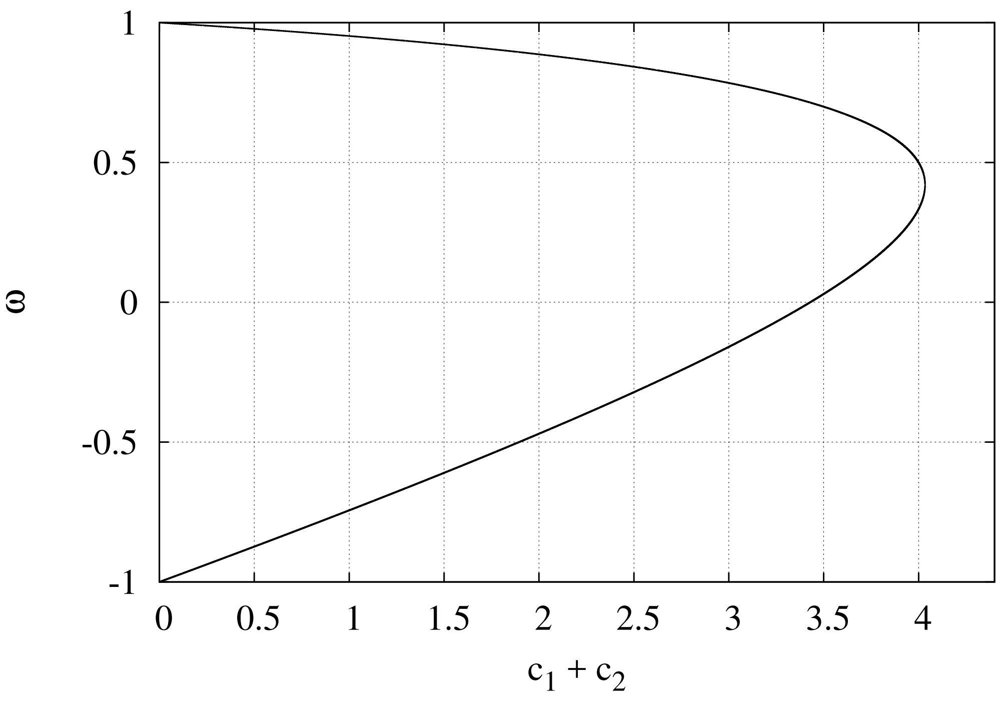


Figure 2.2: Visualization of Poli's theoretically defined region for stable PSO parameters.

this issue, velocity clamping limits the step sizes of particles by providing a limit on the component sizes of the velocity vector. Particles are thus provided with an upper limit on the movement along each dimension, thereby preventing overly large step sizes. To implement velocity clamping, Equation (2.1) is amended to include

$$v_{ij}(t+1) = \begin{cases} -v_{\max,j} & \text{if } v_{ij}(t+1) < -v_{\max,j} \\ v_{\max,j} & \text{if } v_{ij}(t+1) > v_{\max,j} \\ v_{ij}(t+1) & \text{otherwise} \end{cases} \tag{2.7}$$

where $v_{\max,j}$ is the largest allowable step size in dimension j. Commonly, $v_{\max,j}$ is set based on the size of the search space,

$$v_{\max,j} = \frac{x_{\max,j} - x_{\min,j}}{p},$$

where $x_{\min,j}$ and $x_{\max,j}$ are the bounds of the search space in dimension j, and p is an integer controlling the proportion of the search space used to bound the velocity.

While velocity clamping does not necessarily prevent divergent trajectories from occurring, it does prevent particles from having overly large step sizes, thereby delaying divergence. Additionally, velocity clamping may hinder the roaming behaviour of particles, a phenomena known to be beneficial to the overall search capabilities of a particle swarm optimizer [32]. Most importantly, the need for velocity clamping provides a clear indication that the control parameter values were poorly chosen. Therefore, if a PSO algorithm with an adaptive parameter strategy requires the use of velocity clamping to prevent divergent behaviour, the control strategy is clearly ineffective at controlling the search.

2.5 Summary

This chapter presented the standard PSO algorithm. Additionally, a discussion of neighbourhood topologies, theoretical stability analysis, and velocity clamping were also provided. The next chapter provides an extensive review of SAPSO variants that adapt the values of their control parameters over time.

Chapter 3

Self-Adaptive Particle Swarm Optimizers

SAPSO algorithms directly address the static control parameter problem by providing mechanisms whereby the values of the control parameters are refined throughout the search process. Note that the focus of this thesis is on self-adaptive mechanisms that tune the three primary control parameters of the PSO algorithm, namely the inertia weight, cognitive acceleration coefficient, and social acceleration coefficient. Thus, techniques that, for example, adapt the neighbourhood topology [68, 69, 95, 118] or swarm size [24, 68, 84], are outside the scope of this thesis. Similarly, it should be explicitly noted that this study focuses solely on the adaptation of control parameter values and doesn't examine other possible adaptations, such as mutations on particle positions or direct altering of velocity components. Finally, it is recognized that a large body of research exists for parameter control in other computational intelligence algorithms [3, 57, 78, 102], but that such approaches are also outside the scope of this thesis.

Self-adaptive parameter control mechanisms can be classified into two main categories, namely those that make use of introspective observation about the search behaviour to refine the values of the parameters over time, and those that do not make use of any information about the search process. For the latter category, the parameter control mechanisms generally adapt the control parameter values based solely on the number of iterations that have passed. This chapter reviews a number of SAPSO algorithms from both categories. Section 3.1 presents the time-dependent approaches, while Section 3.2 presents the adaptive variants. Finally, Section 3.3 provides a summary of the discussed SAPSO algorithms.

3.1 Non-Adaptive and Time-Varying Parameter Control Strategies

The first category of parameter control strategies are those that make no attempt to intelligently select control parameter values based on the characteristics of the current search. Thus, these strategies can not be considered self-adaptive in the proper sense. Nonetheless, such parameter control strategies can encourage varied exploration and exploitation during the search, which can potentially lead to enhanced search behaviour. Sections 3.1.1 to 3.1.12 discuss a number of non-adaptive parameter control strategies.

3.1.1 Particle Swarm Optimization with Linearly-Decreasing Inertia Weight

The particle swarm optimization with linearly-decreasing inertia weight (PSO-LDIW) algorithm [98, 99] was proposed as a method to linearly decrease the inertia weight over time. This inertia weight control strategy is based on the general consensus that exploration is favoured early in the search process, while exploitation is favoured later. However, it should be noted that recent evidence suggests the opposite is true for high-dimensional search spaces, which should rather focus solely on exploitation [108]. The inertia weight is calculated at each iteration according to

$$\omega(t) = \omega_s + (\omega_f - \omega_s)\frac{t}{T}, \tag{3.1}$$

where ω_s and ω_f are the initial and final inertia weight values, respectively, and T is the maximum number of iterations. The social and cognitive acceleration coefficients remain static throughout the search process. Thus, the PSO-LDIW algorithm tunes the value of the ω parameter at the expense of adding two new control parameters, namely ω_s and ω_f.

A noteworthy observation regarding PSO-LDIW is that convergence toward a global optimum is slowed later in the search as the algorithm lacks global search ability due to the reduction in inertia weight, which causes difficulty escaping local optima [99].

3.1.2 Particle Swarm Optimization with Random Inertia Weight

The particle swarm optimization with random inertia weight (PSO-RIW) algorithm [29] randomly selects the inertia weight at each iteration according to

$$\omega(t) = 0.5 + \frac{r(t)}{2}, \tag{3.2}$$

where $r(t) \sim U(0, 1)$. Effectively, this strategy samples the inertia weight, at each iteration, according to $\omega(t) \sim U(0.5, 1)$. Thus, the PSO-RIW algorithm tunes the ω parameter and does not introduce any new parameters. However, the values of 0.5 and 2 used in Equation (3.2), effectively the range in which to sample the inertia weight values, could be considered two additional parameters.

3.1.3 Particle Swarm Optimization with Time-Varying Acceleration Coefficients

The particle swarm optimization with time-varying acceleration coefficients (PSO-TVAC) algorithm [94] linearly varies the cognitive and social acceleration coefficients over time. The cognitive coefficient is decreased over time, while the social coefficient is increased over time as a means to, in theory, provide a smooth transition from exploration to exploitation as the search progresses. The cognitive and social acceleration coefficients are linearly scaled each iteration according to

$$c_1(t) = c_{1s} + (c_{1f} - c_{1s})\frac{t}{T} \tag{3.3a}$$

and

$$c_2(t) = c_{2s} + (c_{2f} - c_{2s})\frac{t}{T}, \tag{3.3b}$$

where the subscripts s and f represent the initial and final values, respectively. PSO-TVAC also employs the linearly decreasing inertia weight of PSO-LDIW given by Equation (3.1). 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_s, ω_f, c_{1s}, c_{1f}, c_{2s}, and c_{2f}.

3.1.4 Particle Swarm Optimization with Natural Exponent Inertia Weight

The particle swarm optimization with natural exponent inertia weight (PSO-NEIW) algorithm [15] uses a decreasing inertia weight based on the exponential function according to

$$\omega(t) = \omega_{min} + (\omega_{max} - \omega_{min})e^{-\frac{10t}{T}}. \tag{3.4}$$

The PSO-NEIW algorithm thus tunes the ω parameter, but introduces two new parameters, namely ω_{min} and ω_{max}. Additionally, the value 10 used in Equation (3.4) may require additional tuning.

3.1.5 Particle Swarm Optimization with Sugeno Inertia Weight

The particle swarm optimization with Sugeno inertia weight (PSO-SIW) algorithm [62] makes use of a Sugeno function [101], which is method of fuzzy inference, to control the inertia weight over time. This strategy provides a monotonically decreasing inertia weight according to

$$\omega(t) = \frac{1 - \beta(t)}{1 + s\beta(t)}, \tag{3.5}$$

where $\beta(t) = \frac{t}{T}$ and $s > -1$ is a constant controlling the shape of a function that governs the value of the inertia weight over time. When $s < 0$, the inertia weight follows a convex curve, while $s > 0$ leads to a concave curve. When $s = 0$, the inertia weight linearly decreases. Note that the inertia weight value in the PSO-SIW algorithm always decreases from an initial value of 1 to a final value of 0. The PSO-SIW algorithm thus tunes the ω parameter, but also introduces a new parameter s.

3.1.6 Decreasing Inertia Weight Particle Swarm Optimization

The decreasing inertia weight particle swarm optimization (DW-PSO) algorithm [33] employs a time-dependent inertia weight strategy according to

$$\omega(t) = \left(\frac{2}{t}\right)^{0.3}, \tag{3.6}$$

such that the inertia weight decreases over time in a non-linear fashion. While the DW-PSO algorithm tunes the value of the ω control parameter without introducing any new parameters, the values 2 and 0.3 from Equation (3.6) may require tuning, and thus could be considered additional control parameters.

3.1.7 Chaotic Descending Inertia Weight Particle Swarm Optimization

The chaotic descending inertia weight particle swarm optimization (CDIW-PSO) algorithm [34] adopts the use of chaotic dynamics to adapt the inertia weight over time according to

$$\omega(t) = z(t)\omega_{min} + (\omega_{max} - \omega_{min})\frac{T-t}{T}, \tag{3.7}$$

where $z(t)$ is the value of the logistic map

$$z(t+1) = 4z(t)(1 - z(t)) \tag{3.8}$$

with $z(0) \sim U(0,1)$. The CDIW-PSO algorithm thus tunes the ω parameter at the expense of introducing two new parameters, namely ω_{min} and ω_{max}.

3.1.8 Non-Linear Improved Particle Swarm Optimization

The particle swarm optimization with non-linear improved inertia weight (PSO-NLI) algorithm [53] uses a non-linear decreasing inertia weight defined by

$$\omega(t) = \omega_c u^{-t}, \tag{3.9}$$

where $\omega_c \in [0, 1]$ and $u \in [1.0001, 1.005]$. Note that the PSO-NLI algorithm tunes the value of ω over time, but that an initial value, ω_c, must still be supplied along with a value for the parameter u.

3.1.9 Logarithm Decreasing Inertia Weight Particle Swarm Optimization

The logarithm decreasing particle swarm optimization (LD-PSO) algorithm [37] employs a logarithmically decreasing inertia weight according to

$$\omega(t) = \omega_{max} + (\omega_{min} - \omega_{max}) \log_{10}\left(a + \frac{10t}{T}\right), \tag{3.10}$$

where a is a user-supplied constant that controls the speed of convergence toward an optimum. The LD-PSO algorithm thus tunes the ω parameter, but introduces three additional parameters, ω_{min}, ω_{max}, and a. Additionally, the value 10 used in Equation (3.10) may require further tuning.

3.1.10 Particle Swarm Optimization with Oscillating Inertia Weight

The particle swarm optimization with oscillating inertia weight (PSO-OIW) algorithm [61] was proposed as an inertia weight control strategy that did not monotonically decrease the inertia weight, but rather provided an oscillating inertia weight during the search process. The PSO-OIW algorithm employs a sinusoidal inertia weight strategy controlled by

$$\omega(t) = \begin{cases} \frac{\omega_{min}+\omega_{max}}{2} + \frac{\omega_{max}-\omega_{min}}{2}\cos\left(\frac{2\pi t(4k+6)}{3T}\right) & \text{if } t < \frac{3T}{4} \\ \omega_{min} & \text{otherwise.} \end{cases} \tag{3.11}$$

The PSO-OIW algorithm tunes the ω parameter, but introduces three additional parameters, namely ω_{min}, ω_{max}, and k. Additionally, the constants 2, 4, 6, and 3 used in the cosine expression in Equation (3.11) may require additional tuning. Kentzoglanakis and Poole [61] suggested values of $k = 7, \omega_{min} = 0.3$, and $\omega_{max} = 0.9$, which allows Equation

(3.11) to be simplified to

$$\omega(t) = \begin{cases} 0.6 + 0.3\cos\left(\frac{17\pi t}{3750}\right) & \text{if } t < 3750 \\ 0.3 & \text{otherwise.} \end{cases} \tag{3.12}$$

Kentzoglanakis and Poole [61] claimed that the sinusoidal wave specified by Equation (3.11), and consequently Equation (3.12), should complete $k + \frac{3}{2}$ cycles within a single PSO execution. The oscillating inertia weight of PSO-OIW, given the aforementioned control parameter values, is visualized in Figure 3.1.

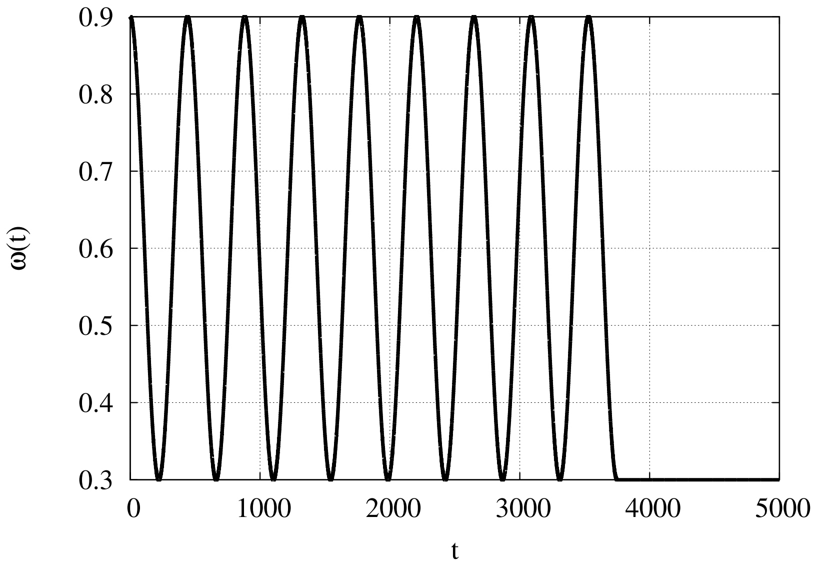


Figure 3.1: The oscillating inertia weight of Kentzoglanakis and Poole [61].

3.1.11 Particle Swarm Optimization with Non-Linear Inertia Coefficient

Yang *et al.* [115] posited that a non-linear, time-varying inertia weight would demonstrate superior performance over the linearly decreasing PSO-LDIW variant, and thus proposed the particle swarm optimization with non-linear inertia weight (PSO-NL) algorithm. To this end, the non-linear inertia weight at time t is given by

$$\omega(t) = \omega_{max} - (\omega_{max} - \omega_{min}) \left(\frac{t}{T}\right)^{\alpha}, \tag{3.13}$$

where α is a user supplied constant. The PSO-NL algorithm thus tunes the ω parameter, but introduces three new parameters, namely ω_{min}, ω_{max}, and α. The authors empirically suggested the use of $\alpha = 1/\pi^2$, $\omega_{min} = 0.4$, and $\omega_{max} = 0.9$. The resulting inertia weight over time is visualized in Figure 3.2.

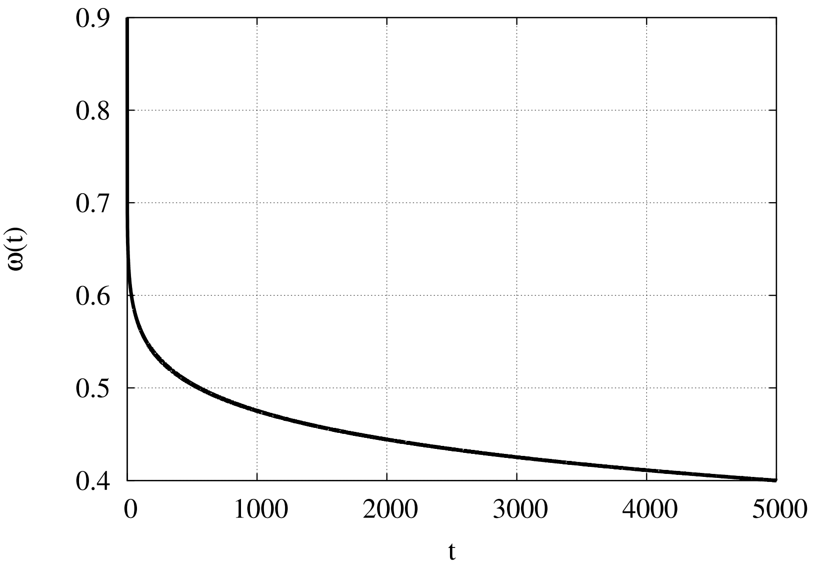


Figure 3.2: Visualization of the non-linear inertia weight strategy of PSO-NL.

3.1.12 Particle Swarm Optimization with Random Acceleration Coefficients

As a baseline used to compare against other SAPSO algorithms, the particle swarm optimization with random acceleration coefficients (PSO-RAC) algorithm is introduced as a technique that employs randomly-generated values for each of the control parameters. Specifically, a new set of values for each of the control parameters, which explicitly adhere to the stability criterion outlined in Equation (2.5), is (randomly) generated for each particle at every iteration. Thus, the PSO-RAC algorithm tunes each of the ω, c_1, and c_2 parameters, but requires no additional control parameters.

3.2 True Self-Adaptive Particle Swarm Optimizers

In contrast to the time-variant approaches, truly self-adaptive PSO variants use introspective observation to adapt the values of their control parameters. Refinement of control parameters is done at either the global (swarm) level or the local (particle) level based on the current state of the algorithm. Sections 3.2.1 to 3.2.17 discuss a number of such self-adaptive PSO algorithms.

3.2.1 Particle Swarm Optimization with Adaptive Inertia Weight Factor

The particle swarm optimization with adaptive inertia weight factor (PSO-AIWF) algorithm [70] adapts the inertia weight based on a particle's fitness relative to the average fitness of the swarm. Liu *et al.* [70] posited that particles with good fitness values should be 'protected' through the use of small inertia weight 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

$$\omega_i(t) = \begin{cases} \omega_{min} + \frac{(\omega_{max}-\omega_{min})(f_i(t)-f_{min}(t))}{\overline{f(t)}-f_{min}(t)} & \text{if } f_i(t) \leq \overline{f(t)} \\ \omega_{max} & \text{if } f_i(t) > \overline{f(t)}, \end{cases} \tag{3.14}$$

where $\overline{f(t)}$ and $f_{min}(t)$ are the average and minimum fitness values, determined using the current particle positions, 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}.

3.2.2 Dynamic Adaptation Particle Swarm Optimization

The dynamic adaptation particle swarm optimization (DAPSO) algorithm [116] 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.* [114], the evolutionary[1]

[1]PSO does not actually exhibit evolution. Nonetheless, the original terminology is used.

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

$$h_i(t) = \left| \frac{\min\{f(\mathbf{y}_i(t-1)), f(\mathbf{y}_i(t))\}}{\max\{f(\mathbf{y}_i(t-1)), f(\mathbf{y}_i(t))\}} \right|. \tag{3.15}$$

Note that $0 \leq h \leq 1$, and smaller values for h correspond to faster 'evolution' as they imply 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

$$s(t) = \left| \frac{\min\{f(\mathbf{y}^*(t)), f_{avg}(t)\}}{\max\{f(\mathbf{y}^*(t)), f_{avg}(t)\}} \right|, \tag{3.16}$$

where $\mathbf{y}_i^*(t)$[2] 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 then calculated as

$$\omega_i(t) = \omega_s - \alpha(1 - h_i(t)) + \beta s(t) \tag{3.17}$$

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

$$\forall t, 1 - \alpha \leq \omega_i(t) \leq 1 + \beta. \tag{3.18}$$

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.* [116] concluded that the parametrization of DAPSO, specifically the α and β parameters, did not have a significant impact on performance. Specifically, using a suite of benchmark problems, the authors provided empirical evidence that selecting α and β anywhere within $[0, 1]$ lead to good performance and that the performance of DAPSO did not strongly depend on these values [116]. Furthermore, it was observed that, on all six benchmark functions,

[2]Note that $\mathbf{y}^*(t)$ differs from $\mathbf{y}_i(t)$ in that $\mathbf{y}^*(t)$ is the best solution found during iteration t while $\mathbf{y}_i(t)$ is the best solution found during any iteration up to, and including, t.

the optimal solution was found 100% of the time, over 50 runs, using any of the 100 combinations of α and β examined [116]. Another noteworthy observation made by the authors was that the velocity of particles within the DAPSO algorithm did not approach zero as in a standard PSO, but rather the velocities tended towards the maximum allowable velocity (i.e., the velocity clamp) [116]. This observation suggests that the DAPSO algorithm does not exhibit order-2 stability.

3.2.3 Particle Swarm Optimization with Individual Coefficient Adjustment

The self-adaptive particle swarm optimization with individual coefficient adjustment (PSO-SAIC) algorithm [112] 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.

The related distance for particle i is first defined as

$$\xi_i(t) = \begin{cases} 0 & \text{if } f(\mathbf{x}_i(t-1)) = 0 \\ \frac{f(\mathbf{x}_i(t-1)) - f(\hat{\mathbf{y}}(t-1))}{f(\mathbf{x}_i(t-1))} & \text{otherwise,} \end{cases} \tag{3.19}$$

which quantifies the efficacy of particle i at time t. Clearly, $\xi_i(t) \approx 1$ when particle i is far from the global best position and, conversely, $\xi_i(t) \approx 0$ when particle i is near the global best. Note that the definition of related distance, as defined in Equation (3.19), presupposes a minimization problem with a positive global minimum.

The inertia weight is then adapted according to

$$\omega_i(t) = \omega_a F(\xi_i(t)) + \omega_b, \tag{3.20a}$$

where

$$F(\xi_i(t)) = 2\left(1 - \cos\left(\frac{\pi\xi_i(t)}{2}\right)\right), \tag{3.20b}$$

while the social acceleration coefficient is calculated as

$$c_{2i}(t) = c_{2a} G(\xi_i(t)) + c_{2b}, \tag{3.21a}$$

where

$$G(\xi_i(t)) = 2.5\left(1 - \cos\left(\frac{\pi\xi_i(t)}{2}\right)\right). \tag{3.21b}$$

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 Equations (3.20b) and (3.21b) may also be treated as control parameters. Figure 3.3 presents the adjustment values as calculated by Equations (3.20b) and (3.21b).

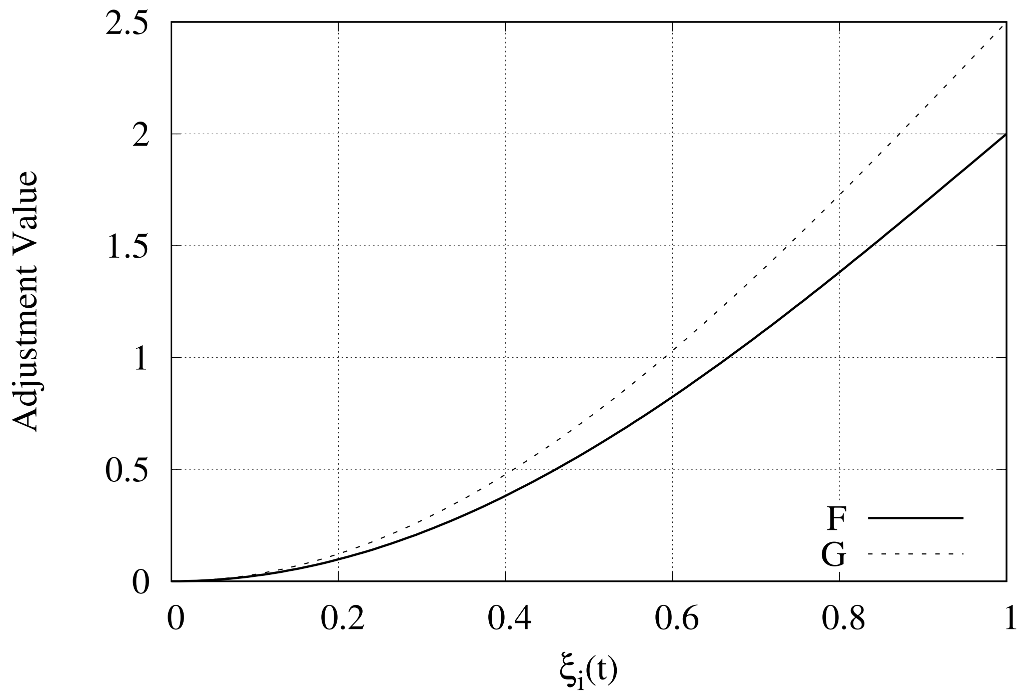


Figure 3.3: The adjustment values described by Equations (3.20b) and (3.21b) for the PSO-SAIC algorithm.

3.2.4 Particle Swarm Optimization with Rank-Based Inertia Weight

Panigrahi *et al.* [86] claimed that the PSO algorithm should be redefined such that the movement of the swarm is controlled by the objective function. To this end, the particle swarm optimization with rank-based inertia weight (PSO-RBI) algorithm [86] calculates 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

$$\omega_i(t) = \omega_{min} + (\omega_{max} - \omega_{min})\frac{R_i(t)}{n_s}, \tag{3.22}$$

where $R_i(t)$ is the fitness rank of particle i. The PSO-RBI algorithm tunes the ω parameter at the expense of introducing two new parameters, namely ω_{min} and ω_{max}. From Equation (3.22), it can be seen that the best-fit particle (i.e., rank 1), will be assigned the lowest inertia weight, while the worst fit particle will be assigned the maximal inertia weight.

3.2.5 Improved Particle Swarm Optimization by Li and Tan

The improved particle swarm optimization by Li and Tan (IPSO-LT) [67] is premised on the assumption that the inertia weight should be in direct relation to the convergence factor,

$$c_i(t) = \frac{|f(\mathbf{y}_i(t-1)) - f(\mathbf{y}_i(t))|}{f(\mathbf{y}_i(t-1)) + f(\mathbf{y}_i(t))}, \tag{3.23}$$

as well as the diffusion factor,

$$d_i(t) = \frac{|f(\mathbf{y}_i(t)) - f(\hat{\mathbf{y}}_i(t))|}{f(\mathbf{y}_i(t)) + f(\hat{\mathbf{y}}_i(t))}, \tag{3.24}$$

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

$$\omega_i(t) = 1 - \left|\frac{\alpha(1 - c_i(t))}{(1 + d_i(t))(1 + \beta)}\right|, \tag{3.25}$$

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.

3.2.6 Self-Adaptive Particle Swarm Optimization by Li *et al.*

The self-adaptive particle swarm optimization by Li, Fu, and Zhang (SAPSO-LFZ) [66] 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

$$\omega_i(t) = 0.15 + \frac{1}{1 + e^{F_i(t)}}, \tag{3.26a}$$

with

$$F_i(t) = \overline{f(\mathbf{y}(t))} - f(\mathbf{y}_i(t)), \tag{3.26b}$$

where $\overline{f(\mathbf{y}(t))}$ is the average personal best fitness. Figure 3.4 visualizes the result of Equation (3.26a) 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 Equation (3.26a) may require tuning and could be considered a control parameter.

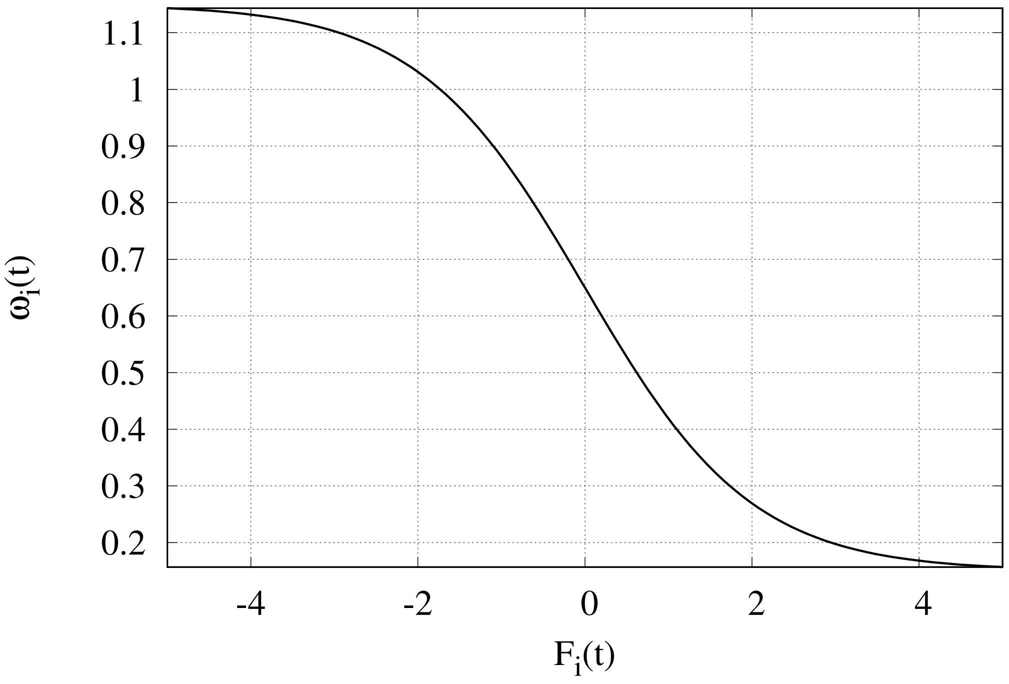


Figure 3.4: The inertia value of the SAPSO-LFZ algorithm based on the value of $F_i(t)$.

3.2.7 Self-Adaptive Particle Swarm Optimization by Dong *et al.*

The self-adaptive particle swarm optimization by Dong, Wang, Chen, and Yu (SAPSO-DWCY) [27] is based on an assumed relationship among various characteristics of the search. The authors proposed that there exists a relationship between the inertia weight, fitness, swarm size, and problem dimension. Dong *et al.* [27] also posited that problems in higher dimensions can benefit from an increased inertia weight, as this would help to escape local optima. Similarly, they claimed that higher inertia weight values, and thereby enhanced exploration, could be used to compensate for smaller swarm sizes. To this end, the inertia weight for a particle i at time t is calculated as

$$\omega_i(t) = \frac{1}{\alpha - e^{-n_s/\beta} + \left(\frac{R_i(t)}{\gamma n_d}\right)^2}, \tag{3.27}$$

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 Equation (3.27) may require tuning and could be considered as additional 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 particles (i.e., particles with relatively good fitnesses) have their inertia weight decreased to encourage exploitation. Figure 3.5 demonstrates the inertia weight values assuming a swarm size of 30 particles and a 50-dimensional problem.

3.2.8 Improved Particle Swarm Optimization by Chen *et al.*

The improved particle swarm optimization by Chen, Li, and Liao (IPSO-CLL) [16] uses an adaptive inertia weight aimed at accelerating the speed of convergence toward an optimum in the PSO algorithm. To this end, the inertia weight at time t is given by

$$\omega(t) = e^{-\lambda(t)}, \tag{3.28a}$$

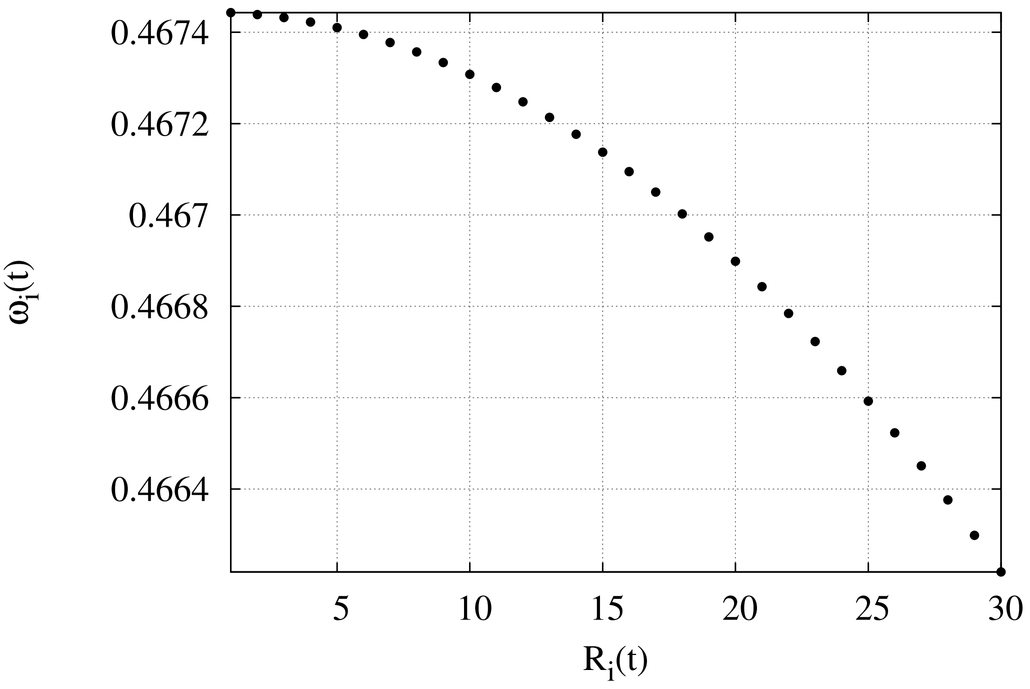


Figure 3.5: The inertia value of the SAPSO-DWCY algorithm based on the fitness rank of the particle, assuming 30 particles in 50 dimensions.

with

$$\lambda(t) = \frac{\alpha(t)}{\alpha(t-1)} \tag{3.28b}$$

and

$$\alpha(t) = \frac{1}{n_s}\sum_{i=1}^{n_s} \left| f(\mathbf{x}_i(t)) - f(\mathbf{y}^*(t)) \right|, \tag{3.28c}$$

where $\mathbf{y}^*(t)$ is the best position found during iteration t. Note that, in Equation (3.28a), the use of the exponential function was not empirically determined, but rather was introduced based on its presence in engineering calculations [16]. 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 state of convergence toward a fixed location among particles. Chen *et al.* [16] claimed that the convergence state of the algorithm was dependent upon the value of $\lambda(t)$[3]; when $\lambda(t) < 1$, the algorithm demonstrated convergent behaviours, while $\lambda(t) > 1$ lead to globally divergent behaviour. Similarly, the value of $\lambda(t)$ will also directly affect the exploration of particles, because a smaller value for $\lambda(t)$ implies a larger $\omega(t)$, as seen in Figure 3.6, and thus increases exploration. The IPSO-CLL algorithm tunes the ω control parameter and introduces no additional parameters.

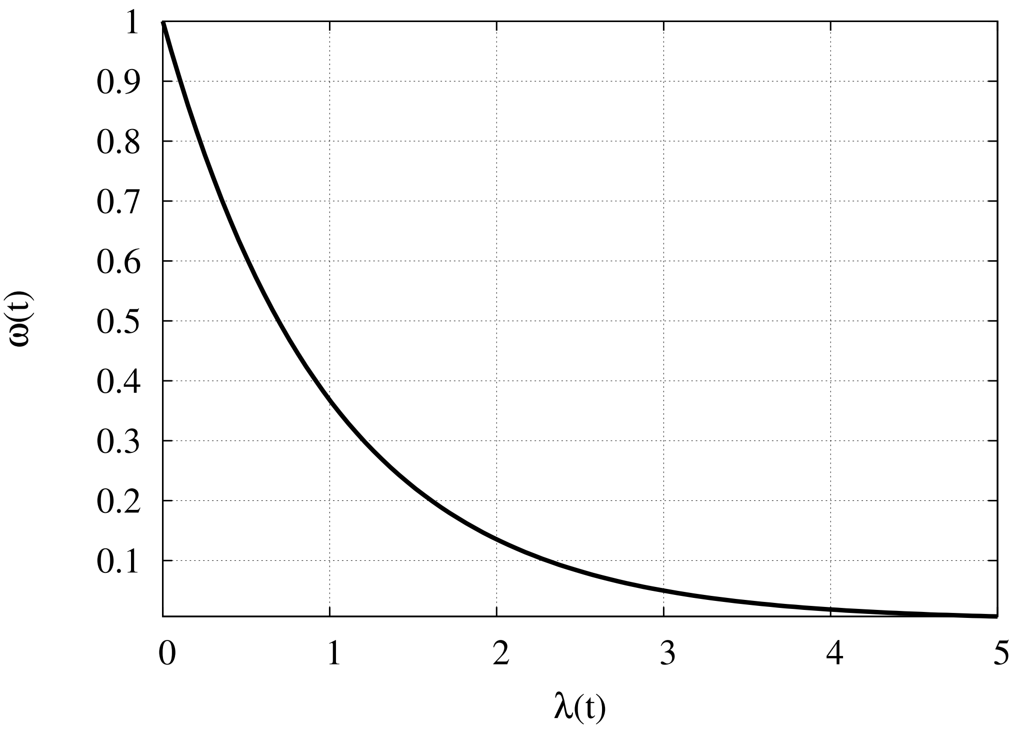


Figure 3.6: The inertia value of the IPSO-CLL algorithm relative to the smoothness function $\lambda(t)$.

3.2.9 Particle Swarm Optimization with Simulated Annealing

The particle swarm optimization with simulated annealing (PSO-ICSA) algorithm [55] adapts both the inertia weight and social acceleration coefficients. Firstly, the “adaptive

[3]No indication of the behaviour when $\lambda(t) = 1$ was provided by Chen *et al.* [16].

coefficient" of particle i at time t, given by

$$\eta_i(t) = \frac{f(\hat{\mathbf{y}}(t-1))}{f(\mathbf{x_i}(t-1))}, \tag{3.29}$$

quantifies the performance of the particle. The adaptive coefficient measures the similarity of a particle's fitness relative to 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 that the particle's fitness is equal to the global best fitness.

The inertia weight of a particle is then given by

$$\omega_i(t) = \omega_a F(\eta_i(t)) + \omega_b, \tag{3.30a}$$

with

$$F(\eta_i(t)) = \begin{cases} 2 & \text{if } \eta_i(t) < 0.0001 \\ 1 & \text{if } 0.0001 \le \eta_i(t) < 0.01 \\ 0.3 & \text{if } 0.01 \le \eta_i(t) < 0.1 \\ -0.8 & \text{if } 0.1 \le \eta_i(t) < 0.9 \\ -5.5 & \text{if } 0.9 \le \eta_i(t) \le 1 \end{cases} \tag{3.30b}$$

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

The social acceleration coefficient of a particle is given by

$$c_{2i}(t) = c_{2a} G(\eta_i(t)) + c_{2b}, \tag{3.31a}$$

with

$$G(\eta_i(t)) = \begin{cases} 2.5 & \text{if } \eta_i(t) < 0.0001 \\ 1.2 & \text{if } 0.0001 \le \eta_i(t) < 0.01 \\ 0.5 & \text{if } 0.01 \le \eta_i(t) < 0.1 \\ 0.2 & \text{if } 0.1 \le \eta_i(t) < 0.9 \\ 0.1 & \text{if } 0.9 \le \eta_i(t) \le 1 \end{cases} \tag{3.31b}$$

where c_{2a} and c_{2b} are user-supplied, positive constants. When $\eta_i(t)$ is small, the social acceleration coefficient is increased as an attempt to attract the particle towards the global best position. This is an explicit attempt to increase the speed of convergence toward a promising position. When $\eta_i(t)$ is large, the social acceleration coefficient is decreased to discourage crowding around the global best position. Additionally, the cognitive acceleration coefficient is decreased linearly according to Equation (3.3a). 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 Equations (3.30b) and (3.31b), may also require tuning, in which case the PSO-ICSA algorithm introduces a further 18 control parameters.

3.2.10 Adaptive Particle Swarm Optimization by Zhan *et al.*

The adaptive particle swarm optimization by Zhan, Zhang, Li, and Chung (APSO-ZZLC) [120] adapts the value for each of the three PSO 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, an evolutionary factor[4] is first calculated based on the spread of particles in the search space, given by

$$f_e(t) = \frac{d_g(t) - d_{min}(t)}{d_{max}(t) - d_{min}(t)}, \tag{3.32a}$$

with

$$d_i(t) = \frac{1}{n_s - 1} \sum_{j=1, j\neq i}^{n_s} \sqrt{\sum_{k=1}^{n_d} (x_{ik}(t) - x_{jk}(t))^2}, \tag{3.32b}$$

where $d_g(t)$ is the value of Equation (3.32b) 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 3.7, are dependent upon the value of f_e given by the following piece-wise functions:

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

Exploration

$$\mu_{S_1}(f_e(t)) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.4 \\ 5f_e(t) - 2 & \text{if } 0.4 < f_e(t) \leq 0.6 \\ 1 & \text{if } 0.6 < f_e(t) \leq 0.7 \\ -10f_e(t) + 8 & \text{if } 0.7 < f_e(t) \leq 0.8 \\ 0 & \text{if } 0.8 < f_e(t) \leq 1.0 \end{cases} \tag{3.33a}$$

Exploitation

$$\mu_{S_2}(f_e(t)) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.2 \\ 10f_e(t) - 2 & \text{if } 0.2 < f_e(t) \leq 0.3 \\ 1 & \text{if } 0.3 < f_e(t) \leq 0.4 \\ -5f_e(t) + 3 & \text{if } 0.4 < f_e(t) \leq 0.6 \\ 0 & \text{if } 0.6 < f_e(t) \leq 1.0 \end{cases} \tag{3.33b}$$

Convergence

$$\mu_{S_3}(f_e(t)) = \begin{cases} 1 & \text{if } 0.0 \leq f_e(t) \leq 0.1 \\ -5f_e(t) + 1.5 & \text{if } 0.1 < f_e(t) \leq 0.3 \\ 0 & \text{if } 0.3 < f_e(t) \leq 1.0 \end{cases} \tag{3.33c}$$

Jumping-out

$$\mu_{S_4}(f_e(t)) = \begin{cases} 0 & \text{if } 0.0 \leq f_e(t) \leq 0.7 \\ 5f_e(t) - 3.5 & \text{if } 0.7 < f_e(t) \leq 0.9 \\ 1 & \text{if } 0.9 < f_e(t) \leq 1.0 \end{cases} \tag{3.33d}$$

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.* [120] posited that the PSO algorithm should transition according to the following sequence $S_1 \to S_2 \to S_3 \to S_4 \to S_1 ...$, 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:

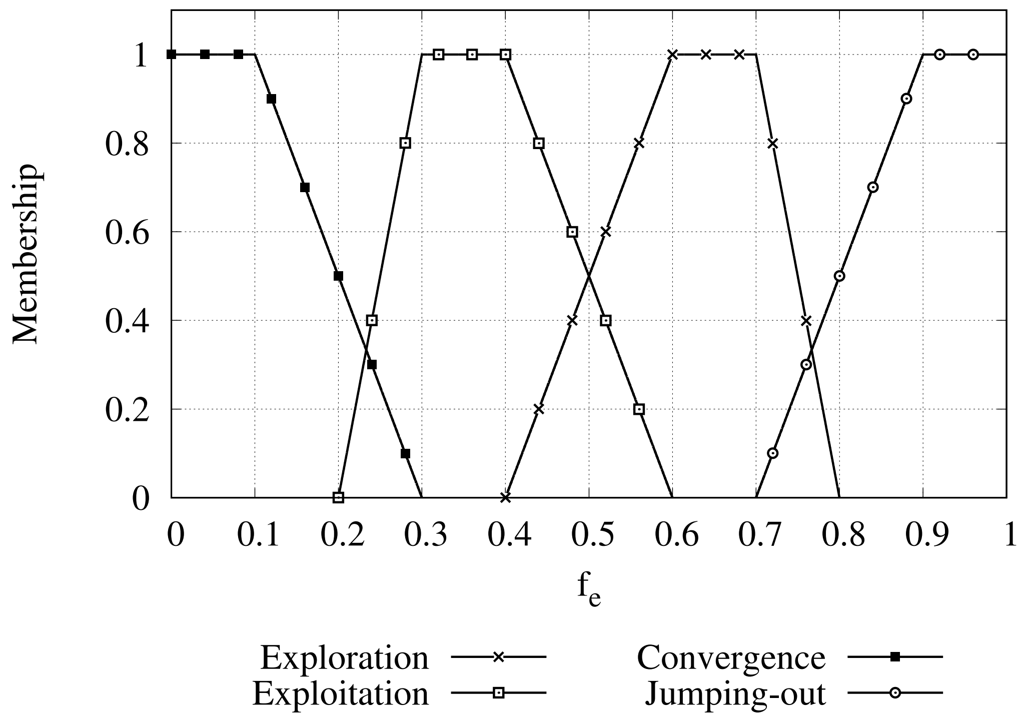


Figure 3.7: The fuzzy membership functions of the APSO-ZZLC algorithm for the four evolutionary states.

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 (e.g., $S_1 \rightarrow S_2$ 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 values of the control parameters can be calculated. The inertia weight is provided by

$$\omega(f_e) = \frac{1}{1 + 1.5e^{-2.6f_e}} \in [0.4, 0.9], \tag{3.34}$$

while the cognitive and social acceleration coefficients are either increased or decreased based on the algorithmic state, as described in Table 3.1. An entry which indicates an

Table 3.1: 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

"Increase" denotes that the value of the corresponding control parameter is increased by $\delta(t)$ and an entry marked "Decrease" denotes that the value of 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

$$c_i(t) = 4.0\frac{c_i(t)}{c_1(t) + c_2(t)}, \quad i = 1, 2$$

to effectively bound the range of $c_1(t) + c_2(t)$ to [3.0, 4.0], as suggested by their earlier work [119].

Note that, while the APSO-ZZLC algorithm can be employed without any additional parameters, there are numerous constants that may be treated as control parameters. Specifically, for each of the membership functions in Equation (3.33), 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, Equation (3.34) 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 constants are assumed to require tuning.

3.2.11 Adaptive Inertia Weight Particle Swarm Optimization

The adaptive inertia weight particle swarm optimization (AIWPSO) algorithm [82] 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 that improved their personal best position during iteration t. The inertia weight is then adapted according to

$$\omega(t) = (\omega_{max} - \omega_{min})P_s(t) + \omega_{min}, \tag{3.35a}$$

with

$$P_s(t) = \frac{\sum_{i=1}^{n_s} S_i(t)}{n_s} \tag{3.35b}$$

and

$$S_i(t) = \begin{cases} 1 & \text{if } f(\mathbf{y}_i(t)) < f(\mathbf{y}_i(t-1)) \\ 0 & \text{otherwise.} \end{cases} \tag{3.35c}$$

The justification for this behaviour was that the algorithm increases the inertia weight when particle successes are high to heighten exploration, and decreases the inertia weight when particle successes are low to enhance exploitation [82]. 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}.

3.2.12 Fine Grained Inertia Weight Particle Swarm Optimization

The fine-grained inertia weight particle swarm optimization (FG-PSO) algorithm [14, 25] provides individualized inertia weights for each particle given by

$$\omega_i(t+1) = \omega_i(t) - \left((\omega_i(t) - 0.4)e^{-||\hat{\mathbf{y}}(t),\mathbf{y}_i(t)||_2 \frac{t}{T})}\right). \tag{3.36}$$

Additionally, the inertia weight of each particle is initialized to 0.9. The FG-PSO algorithm thus tunes the value of the ω parameter without introducing any new parameters. However, the value of 0.4 in Equation (3.36) may require tuning.

3.2.13 Adventurous Unified Particle Swarm Optimization

The adventurous unified adaptive particle swarm optimization (UAPSO-A) algorithm [46, 47] adapts the value of each control parameter using an independent learning automaton [80]. 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 [80]. 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 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 $\omega, c_1,$ and c_2. Use of the selected parameters constitutes applying the action to the environment, and their performance is then 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 that have improved their fitness during the current iteration. If the proportion of particles that 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 when employing the parameter at index i, the probabilities are updated according to

$$p_j(t+1) = \begin{cases} p_j(t) + a(1 - p_j(t)) & \text{if } i = j \\ p_j(t)(1-a) & \text{otherwise,} \end{cases} \tag{3.37}$$

where a is the reward step size. Note that, the probability of selection is increased for the parameter value that was used during the successful iteration, while the remainder of the parameter values have their probabilities decreased slightly. Conversely, for an unsuccessful iteration employing parameter i, the probabilities are updated according to

$$p_j(t+1) = \begin{cases} p_j(t)(1-b) & \text{if } i = j \\ \frac{b}{|A|-1} + p_j(t)(1-b) & \text{otherwise,} \end{cases} \tag{3.38}$$

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.

3.2.14 Grey Particle Swarm Optimization

The grey particle swarm optimization (GPSO) algorithm [64] uses a measure of similarity for finite sequences under incomplete information, namely grey relational analysis [54], to aid with control parameter adaptation. Grey relational analysis is used to modify the inertia weight and social acceleration coefficient based on the grey relational grade. To calculate the grey relational grade, the relational coefficient of particle i is first calculated according to

$$r_{ij}(t) = \frac{\Delta_{min}(t) + \xi\Delta_{max}(t)}{\Delta_{ij}(t) + \xi\Delta_{max}(t)}, \tag{3.39a}$$

with

$$\Delta_{ij}(t) = |\hat{y}_{ij}(t) - x_{ij}(t)|, \tag{3.39b}$$

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

$$g_i(t) = \sum_{j=1}^{n_d} (\alpha_j r_{ij}(t)), \tag{3.40a}$$

such that

$$\sum_{j=1}^{n_d} \alpha_j = 1, \tag{3.40b}$$

where α_j is the weighting factor of the relational coefficient for dimension j, and n_d is the number of dimensions in the current problem. In general, it is acceptable to set $\alpha_j = \frac{1}{n_d}$ for all dimensions j [64]. 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

$$\omega_i(t) = \frac{\omega_{min} - \omega_{max}}{g_{max}(t) - g_{min}(t)} g_i(t) + \frac{\omega_{max} g_{max}(t) - \omega_{min} g_{min}(t)}{g_{max}(t), -g_{min}(t)} \tag{3.41}$$

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 value of the social acceleration coefficient according to

$$c_{2i}(t) = \frac{c_{max}(t) - c_{min}(t)}{g_{max}(t) - g_{min}(t)} g_i(t) + \frac{c_{min}(t) g_{max}(t) - c_{max}(t) g_{min}(t)}{g_{max}(t) - g_{min}(t)}, \tag{3.42}$$

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

$$c_{min}(t) = (C_{final} - C_{min}) \frac{t}{T} + C_{min} \tag{3.43a}$$

and

$$c_{max}(t) = (C_{final} - C_{max}) \frac{t}{T} + C_{max}, \tag{3.43b}$$

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 acceleration coefficient is given by

$$c_{1i}(t) = 4.0 - c_{2i}(t), \tag{3.44}$$

such that $\forall i, c_{1i}(t) + c_{2i}(t) = 4.0$. 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 ξ. Additionally, the values for α_j can be interpreted as control parameters, leading to an additional n_d parameters.

3.2.15 Double Exponential Self-Adaptive Inertia Weight Particle Swarm Optimization

The double exponential self-adaptive inertia weight particle swarm optimization (DE-PSO) algorithm [14] incorporates a double exponential function, referred to as a Gompertz function, to select the inertia weight according to

$$\omega_i(t) = e^{-e^{-R_i(t)}}, \tag{3.45a}$$

with

$$R_i(t) = ||\hat{\mathbf{y}}(t), \mathbf{y}_i(t)||_2 \left(\frac{T-t}{T}\right). \tag{3.45b}$$

The inertia weight of each particle is initialized to 0.9. The DE-PSO algorithm thus tunes the value of the ω parameter without introducing any new parameters.

3.2.16 Adaptive Parameter Tuning Based on Velocity Information

The adaptive parameter tuning of particle swarm optimization based on velocity information (APSO-VI) algorithm [113] adapts the inertia weight based on the current velocities of the particles, with the intent to push the velocity closer to their definition of an "ideal" velocity. The concept of a decreasing target velocity in the APSO-VI algorithm is borrowed from earlier work by Yasuda *et al.* [117], 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

$$\overline{v(t)} = \frac{1}{n_d n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n_d} |v_{ij}(t)|, \tag{3.46}$$

where n_d and n_s represent the number of problem dimensions and the size of the swarm, respectively. Xu [113] proposed an equation defining the ideal (average) velocity such that the ideal velocity decreases over time, leading to heightened exploitation near the end of the search. Furthermore, Xu [113] claimed that the ideal velocity must be non-linear and should have long exploratory and exploitative phases with a minimal transition period. The ideal velocity at time t for the APSO-VI algorithm is given by

$$v_{ideal}(t) = v_s \left(\frac{1 + \cos\left(\pi \frac{t}{T_{0.95}}\right)}{2} \right), \tag{3.47}$$

where v_s is the initial 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 3.8.

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

$$\omega(t+1) = \begin{cases} \max\{\omega(t) - \Delta\omega, \omega_{min}\} & \text{if } \overline{v(t)} \geq v_{ideal}(t+1) \\ \min\{\omega(t) + \Delta\omega, \omega_{max}\} & \text{if } \overline{v(t)} < v_{ideal}(t+1), \end{cases} \tag{3.48}$$

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 $\Delta\omega$.

3.2.17 Self-Regulating Particle Swarm Optimization

The self-regulating particle swarm optimization (SRPSO) algorithm [104] controls the inertia weight of each particle such that the value is increased for the best particle, and decreased for all other particles. This adaptation scheme was premised on the idea that the best particle of the swarm should have a high level of confidence in its direction and thus accelerate more rapidly [104]. The remainder of particles then follow a linearly

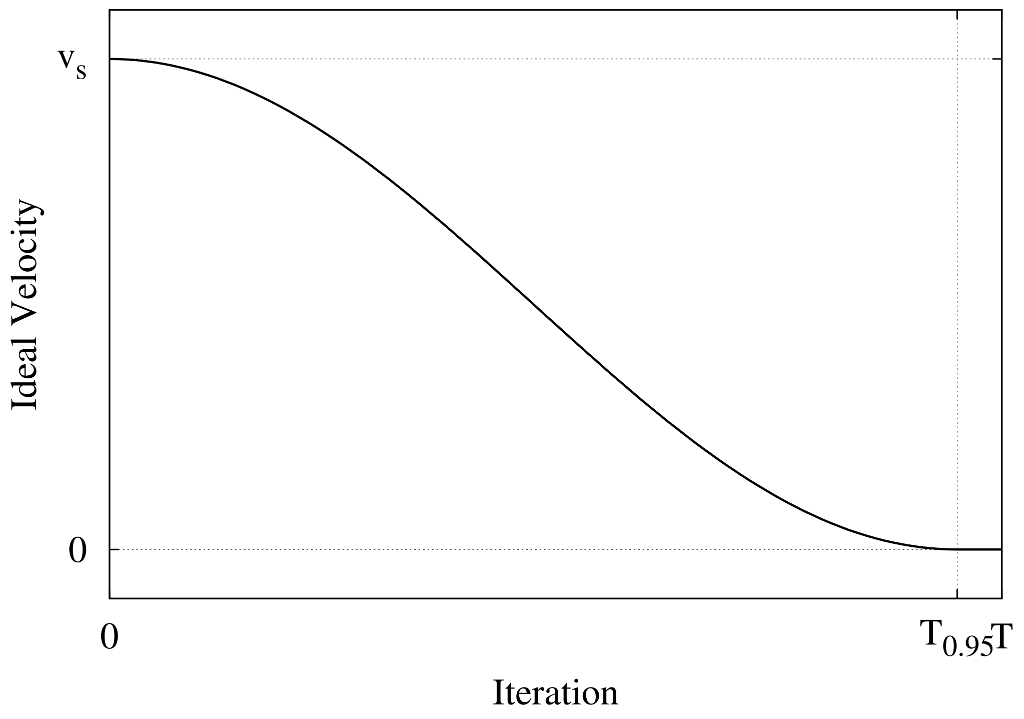


Figure 3.8: The ideal velocity of APSO-VI as a function of time.

decreasing inertia weight strategy, similar to the PSO-LDIW algorithm. The inertia weight in the SRPSO algorithm is given by

$$\omega_i(t) = \begin{cases} \omega_i(t-1) + \eta\Delta\omega & \text{for the best particle} \\ \omega_i(t-1) - \Delta\omega & \text{for all other particles,} \end{cases} \tag{3.49a}$$

with

$$\Delta\omega = \frac{\omega_s - \omega_f}{T}, \tag{3.49b}$$

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 SRPSO algorithm thus tunes the value of the ω parameter, but introduces three additional parameters, namely ω_s, ω_f, and η.

3.3 Summary

This chapter provided a review of self-adaptive particle swarm optimizers. Section 3.1 presented time-variant approaches, which vary control parameters based on the iteration. In Section 3.2, a number of true self-adaptive algorithms, i.e., those which adapt their control parameters based on the algorithmic state, were presented.

Table 3.2 summarizes the SAPSO algorithms discussed by outlining the control parameters they tune and the net change in the total number of control parameters relative to the standard PSO. Values in the 'Net Change' column of Table 3.2 should be interpreted as the overall change in the number of control parameters, relative to the standard PSO algorithm, while entries in the 'Net Change with Constants' column indicate the net change if the constants defined by that algorithm are treated as control parameters. 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 standard PSO. For the case of GPSO, n_d is the number of problem dimensions.

Table 3.2 shows that a significant amount of effort has been devoted to solely adapting the value of the inertia weight parameter. A further key observation is regarding the net change in the number of parameters: of the 29 examined SAPSO algorithms, only nine depict a net reduction in the number of parameters. Removal of the algorithm that simply generates random parameters adhering to the stability criterion each iteration (i.e., PSO-RAC) leaves eight out of 28 algorithms that 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 three of the algorithms lead to a reduction of parameters, while two algorithms lead 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 that 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.

Clearly, many of the examined algorithms only adapt the inertia coefficient. The exploration/exploitation trade-off is arguably most easily controlled using the inertia coefficient, as evidenced by its leading presence in self-adaptive strategies. However, the performance of PSO is also heavily dependent upon the social and cognitive acceleration coefficients. While some of the examined algorithms do vary the cognitive and social acceleration coefficients, their adaptation is typically secondary to the adaptation of the inertia coefficient. Furthermore, it is more common that the social acceleration coefficient is tuned, while the cognitive coefficient remains either static or time-dependent. Evidently, there is a need for self-adaptive PSO variants that introduce minimal control parameters and, when necessary, provide guidelines for selecting their values.

Given that many of the SAPSO techniques introduced in this chapter take no regard for the theoretical criterion needed for particle stability to be exhibited, an investigation of whether the algorithms will lead to stability is warranted. In the following chapter, an analysis of stability is provided for various SAPSO algorithms discussed in this chapter.

Table 3.2: Net change in the number of control parameters of various SAPSO algorithms relative to the standard PSO. Entries in the 'Net Change with Constants' column 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	Net Change with Constants
PSO-LDIW [98, 99]	ω	+1	–
PSO-RIW [29]	ω	-1	+1
PSO-AIWF [70]	ω	+1	–
PSO-NEIW [15]	ω	+1	+2
PSO-SIW [62]	ω	0	–
CDIW-PSO [34]	ω	+1	–
DW-PSO [33]	ω	-1	+1
DAPSO [116]	ω	+2	–
LD-PSO [37]	ω	+2	+3
PSO-NLI [53]	ω	+1	–
IPSO-LT [67]	ω	+1	–
SAPSO-LFZ [66]	ω	-1	0
SAPSO-DWCY [27]	ω	-1	+2
PSO-RBI [86]	ω	+1	–
PSO-OIW [61]	ω	+2	+6
IPSO-CLL [16]	ω	-1	–
FG-PSO [14, 25]	ω	-1	0
AIWPSO [82]	ω	+1	–
DE-PSO [14]	ω	-1	–
APSO-VI [113]	ω	+2	–
PSO-NL [115]	ω	+2	–
SRPSO [104]	ω	+2	–
PSO-SAIC [112]	ω, c_2	+2	+4
PSO-RAC	ω, c_1, c_2	-3	–
PSO-TVAC [94]	ω, c_1, c_2	+3	–
PSO-ICSA [55]	ω, c_1, c_2	+3	+31
APSO-ZZLC [120]	ω, c_1, c_2	-3	+35
UAPSO-A [46]	ω, c_1, c_2	+6	–
GPSO [64]	ω, c_1, c_2	+3	$+(n_d + 3)$

Chapter 4

Stability Analysis of Self-Adaptive Particle Swarm Optimizers

The focus of this chapter is to analyse the order-2 stability of various SAPSO techniques discussed in Chapter 3. Their respective parameter adaptation mechanisms are analytically dissected to determine if and when they will lead to stable behaviour. Moreover, to support and complement the analytical findings, their search behaviour is empirically examined on a specially-formulated benchmark problem that isolates the stagnation behaviour. Algorithms are presented in order of the parameters they adapt.

The remainder of this chapter is structured as follows. The experimental design is provided in Section 4.1. The results of both the theoretical and empirical analysis are provided in Section 4.2. Finally, a summary of the findings in this chapter are presented in Section 4.3.

4.1 Empirical Analysis of Self-Adaptive Particle Swarm Optimizers

The empirical results in this chapter were obtained via a specially-formulated benchmark function, which is a vertically-shifted version of the function proposed by Cleghorn and Engelbrecht [19, 20]. This function provided the algorithms with an environment in which complete stagnation was highly unlikely, and thereby isolated the ability for particles to attain stability. The function is given by

$$F(\mathbf{x}) \sim U(0, 2000), \tag{4.1a}$$

such that

$$F(\mathbf{x}_1) = F(\mathbf{x}_2) \text{ if } \mathbf{x}_1 = \mathbf{x}_2. \tag{4.1b}$$

Equation (4.1a) thus defines the fitness value of each position in the search space as a randomly-sampled real value within the range [0, 2000]. Equation (4.1b) 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 was randomly determined, but remained fixed throughout each independent simulation. Given the function has many discontinuities and is completely unstructured, an optimizer is unlikely to enter complete stagnation [20]. Therefore, the function is a reasonable synthetic environment for examining the ability of the PSO algorithm to exhibit order-2 stability without the stability being a direct result of particle stagnation. Note that the function was shifted to produce strictly non-negative fitness values, because 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 Equation (4.1) to correctly identify scenarios in which the particles' behaviour is stable [19].

To empirically measure stability, the average Euclidean distance of the particles' movement according to

$$\Delta(t+1) = \frac{1}{n_s} \sum_{i=1}^{n_s} ||\mathbf{x}_i(t+1) - \mathbf{x}_i(t)||, \tag{4.2}$$

where n_s is the number of particles, was also measured. A sensible upper threshold, Δ_{max}, was defined as the maximum distance between any two points in the feasible search space [20], according to

$$\Delta_{max} = \sqrt{n_d(l-u)^2}, \tag{4.3}$$

where $[l, u]$ is the (feasible) domain of the objective function, and n_d is the dimensionality of the problem. Given that Equation (4.3) refers to the maximal distance between two points in the search space, particles that exhibited movement values above this value were considered to be demonstrating divergent behaviour. Note that, while this is a practical definition of divergent behaviour, not a theoretical definition, the use of Δ_{max} as the criterion to empirically determine particle divergence has been demonstrated to be 98.79% accurate when correlated with the criterion in Equation (2.5) [20]. In this investigation, the domain of the problem was fixed at $[-100, 100]^{50}$, and therefore, $\Delta_{max} = 1414.214$. Additionally, the movement values were capped at 2000 to prevent excessively divergent behaviour from muddling the results.

An analogous version of Equation (4.2) was 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 control parameter values leading to theoretical stability (i.e., values that adhere to Equation (2.5)) and the percentage of particles that were outside the feasible region (i.e., have a bound violation in at least one dimension) were measured at each iteration.

Note that the purpose of this chapter is not to empirically analyse and compare algorithms to determine which performs best, but only to analyse their search behaviour. Furthermore, this chapter 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 author's knowledge, still unanswered. Finally, the purpose of this chapter is not to conduct a sensitivity analysis of the parameters introduced by each algorithm.

4.2 Critical Analysis of Self-Adaptive Particle Swarm Optimizers

Chapter 3 indicated that SAPSO algorithms fall broadly into two categories, namely time-variant approaches and (true) self-adaptive approaches. This section presents a critical analysis of a number of algorithms from both categories. A theoretical analysis of the order-2 stability was performed, ultimately describing the algorithmic conditions necessary for the respective algorithm to exhibit stability. Furthermore, Equation (4.1) was employed to empirically investigate the convergence of both the particles and the control parameter values to fixed values. This empirical investigation provides an individualized profile of the search behaviour of each algorithm and primarily serves to support and complement the analytical results, rather than to provide a direct comparison between algorithms.

The four performance measures employed to evaluate the behaviour of each algorithm are summarized as follows:

1. The **average particle movement** was calculated using Equation (4.2) and quantifies the average particle step size. If particle steps sizes do not decrease, particles will not converge towards a fixed point. Furthermore, large step sizes may prevent particles from exploiting promising locations in the search space.

2. The **percentage of particles with theoretically stable control parameters** measures the proportion of particles that employ parameter settings that adhere to Poli's criterion in Equation (2.5). This measure provides an indication of an algorithm's ability to generate parameters adhering to order-2 stability criterion.

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 according to Equation (4.2), using the control parameter values rather than the particle position.

4. The **percentage of particles with a bound violation** measures the proportion of particles that violated the boundary constraints in at least one dimension. This measure provides an indication of the search effort that 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 star topology and a synchronous iteration strategy [12]. Particle positions were randomly initialized within the search space, and their initial velocity was set to the zero vector [30]. To prevent infeasible 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. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. Algorithmic parameters, as described below and summarized in Table 4.1, 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 suggested by the respective authors and, therefore, this study examines the 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 require tuning.

4.2.1 Particle Swarm Optimization with Linearly-Decreasing Inertia Weight

As employed by Shi and Eberhart [99], the parameters for the PSO-LDIW algorithm were set as $\omega_s = 0.9$ and $\omega_f = 0.4$, with $c_1 = c_2 = 1.49618$. Given that $\forall t, \omega(t) \geq 0.4$, Equation (2.5) can be simplified to $\omega(t) < 0.78540$. From Equation (3.1), it follows that $\omega(t) = 0.78540$ occurs when $\frac{t}{T} = 0.22920$, i.e., when approximately 22.9% of the search process has completed. Thus, the PSO-LDIW algorithm is not expected to exhibit stability until nearly a quarter of the search process has completed, but will exhibit stable behaviour overall.

As Figure 4.1a depicts, the average particle movement for PSO-LDIW dropped below Δ_{max} after roughly 1000 iterations. However, as Figure 4.2a shows, the resulting control parameter values were not below the threshold 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 stability criterion. Given the aforementioned parametrization, the inertia weight value changed by 0.0001 each iteration leading to the constant, near-zero parameter movement values shown in Figure 4.3a. The PSO-LDIW algorithm depicted a gradual decrease in particle bound violations over time, as shown in Figure 4.4a. This was likely a result of the decreasing inertia weight gradually altering the behaviour of the algorithm from exploration to exploitation as the search progressed.

4.2.2 Particle Swarm Optimization with Adaptive Inertia Weight Factor

Liu *et al.* [70] employed parameters $\omega_{min} = 0.2, \omega_{max} = 1.2$, and $c_1 = c_2 = 2$ for the PSO-AIWF algorithm, which leads to a range of $\frac{1}{3} < \omega_i(t) < \frac{1}{2}$ for stable behaviour to be exhibited. Equation (3.14) suggests that any particle with a fitness value worse than the average will have an inertia value of 1.2, and therefore will not demonstrate stable behaviour. Furthermore, the only scenario in which stable behaviour can occur is if

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

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 unstable behaviour on any given iteration, the PSO-AIWF algorithm is expected to demonstrate rapid divergence of particle positions.

The rapid divergence of the PSO-AIWF algorithm is exemplified by the average particle movement, which immediately reached the maximal value of 2000 and never decreased, as shown in Figure 4.1b. Moreover, Figure 4.2b depicts that no particles employed theoretically stable parameter configurations, thus indicating that Equation (3.14) was unsuccessful at generating inertia values within the necessary range. In fact, Figure 4.3b depicts that the average change in parameter space was immediately near zero, suggesting that the adaptation mechanism was failing to adapt the inertia weight altogether. As a result, particles immediately exited the feasible region and never returned; Figure 4.4b indicates that 100% of the particles were outside the bounds of the search space throughout the entirety of the search.

4.2.3 Dynamic Adaptation Particle Swarm Optimization

Yang *et al.* [116] concluded that the parametrization of DAPSO, specifically the α and β parameters, does not have a significant impact on performance. Therefore, the parameter values used in this work, namely $\alpha = 1.0, \beta = 0.1, \omega_s = 1.0$, and $c_1 = c_2 = 1.496180$, were taken from previous studies [109, 116].

From Equation (3.18), it follows that $\forall t, 0.0 \leq \omega_i(t) \leq 1.1$, and, given the aforementioned values for the social and cognitive control parameters, the DAPSO algorithm thus requires $\omega_i(t) < 0.78540$ to exhibit stability. Equation (3.17) can then be simplified to

$$\omega_i(t) = h_i(t) + 0.1s(t), \tag{4.4}$$

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

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

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

$$h_i(t) < 0.68450$$

becomes the predominant requirement for particle stability. 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 stable 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 depicted immediately divergent behaviour, as shown in Figure 4.1c. Furthermore, as Figures 4.2c and 4.3c demonstrate, the parameters generated via the adaptation mechanism were never within the stable region and were thus wildly unstable. In fact, the adaptation mechanism failed to generate reasonable parameters as a result of the explosive divergence, which caused fitness values to deteriorate such that they became too large to represent as an IEEE-754 64-bit floating point value. As a result, Figure 4.4c shows that particles immediately exited the feasible region and never returned. Thus, the adaptation mechanism of the DAPSO algorithm is wildly deficient.

4.2.4 Improved Particle Swarm Optimization by Li and Tan

For the IPSO-LT algorithm, Li and Tan [67] employed parameter values 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 $\alpha = \beta = 0.5$. Substituting $\alpha = \beta = 0.5$ into Equation (3.25) provides a simpler equation for the inertia weight, given by

$$\omega_i(t) = 1 - \left| \frac{1 - c_i(t)}{3(d_i(t) + 1)} \right|, \tag{4.5}$$

which further simplifies the stability criterion such that

$$\frac{2}{3} < \left| \frac{1 - c_i(t)}{d_i(t) + 1} \right| < 2 \tag{4.6}$$

is required for stable behaviour to be exhibited.

As evidenced by the average particle movement in Figure 4.1d, the IPSO-LT algorithm demonstrated immediate divergent behaviour. Thus, it can be concluded that the condition in Equation (4.6) was never satisfied, thereby causing all particles to have divergent behaviours. This is further evidenced by Figure 4.2d depicting that no particles employed parameter configurations that are theoretically stable. Moreover, Figure 4.3d shows that the values of the parameters were not changing over time, indicating that the adaptation mechanism is inherently flawed and is incapable of tuning the inertia weight control parameter. However, Figure 4.4d shows that not all particles were outside of the search space. Specifically, the particle corresponding to the best position remained within the feasible region, given that it would have zero velocity as a result.

4.2.5 Self-Adaptive Particle Swarm Optimization by Li *et al.*

Li *et al.* [66] employed parameter values of $c_1 = c_2 = 1.496180$ for the SAPSO-LFZ algorithm, which simplifies the stability criterion to

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

From Equation (3.26a), it follows that $\forall t, 0.15 < \omega_i(t) < 1.15$, and furthermore, $\omega_i(t) < 0.78540$ occurs when

$$f(\mathbf{y}_i(t)) < \overline{f(\mathbf{y}_i(t))} + 0.55545. \tag{4.7}$$

Thus, only particles that have a personal best fitness that is no greater than 0.55545 above the average will exhibit stable behaviour. Given that particles tend to roam and exit the feasible search space early in the search [30, 32], 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 4.1e depicts that the particle movement values immediately hit the maximum value of 2000 and never decreased, 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 stable behaviour, it was expected that roughly 50% of the particles will exhibit stable behaviour. As evidenced by Figure 4.2e, just under 50% of the particles showed stable behaviour at any given iteration. To further support this observation, Figure 4.4e shows that just over 50% of particles (i.e., those without theoretically stable parameter configurations) were outside the search space after only a few iterations. However, as Figure 4.3e shows, there was very little adaptation of parameters occurring, which is indicative of stagnating personal best fitness values in the SAPSO-LFZ algorithm. Thus, as expected, initially divergent behaviours caused no improvements in fitness, thereby causing the adaptation mechanism to have virtually no effect.

4.2.6 Self-Adaptive Particle Swarm Optimization by Dong *et al.*

Dong *et al.* [27] employed control parameter values of $c_1 = c_2 = 2.0$ for the SAPSO-DWCY algorithm, leading to the necessary condition of

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

for stable behaviour to be depicted. By substituting minimum and maximum ranks of 1 and 30 into Equation (3.27), 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 stable range, and therefore the SAPSO-DWCY algorithm will exhibit stable 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 stable region. As a result, the SAPSO-DWCY algorithm is expected to exhibit an unreasonably slow decline in particle movement [19].

While Figure 4.1f does depict the average particle movement value was below the threshold for most iterations, there was no decline in particle movement over time; the average particle movement value hovered around 1400. Moreover, as Figures 4.2f and 4.3f depict, the parameters were always within the stable region and were undergoing virtually no change over time. While the SAPSO-DWCY algorithm could thus be considered as stable in a sense, it cannot be said that the particles were stagnating nor that the swarm was converging to a stable point. Finally, despite the categorically stable behaviour, Figure 4.4f demonstrates that no solutions were retained within the feasible region; the particles immediately exited the feasible region and never returned.

4.2.7 Particle Swarm Optimization with Rank-Based Inertia

Parameter values of $\omega_{min} = 0.4$ and $\omega_{max} = 0.9$ were employed by Panigrahi *et al.* [86] for the PSO-RBI algorithm. However, there was no mention of the social and cognitive acceleration coefficients employed, and thus sensible defaults of $c_1 = c_2 = 1.496180$ [107] are used for the purposes of this study. Assuming a swarm size of 30 particles, it follows from Equation (3.22) that the PSO-RBI algorithm will exhibit stability when $\frac{R_i(t)}{n_s} \geq 0.22920$. That is, when a particle is ranked within the worst 22.9% of the swarm, the particle will exhibit divergent behaviour. Therefore, more than one fifth of the swarm will exhibit divergent behaviour at any given time, causing the PSO-RBI to exhibit overall divergent behaviour.

The divergent behaviour of the PSO-RBI algorithm is evidenced by the particle movement over time in Figure 4.1g, which depicts a rapid increase in particle movement until the value reached 2000, from which it never decreased. The empirical results in Figure 4.2g suggest that 20% of the particles employed unstable parameter values, which is directly in line with the theoretical prediction. Figure 4.3g demonstrates that a non-zero particle movement was maintained throughout the search, indicating the adaptive mechanism did not stagnate, and thereby suggesting that the rank of particles was constantly changing. Despite the overall divergent behaviour depicted by the PSO-RBI algorithm, Figure 4.4g demonstrates that the number of infeasible particles was 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 remained within the bounds of the feasible region.

4.2.8 Improved Particle Swarm Optimization by Chen *et al.*

Chen *et al.* [16] employed control parameter values of $c_1 = c_2 = 2.0$ in their initial proposal of the IPSO-CLL algorithm. Substitution of these values into Equation (2.5) reduces the stability criterion to

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

for the IPSO-CLL algorithm. It then follows from Equation (3.28a) that

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

is necessary for stability to occur. Moreover, order-2 stability 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, stability 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 4.1h 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 theoretically stable parameters, as evidenced by Figure 4.2h. The only reasonable explanation for this anomalous event was that the parameter values were generated sufficiently close to the boundaries of the stable region, such that the movement values were divergent; this is likely an example of the slight inaccuracy associated with using Δ to classify empirical stability [19]. As Figure 4.3h indicates, the adaptation mechanism completely failed to adapt the parameters and lead to no change in the parameter values over time. Finally, as indicated in Figure 4.4h, particles immediately exited feasible space and never returned. Therefore, the adaptation mechanism of the IPSO-CLL algorithm was ineffective at adapting the values of the control parameters.

4.2.9 Adaptive Inertia Weight Particle Swarm Optimization

For the AIWPSO algorithm, Nickabadi *et al.* [82] employed parameter values of $\omega_{min} = 0.0$ and $\omega_{max} = 1.0$, while no indication of the values for the social and cognitive control parameters was given. Therefore, the commonly employed parameters of $c_1 = c_2 = 1.496180$ [107], which are theoretically stable when $\omega(t) < 0.78540$, are assumed. Given the parameter values of $\omega_{min} = 0.0$ and $\omega_{max} = 1.0$, Equation (3.35a) simplifies to

$$\omega(t) = P_s(t), \tag{4.8}$$

and therefore, $\omega(t) < 0.78540$ occurs when $P_s(t) < 0.78540$. This behaviour indicates that the swarm may not exhibit stability when more than 78.54% of the particles improve their personal best fitness in any given iteration. However, it is unrealistic to sustain such a high success rate for any extended period of time due to the stochastic, exploratory nature of particle movement, which effectively prevents particles from always moving in the optimal direction. Thus, the AIWPSO algorithm is expected to exhibit stability.

As expected, Figure 4.1i demonstrates the average particle movement was well below Δ_{max}, while Figure 4.2i depicts that the generated parameter configurations were always within the stable region. These two observations provide conclusive evidence that the high rate of improvement necessary to not satisfy the theoretical stability criterion was unreasonable and will likely not be achieved in practice. However, despite stability being exhibited, particle movement in the AIWPSO was strikingly low, which suggests that the algorithm suffers from premature convergence. This is further supported by the stagnant parameter movement, as shown in Figure 4.3i. Thus, the adaptation mechanism of AIWPSO struggled to effectively adapt the parameters during the search. The average percentage of particle bound violations, shown in Figure 4.4i, was by far the lowest among all the examined algorithms. Over 80% of the particles were within the search space after only a few iterations, while only four of the remaining algorithms even attained 80% feasible solutions within 5000 iterations.

4.2.10 Adaptive Particle Swarm Optimization Based on Velocity Information

Xu [113] used parameters $\omega_{min} = 0.3$, $\omega_{max} = 0.9$, and $\Delta\omega = 0.1$ with $c_1 = c_2 = 1.496180$ for the APSO-VI algorithm. As the ideal velocity of Xu [113] decreases over time, the employed value of $\omega(t)$ will also decrease over time and thus will eventually stabilize within the theoretical range for stability. Furthermore, given that the ideal velocity tends toward zero, by definition the average velocity will also tend toward zero regardless of the inertia weight. Therefore, the APSO-VI algorithm is expected to exhibit stability.

Figure 4.1j 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 magnitude of particle movement. The percentage of stable particles over time, shown in Figure 4.2j, depicted an inverse relationship with the particle movement. That is, the number of particles with stable parameters showed a gradual increasing trend over time, such that after approximately 4000 iterations, the entire swarm had stable parameters. It is noted that around the same time that all particles employed stable parameter configurations, i.e., around 4000 iterations, the parameter movement also stabilized. This is evidenced by Figure 4.3j, which shows that the parameter adaptation abruptly ceased just after 4000 iterations. Finally, the percentage of particles outside the feasible region over time is shown in Figure 4.4j and indicated that almost all particles immediately exited the feasible region and did not return until nearly half of the search had completed. However, during the second half of the search, particle bound violations steadily decreased, eventually stabilizing around 6%. Thus, the adaptation mechanism of APSO-VI was successfully controlling the inertia weight.

4.2.11 Self-Regulating Particle Swarm Optimization

Tanweer *et al.* [104] employed parameters of $\omega_s = 1.05, \omega_f = 0.5$, $\eta = 1$, and $c_1 = c_2 = 1.49445$ in their study of the SRPSO algorithm. Furthermore, the following assumptions were made to continue with the analysis: a swarm size of 30, 5000 iterations were to be executed, and each particle was equally likely to have the best fitness at any iteration. It follows that this parametrization can only lead to stability when

$$2572.69 < t < 11816.70.$$

Therefore, the SRPSO algorithm will only exhibit stability 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 4.1k depicts the particle movement over time for the SRPSO algorithm. As expected, the algorithm was divergent for the first half of the search. However, it took slightly longer than the theoretically-predicted 2573 iterations to begin exhibiting stable behaviour, suggesting that the observed probability of particles attaining the best position was not uniform. Furthermore, the movement values only briefly fell below the threshold before gradually increasing back to the maximum value of 2000. Figure 4.2k shows that after roughly half of the search had completed, only 3.3% of the particles (i.e., a single particle) employed control parameters that did not adhere to the stability criterion. This further evidences that the best position was predominantly obtained by a single particle, and thereby caused this particle to exhibit divergent behaviour. As shown in Figure 4.3k, no parameter changes occurred after 5000 iterations. However, as shown in Figure 4.4k, the percentage of particles with a bound violation decreased near the end of the search. Despite nearly all of the particles initially exiting the search space, particles began re-entering the feasible region after approximately 3000 iterations.

4.2.12 Particle Swarm Optimization with Individual Coefficient Adjustment

Wu and Zhou [112] employed parameters of $\omega_a = 0.9$, $\omega_b = 0.45$, $c_{2a} = 0.5$, $c_{2b} = 2.5$, and $c_1 = 2.05$ in the PSO-SAIC algorithm. It follows from Equation (3.20) that

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

and from Equation (3.21) 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 never lead to theoretical stability.

In line with the theoretical expectations, the average particle movement over time, presented in Figure 4.1l, immediately reached the maximum value and never decreased. This was a result of the inability to generate parameter configurations adhering to the stability criterion, which was empirically evidenced in Figure 4.2l. Due to the rapid divergence, invalid fitness values (which exceeded the limitation of an IEEE-754 64-bit floating point value) were obtained, thereby causing Equation (3.19) to fail at producing feasible values for the related distance. Moreover, this resulted in invalid values calculated for the average change in parameter values, causing Figure 4.3l to appear empty. As should be expected, all particles immediately exited the feasible region and never returned, as shown in Figure 4.4l. Thus, the adaptation mechanism of the PSO-SAIC algorithm is flawed in such an extreme manner that the entire algorithm failed to even complete the search.

4.2.13 Particle swarm Optimization with Random Acceleration Coefficients

As a baseline, the PSO-RAC algorithm, which employs random, theoretically stable values for the control parameters, was used as a control to ascertain how the various

SAPSO algorithms compared to a solely random parameter maintenance strategy. Note that by definition, the PSO-RAC algorithm will always exhibit stable behaviour but should, in theory, demonstrate relatively large parameter changes.

Figures 4.1m and 4.2m, which plot the average particle movement and percentage of particles with theoretically stable parameters over time, depict that the PSO-RAC algorithm exhibited non-divergent behaviour. Figure 4.3m shows that the average parameter movement was approximately 1.6 at each iteration. Given that this strategy randomly selected the parameters each iteration, a reasonable SAPSO strategy should (ideally) exhibit average parameter movement values below 1.6. Finally, Figure 4.4m plots the number of particles with a bound violation over time, which demonstrates that after 5000 iterations, more than 40% of the particles were 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 explicit 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 Equation (4.1) is highly irregular and was largely meant to examine particle stability 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 situations when a more regular search space is employed.

4.2.14 Particle Swarm Optimization with Time-Varying Acceleration Coefficients

Parameters for the PSO-TVAC algorithm were set to the commonly employed values of $\omega_i = 0.9, \omega_f = 0.4$, $c_{1s} = 2.5, c_{1f} = 0.5$, 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 Equation (2.5) that

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

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

As Figure 4.1n depicts, the PSO-TVAC algorithm exhibited initial divergent behaviour, but demonstrated a rapid decrease in particle movement slightly after 1000 iterations had passed. The rapid decrease was caused by the immediate switch from parameters not adhering to the stability criterion to configurations that did adhere to the criterion, as evidenced in Figure 4.2n. Given the linear nature of the parameter adaptation mechanism, the average change in parameters, shown in Figure 4.3n, was constant at 5.74E-4. Finally, the percentage of particles with bound violations, presented in Figure 4.4n, demonstrates that the initial explosive movements caused particles to initially exit the feasible search space. However, after roughly 1000 iterations (and coinciding with the switch to theoretically stable parameter configurations), the PSO-TVAC algorithm depicted 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) were outside the feasible region, on average. Thus, despite not being a truly self-adaptive algorithm, the PSO-TVAC algorithm demonstrated good search behaviours relative to the other algorithms.

4.2.15 Particle Swarm Optimization with Simulated Annealing

Jun and Jian [55] used parameter values of $\omega_a = 0.9, \omega_b = 0.45$, $c_{2a} = 0.5, c_{2b} = 2.5$, and $c_{1s} = 2.5, c_{1f} = 0.5$. From Eqs. (3.30b) and (3.31b), it is clear that there are five distinct scenarios which can occur:

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

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 these cases satisfy the stability criterion outlined in Equation (2.5). Therefore, particle stability 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 Equation (2.5) that $c_1(t) < 1.64853$ is necessary for particle stability. This can only occur when $\frac{t}{T} = 0.42574$, i.e., after roughly 42.6% of the search is completed. Therefore, case (3) can only lead to stability 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 Equation (2.5) that $c_1(t) < 2.08378$ is necessary for stability to be attained. This can only occur when $\frac{t}{T} = 0.20811$, or after roughly 20.8% of the search has completed. Therefore, case (4) can only lead to stability after 20.8% of the search is completed.

Summary: During the first 20.8% of the search process, the PSO-ICSA strictly can not satisfy the stability criterion. Moreover, there are only two (unlikely) scenarios that can lead to the stability criterion being 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 that is neither too close nor too far from the fitness of the global best position. Given the extremely strict requirements for stability, the PSO-ICSA algorithm is expected to exhibit overall divergent behaviour.

Figure 4.1o presents the average particle movement over time for the PSO-ICSA algorithm and clearly demonstrates that stable behaviour was never exhibited. Furthermore, the percentage of particles with theoretically stable parameters shown in Figure 4.2o was never above 0%, indicating that the adaptation mechanism completely failed to produce parameters adhering to the stability criterion. Moreover, the average change in parameter values over time, shown in Figure 4.3o, was constant and had a value of 4.00E-4, indicating that the only change in parameters was due to the linearly decreasing cognitive coefficient. Given the observed divergent behaviour, it was not unexpected that the bound violation percentage, shown in Figure 4.4o, immediately reached 100% and never decreased. Thus, the adaptation mechanism of the PSO-ICSA is extremely flawed and caused immediate divergence.

4.2.16 Adaptive Particle Swarm Optimization

Zhan *et al.* [120] 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 2.0 [120], which leads to the condition of

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

for stability to be exhibited. Assuming that $c_1 + c_2 = 4.0$ (note the conditions for stability become easier to satisfy if $c_1 + c_2 < 4.0$), then $\omega(t)$ will be within the theoretical stable range when $-0.11065 < f_e(t) < 0.15595$. Revisiting Equation (3.32), it is expected that the global best should, in general, be near the centre of the swarm, thereby causing $d_g(t) \approx d_{min}(t)$. When $d_g(t) \approx d_{min}(t)$, the value of $f_e(t)$ should tend towards 0. Given that, in the strictest case, $f_e(t) < 0.15595$ is necessary for stability, the APSO-ZZLC algorithm should exhibit overall stable behaviour. However, given that $c_1 + c_2$ will tend towards 4.0, the particle movement sizes are expected to be rather large [19].

Directly in line with the theoretical expectations, Figure 4.1p depicts an average particle movement that was just slightly below Δ_{max} over the entirety of the search. Furthermore, Figure 4.2p depicts that the generated parameter configurations always adhered to the stability criterion given by Equation (2.5). As Figure 4.3p depicts, the parameter configurations depicted only slight adaptations, and were only changing by an average of 4.51E-5 after 5000 iterations. As such, the APSO-ZZLC algorithm could be naively classified as stable. However, as Figure 4.4p depicts, nearly all particles exited the feasible region and never returned. Thus, despite having parameters adhering to the stability criterion at all times, the APSO-ZZLC algorithm exhibited such large particle movement values that it was incapable of retaining particles within the feasible search space.

4.2.17 Adventurous Unified Particle Swarm Optimization

Hashemi and Meybodi [46, 47] 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 Equations (3.37) and (3.38), the respective automata will learn the values of each parameter that 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 stability given that each of the parameter values is selected independently. However, parameters that exhibit divergence will likely not demonstrate successful behaviour and thus will eventually be given smaller probabilities of selection, thereby promoting stable behaviours. Thus, the UAPSO-A is expected to exhibit overall stability.

Figure 4.1q shows the average particle movement over time for the UAPSO-A algorithm. This figure depicts relatively small initial movement values of approximately 500, which then gradually decreased over time. The average percentage of particles with parameter configurations adhering to the stability criterion, as shown in Figure 4.2q, fluctuated wildly between 60% and 80%. These results indicate that the UAPSO-A algorithm was able to retain overall stable behaviour, despite having between 20% and 40% of the particles exhibiting divergent tendencies at any given iteration. This suggests that, even when particles did exhibit divergent behaviour, parameter configurations were subsequently adapted such that the divergent behaviour did not persist for an extended period of time. However, the significant portion of particles with unstable behaviours at any given time suggests that the divergent parameter configurations were never completely eliminated from consideration. Given the probabilistic nature of parameter selection, the relatively high average change in parameters, shown in Figure 4.3q, was to be expected. Finally, the average percentage of bound violations is presented in Figure 4.4q, where the UAPSO-A demonstrated a smooth decrease in violations over time. After 5000 iterations, the UAPSO-A algorithm had only 38.1% of the particles outside the search space, on average, which is relatively low compared to the other examined algorithms.

4.2.18 Grey Particle Swarm Optimization

Leu and Yeh [64] 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 stability. By substituting the aforementioned parameters into Equation (3.41), the condition for stable behaviour simplifies to

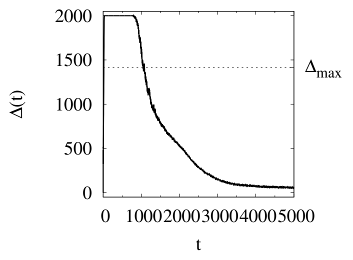
$$\frac{1}{3} \leq \frac{0.9g_{max}(t) - 0.4g_{min}(t) - 0.5g_i(t)}{g_{max}(t) - g_{min}(t)} \leq \frac{1}{2},$$

which will always lead to $\omega_i(t) = 0.9$, and thereby divergent behaviour, in at least one particle, namely the particle i where $g_i(t) = g_{min}(t)$. Moreover, note that the only inertia weight values leading to stability that can be produced by Equation (3.41) are those within the range $[0.4, 0.5]$, and that this range accounts for only 20% of the possible inertia values that Equation (3.41) can produce. Thus, if the inertia weights calculated by Equation (3.41) follow a uniform distribution over the range of $[0.4, 0.9]$, it is expected that only approximately 20% of the particles will exhibit stability during each iteration. Therefore, the GPSO algorithm is expected to exhibit purely divergent behaviour.

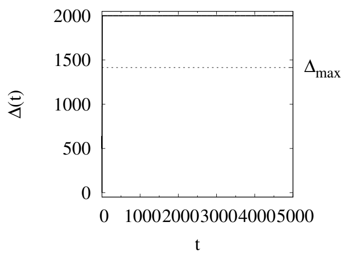
Figure 4.1r presents the average particle movement over time for the GPSO algorithm and empirically confirms that the algorithm demonstrated purely divergent behaviour. Considering Figure 4.2r, it was observed that slightly fewer than 20% of the particles exhibited stability during each iteration, which is directly in line with the theoretical expectation under the assumption that Equation (3.41) uniformly distributes the inertia value. As Figure 4.3r depicts, the average change in parameters was approximately 0.25 initially, but gradually decreased 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 were continually decreasing, and therefore may have lead to a stable set of parameters the number of iterations was larger. However, a stable set of parameters will likely not be of benefit to the algorithm given that the entire swarm exited the feasible region and never returned, as evidenced by Figure 4.4r.

4.3 Summary

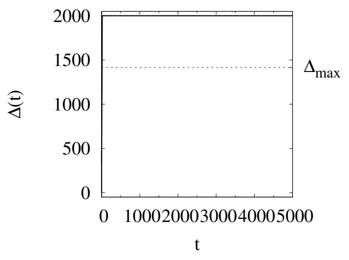
This section provides a summary of the empirical results presented in this chapter. Table 4.1 summarizes the algorithmic parameters employed, while Table 4.2 provides an overview of the final measure values obtained by the examined algorithms, in contrast



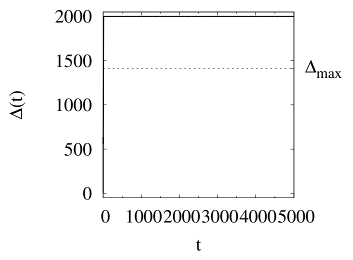
(a) PSO-LDIW



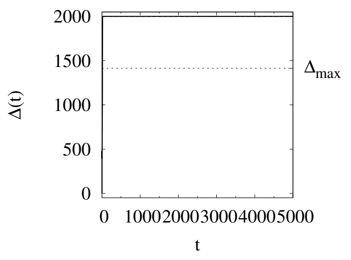
(b) PSO-AIWF



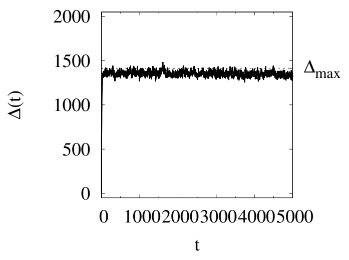
(c) DAPSO



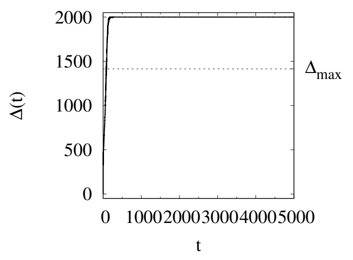
(d) IPSO-LT



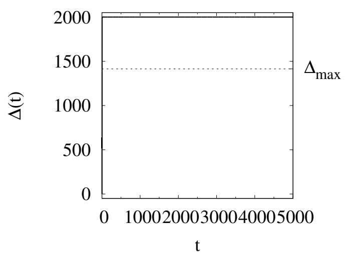
(e) SAPSO-LFZ



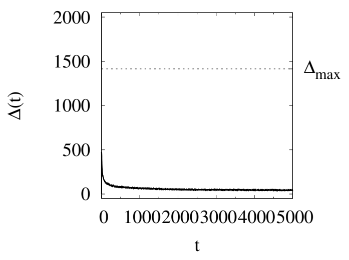
(f) SAPSO-DWCY



(g) PSO-RBI



(h) IPSO-CLL



(i) AIWPSO

Figure 4.1: Average particle movement over time (part I: a–i).

to the standard PSO. Examining Table 4.2, 10 of the 18 examined algorithms (not accounting for the standard PSO) attained average particle movement magnitudes that hit the maximum (capped) value of 2000. 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 parameters adhering to the stability criterion, indicating a complete failure of their respective adaptation mechanisms. Furthermore, only an additional two algorithms had fewer than 50% of the

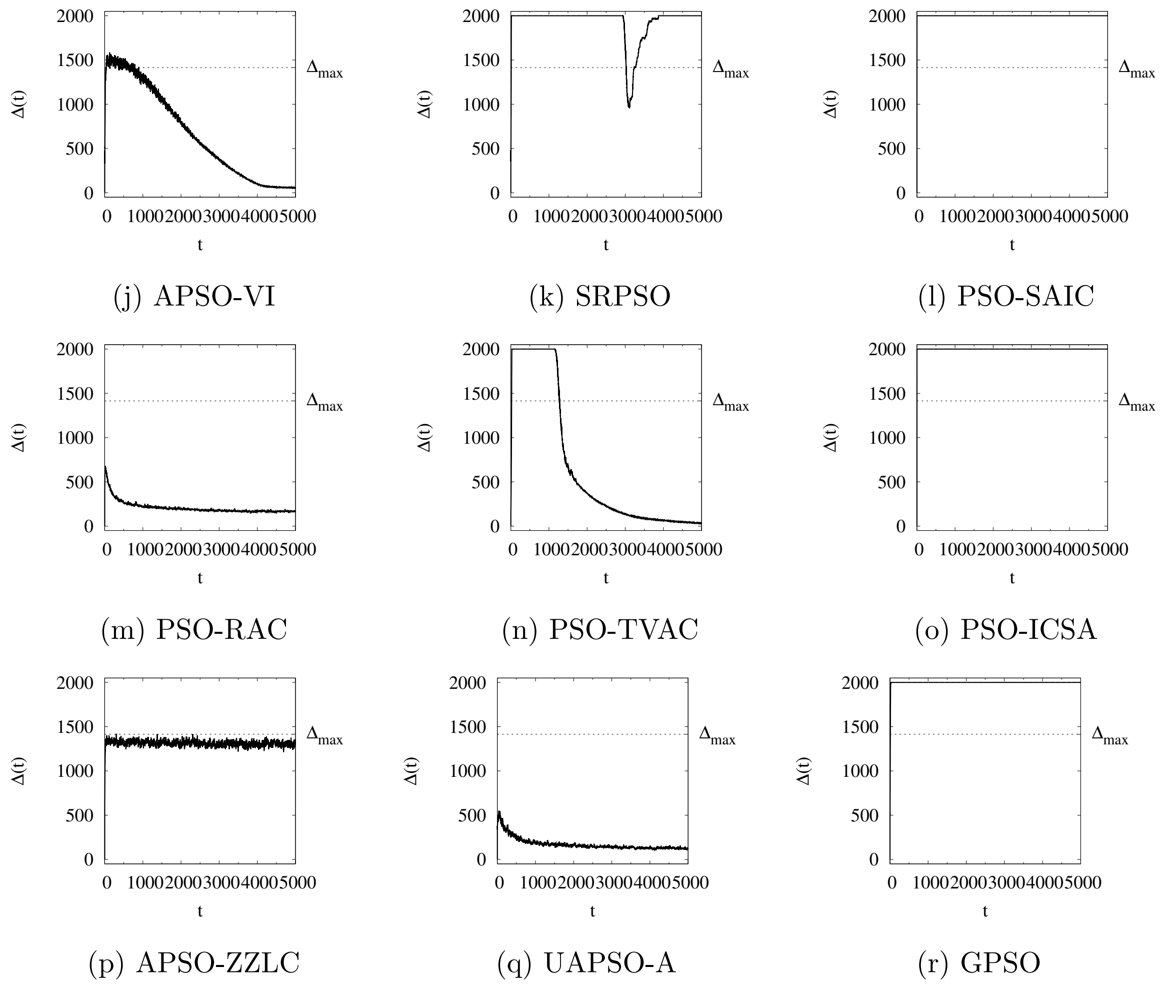
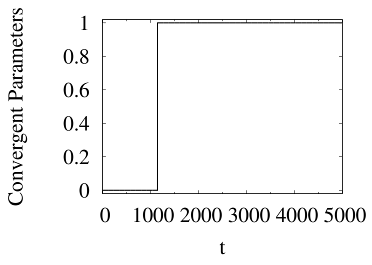
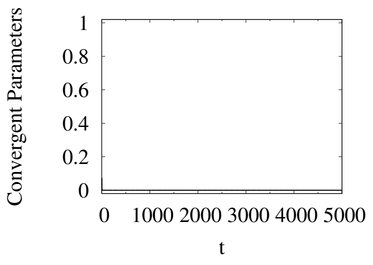


Figure 4.1: Average particle movement over time (part II: j–r).

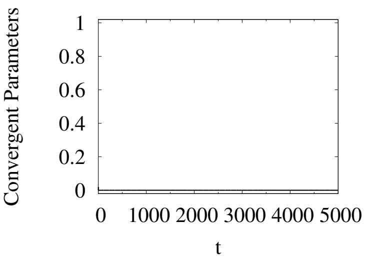
particles with theoretically stable parameters. Of the 18 algorithms, only eight managed to result in stable parameters being employed by all particles. Regarding the average parameter movement, eight of the algorithms depicted no change in parameters during the final iteration, while a majority depicted 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 that are outside the feasible search



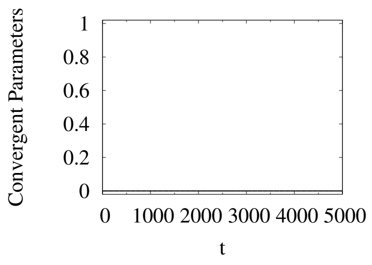
(a) PSO-LDIW



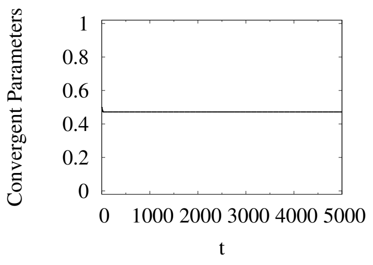
(b) PSO-AIWF



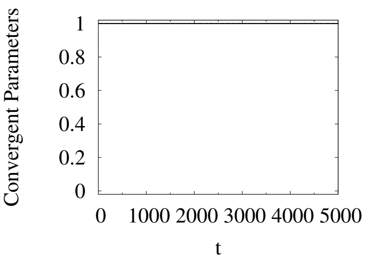
(c) DAPSO



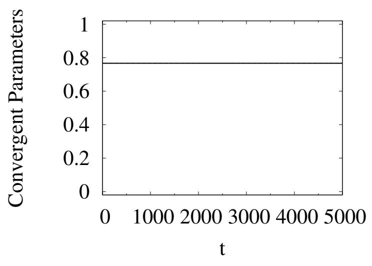
(d) IPSO-LT



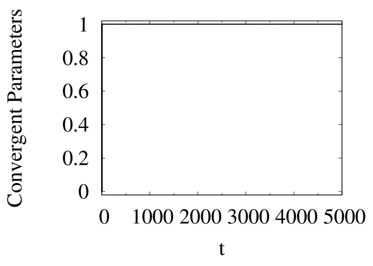
(e) SAPSO-LFZ



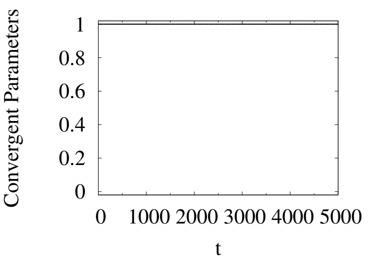
(f) SAPSO-DWCY



(g) PSO-RBI



(h) IPSO-CLL

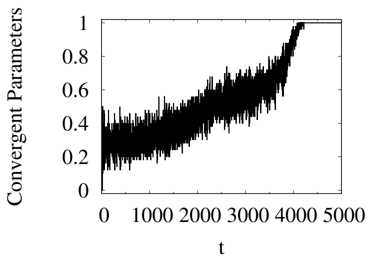


(i) AIWPSO

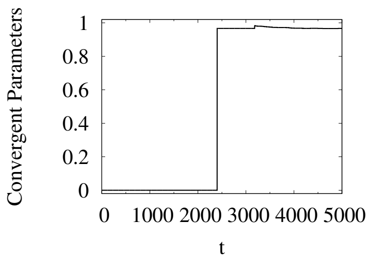
Figure 4.2: Percentage of particles with theoretically stable control parameters (part I: a–i).

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 standard 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.

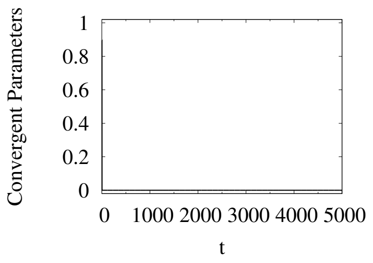
A further noteworthy observation was that care must be taken with regards to the classification of algorithms as exhibiting stability based solely upon the particle move-



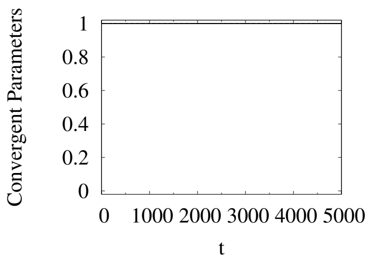
(j) APSO-VI



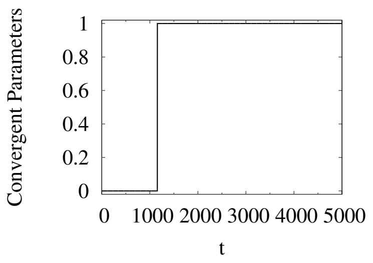
(k) SRPSO



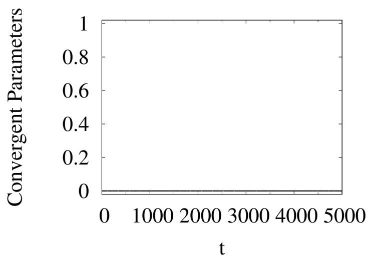
(l) PSO-SAIC



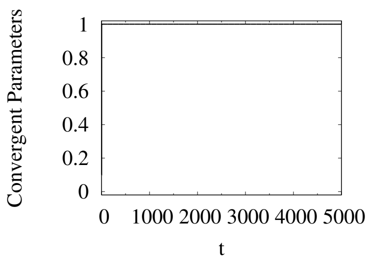
(m) PSO-RAC



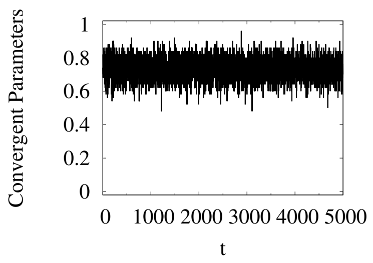
(n) PSO-TVAC



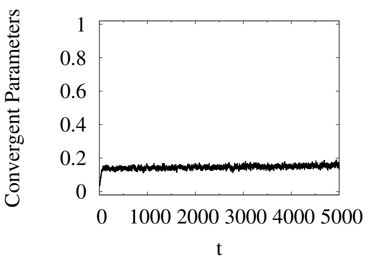
(o) PSO-ICSA



(p) APSO-ZZLC



(q) UAPSO-A

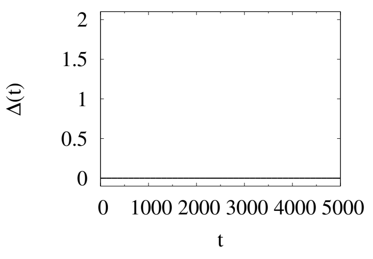


(r) GPSO

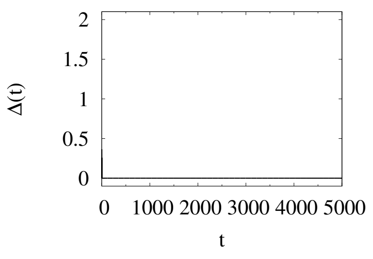
Figure 4.2: Percentage of particles with theoretically stable control parameters (part II: j–r).

ment sizes. As evidenced by the SAPSO-DWCY algorithm in Section 4.2.6, 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 [19], which indicated that parameters near the boundaries of the region defined by Equation (2.5) may exhibit large particle movement values coupled with unreasonably slow convergence, despite being classified as stable.

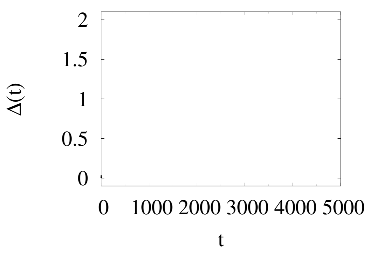
Given that many of the SAPSO techniques focus solely on adapting the inertia weight, the next chapter provides an extensive analysis of both stability and performance for inertia weight control strategies.



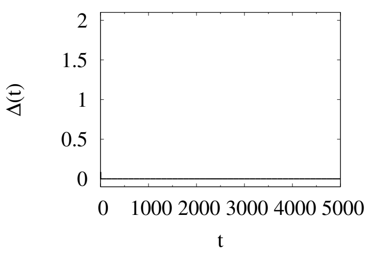
(a) PSO-LDIW



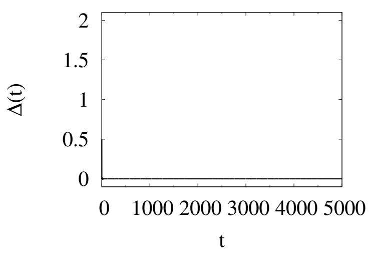
(b) PSO-AIWF



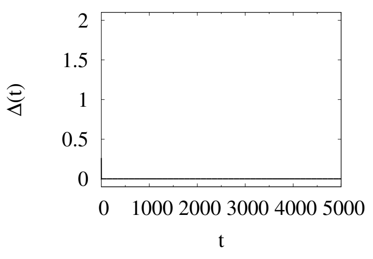
(c) DAPSO



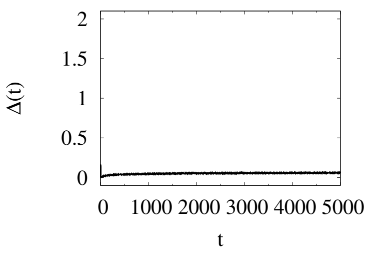
(d) IPSO-LT



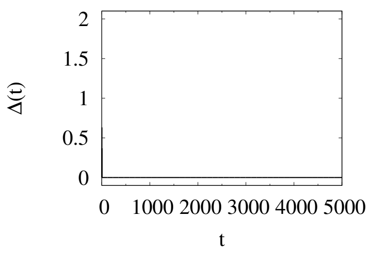
(e) SAPSO-LFZ



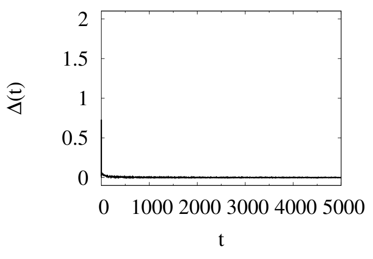
(f) SAPSO-DWCY



(g) PSO-RBI

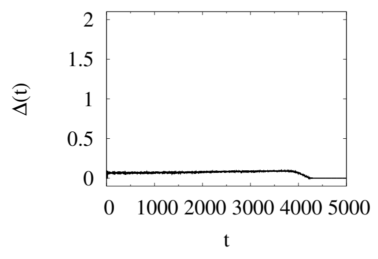


(h) IPSO-CLL

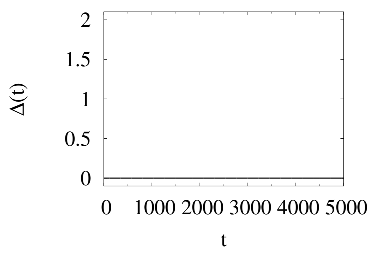


(i) AIWPSO

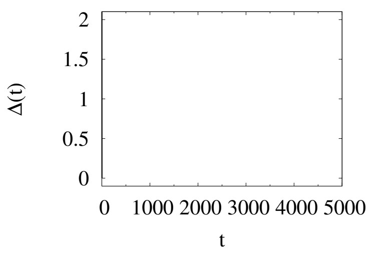
Figure 4.3: Average parameter movement over time (part I: a–i).



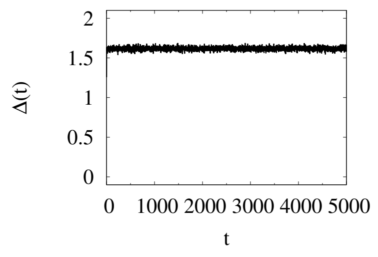
(j) APSO-VI



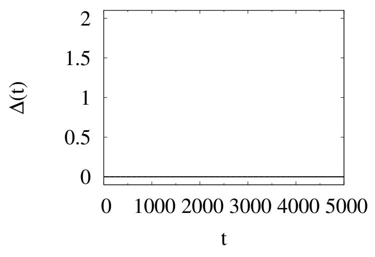
(k) SRPSO



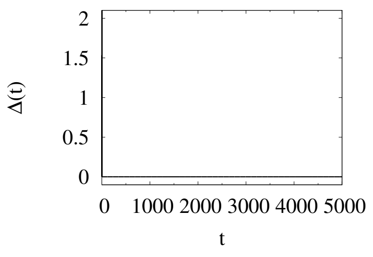
(l) PSO-SAIC



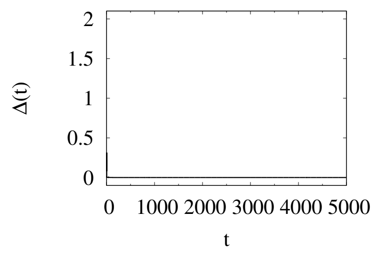
(m) PSO-RAC



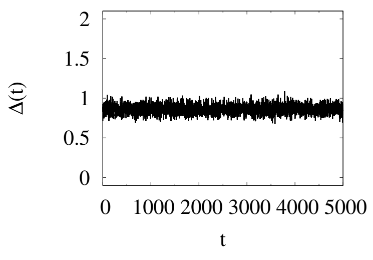
(n) PSO-TVAC



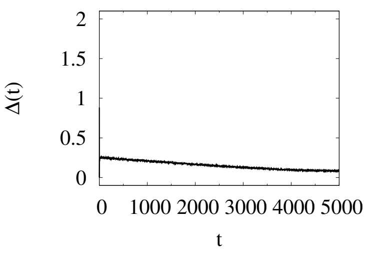
(o) PSO-ICSA



(p) APSO-ZZLC

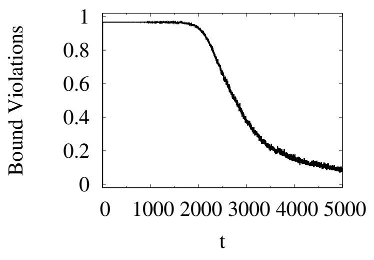


(q) UAPSO-A

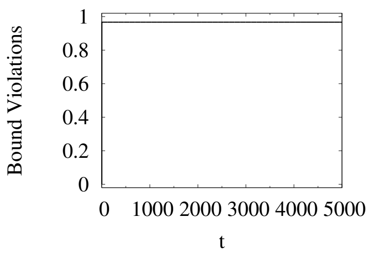


(r) GPSO

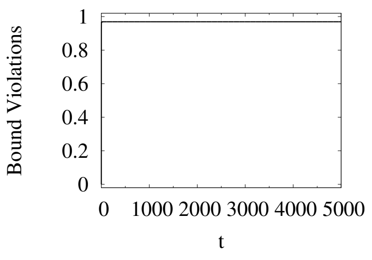
Figure 4.3: Average parameter movement over time (part II: j–r).



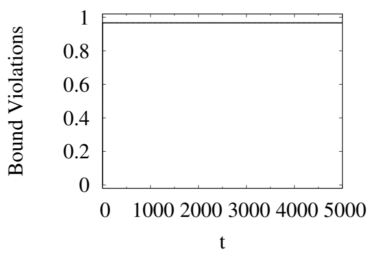
(a) PSO-LDIW



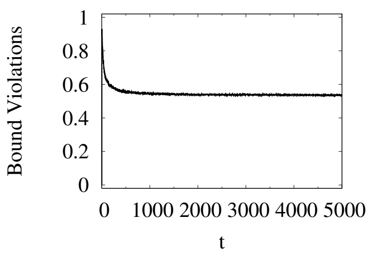
(b) PSO-AIWF



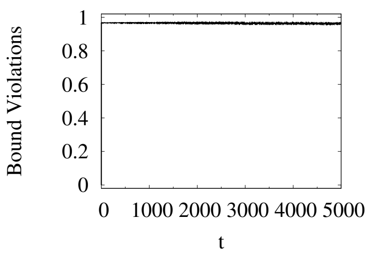
(c) DAPSO



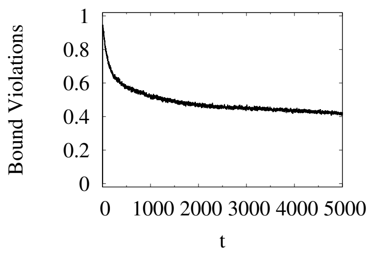
(d) IPSO-LT



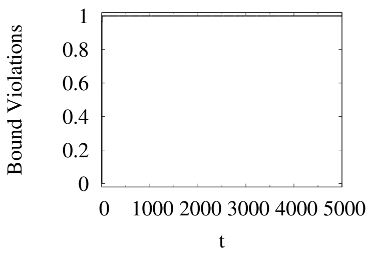
(e) SAPSO-LFZ



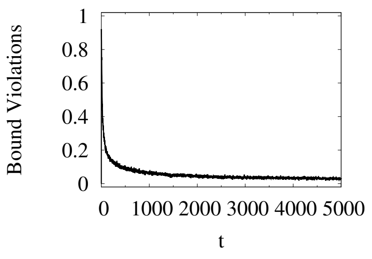
(f) SAPSO-DWCY



(g) PSO-RBI

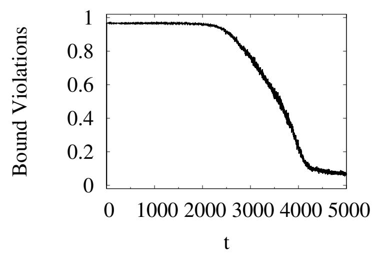


(h) IPSO-CLL

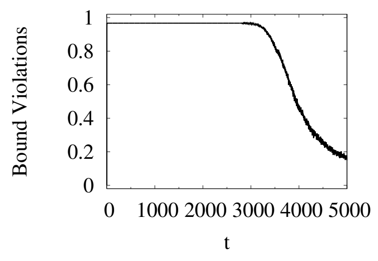


(i) AIWPSO

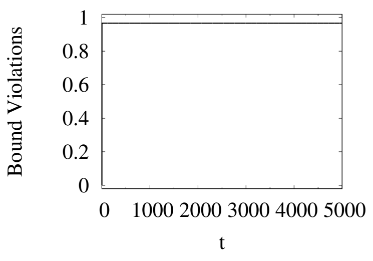
Figure 4.4: Percentage of particles with a bound violation in at least one dimension (part I: a–i).



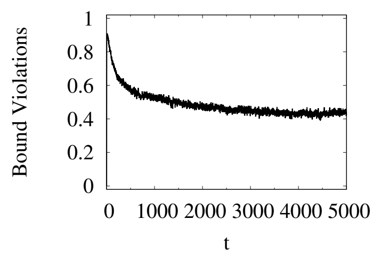
(j) APSO-VI



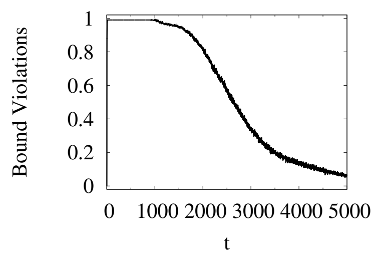
(k) SRPSO



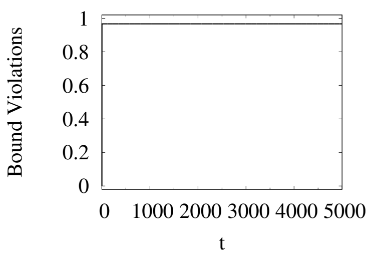
(l) PSO-SAIC



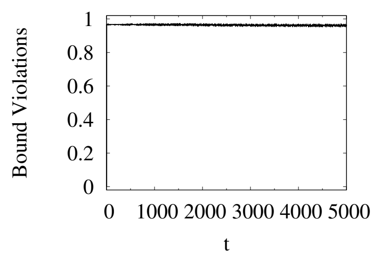
(m) PSO-RAC



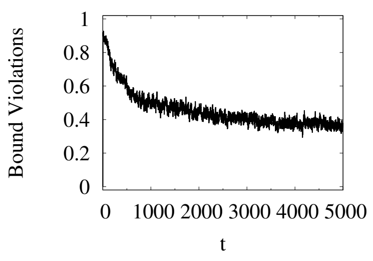
(n) PSO-TVAC



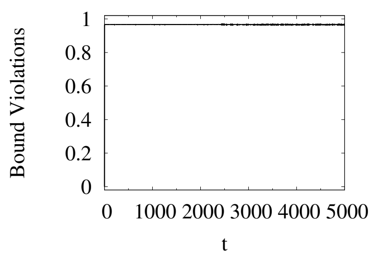
(o) PSO-ICSA



(p) APSO-ZZLC



(q) UAPSO-A



(r) GPSO

Figure 4.4: Percentage of particles with a bound violation in at least one dimension (part II: j–r).

Table 4.1: Control parameter values employed by the SAPSO algorithms.

Algorithm	Parameters
PSO	$\omega = 0.729844, c_1 = c_2 = 1.496180$
PSO-LDIW	$\omega_s = 0.9, \omega_f = 0.4, c_1 = c_2 = 1.496180$
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	$\omega_{min} = 0.4, \omega_{max} = 0.9, C_{min} = 1.5, C_{max} = C_{final} = 2.5, \xi = 1.0$

Table 4.2: Average measure values after 5000 iterations. Δ = average particle movement, SP = stable parameters, Δ_p = average parameter movement, IP = invalid particles. Missing values were too large to compute.

Algorithm	Δ	SP	Δ_p	IP
PSO	415.125	100%	0.0	70.7%
PSO-LDIW	56.489	100%	1.00e-4	9.6%
PSO-AIWF	2000.000	0%	0.0	96.7%
DAPSO	2000.000	0%	–	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%	–	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%

Chapter 5

Analysis of Inertia Weight Control Strategies

As discovered in Chapter 3, a majority of the research on adapting the values of PSO control parameters has focused on adapting the inertia weight. While a number of recent studies have provided reviews of existing inertia weight control strategies [1, 8, 14, 48, 82, 85, 103, 109], such studies typically lack in empirical investigation. The field of inertia weight control strategies is thus left with a plethora of strategies to chose from, with no clear guidance on which strategies perform best. There are also no studies that indicate if or when the various control strategies adhere to the theoretical stability criterion. Furthermore, none of the previous studies have examined the performance relative to fitness landscape features such as modality and separability. To address these shortcomings, this chapter examines 18 inertia weight control strategies, discussed in Chapter 3, and dissects them analytically to determine if and when they will lead to order-2 stability. Moreover, the performance of the strategies is empirically investigated.

The remainder of this chapter is structured as follows. Further motivation for this chapter is provided in Section 5.1. Section 5.2 provides a theoretical analysis of stability for the examined strategies. The experimental procedure is described in Section 5.3, while the results are provided in Section 5.4. Finally, Section 5.5 provides a summary of the findings in this chapter.

5.1 Motivation

Given the variety of inertia weight control strategies in the literature, a comprehensive performance analysis is warranted. However, previous studies that empirically compared multiple inertia weight strategies often only examined a limited set of strategies; Nickabadi *et al.* [82] examined five, Chauhan *et al.* [14] examined only two, Pandey *et al.* [85] examined six, van Zyl and Engelbrecht [109] examined six, Taherkhani and Safabakhsh [103] examined seven, Hu *et al.* [48] did not empirically examine any strategies apart from their proposed strategy, Bonyadi and Michalewicz [8] did not empirically examine any strategies, and Bansal *et al.* [1] examined 15 strategies. Note that, the study of Bansal *et al.* [1] was relatively comprehensive with regards to the inertia weight strategies they examined, but was severely limited by the use of only five benchmark problems.

As Taherkhani and Safabakhsh [103] pointed out, none of the previous works on inertia weight control strategies focused on ensuring that the PSO algorithm retained theoretical stability. Thus, there is a general tendency for adaptive PSO variants to exhibit divergent behaviour [42, 44, 103, 109]. To circumvent divergent behaviour, many authors have employed velocity clamping as a means to limit particle step sizes [13, 14, 16, 33, 37, 70, 86, 104, 109]. As discussed in Section 2.4, velocity clamping does not necessarily prevent divergence, but rather delays it.

The field of inertia weight control strategies is thus left with a plethora of strategies to chose from, with no clear guidance on which strategies perform best. Furthermore, none of the previous studies have examined the performance relative to fitness landscape features such as modality and separability. With the exception of the results shown in Chapter 4, there are no studies that indicate if or when the various control strategies adhere to the theoretical stability criterion. While Chapter 4 provided a theoretical analysis of a number of the algorithms examined in this chapter, the values for the control parameters were taken as those suggested by their respective authors. In contrast, this chapter fixes the values of c_1 and c_2 for all examined algorithms, thereby leading to (slightly) different stability criterion being derived for the algorithms common to both studies.

5.2 Theoretical Stability Analysis

To facilitate a more straightforward derivation of the theoretical stability criterion for the inertia weight strategies, the condition for order-2 stability provided in Equation (2.5) was reformulated as a condition on ω, given by

$$\frac{5C - g(C)}{48} < \omega < \frac{5C + g(C)}{48}, \tag{5.1a}$$

with

$$C = c_1 + c_2 \text{ and } g(C) = \sqrt{25C^2 - 672C + 2304}. \tag{5.1b}$$

In the interest of fairness for the empirical comparison, values for the social and cognitive control parameters have been set to $c_1 = c_2 = 1.496180$ [107] for all examined strategies. It then follows from Equation (5.1) that

$$-0.16199 < \omega(t) < 0.78540 \tag{5.2}$$

is necessary for the algorithms to exhibit order-2 stability. The criteria given by Equation (5.2) was used as the basis for the following stability analyses. Additional parameters for each of the algorithms are given in Table 5.1. Finally, where necessary to complete the analysis, the maximum number of iterations was set as $T = 5000$.

5.2.1 Particle Swarm Optimization with Constant Inertia Weight

For the purposes of this study, a constant inertia weight of $\omega = 0.729844$ was used based on the demonstrated ability to lead to stable trajectories, given the aforementioned values of c_1 and c_2 [19, 28, 107]. Note that the employed value of ω trivially satisfies Equation (5.2) and will thus lead to particle stability.

5.2.2 Particle Swarm Optimization with Random Inertia Weight

Following from Equation (5.2), the PSO-RIW algorithm will exhibit stability when $U(0.5, 1.0) < 0.78540$, which will, in theory, occur 57.08% of the time. Given that

Table 5.1: Parameters employed by the inertia weight control strategies.

Strategy	Equation	Parameters
AIWPSO	(3.35)	$\omega_{min} = 0.0, \omega_{max} = 1.0$
APSO-VI	(3.48)	$\omega_{min} = 0.3, \omega_{max} = 0.9$
CDIW-PSO	(3.7)	$\omega_{min} = 0.4, \omega_{max} = 0.9$
Constant	(2.1)	$\omega = 0.729844$
DE-PSO	(3.45)	$\omega(0) = 0.9$
DW-PSO	(3.6)	–
FG-PSO	(3.36)	$\omega(0) = 0.9$
IPSO-LT	(3.25)	$\alpha = 0.5, \beta = 0.5$
LD-PSO	(3.10)	$\omega_{min} = 0.4, \omega_{max} = 0.9$
PSO-NL	(3.13)	$\omega_{min} = 0.4, \omega_{max} = 0.9, \alpha = 1/\pi^2$
PSO-NLI	(3.9)	$\omega = 0.3, u = 1.0002$
PSO-NEIW	(3.4)	$\omega_{min} = 0.4, \omega_{max} = 0.9$
PSO-OIW	(3.11)	$\omega_{min} = 0.3, \omega_{max} = 0.9, k = 7$
PSO-RIW	(3.2)	–
PSO-RBI	(3.22)	$\omega_{min} = 0.4, \omega_{max} = 0.9$
SRPSO	(3.49)	$\omega_s = 0.9, \omega_f = 0.4, \eta = 1$
PSO-SIW	(3.5)	$s = 2$
PSO-LDIW	(3.1)	$\omega_{min} = 0.4, \omega_{max} = 0.9$

the PSO-RIW algorithm will employ theoretically stable parameters more often than not, it is expected to exhibit stability.

5.2.3 Particle Swarm Optimization with Linearly-Decreasing Inertia Weight

Given that $\forall t, \omega(t) \geq 0.4$, Equation (5.2) simplifies to $\omega(t) < 0.78540$. From Equation (3.1), it follows that $\omega(t) = 0.78540$ occurs when $\frac{t}{T} = 0.22920$, i.e., when 22.9% of the search process has completed. Thus, the PSO-LDIW algorithm is not expected to exhibit stability until nearly a quarter of the search process has completed, but will exhibit stable behaviour overall.

5.2.4 Particle Swarm Optimization with Non-Linear Inertia Coefficient

Despite the initial value of 0.9, the inertia weight rapidly decreases during the beginning of the search and thus $\omega(t) < 0.78540$ occurs after the point where $\frac{t}{T} = 4.8481 \times 10^{-7}$. Therefore, the PSO-NL algorithm will exhibit stability.

5.2.5 Particle Swarm Optimization with Non-Linear Improved Inertia Weight

Jiao *et al.* [53] employed parameters of $u = 1.0002$ and $\omega = 0.3$. Given that $\forall t, 0 < \omega(t) \leq 0.3$, the algorithm will always exhibit stability. However, because of the very low inertia weight, the PSO-NLI algorithm is expected to exhibit premature convergence.

5.2.6 Decreasing Inertia Weight Particle Swarm Optimization

The range of the inertia weight for the DW-PSO algorithm is given by $0 < \omega(t) < 1.23114$. Furthermore, $\omega(t) < 0.78540$ occurs when $t = 4.47432$, due to the rapidly decreasing inertia weight. Therefore, the DW-PSO will exhibit stability after only five iterations have passed.

5.2.7 Chaotic Descending Inertia Weight Particle Swarm Optimization

The CDIW-PSO algorithm will exhibit stability when

$$z(t) < \frac{t + 2854}{4000},$$

assuming that $T = 5000$. Note that the maximum value of $z(t)$ is 1.0, and that

$$\frac{t + 2854}{4000} = 1.0$$

when $t = 1146$, i.e., when $\frac{t}{T} = 0.22920$. Thus, if $z(t)$ is removed from consideration, the stability analysis of the CDIW-PSO algorithm is the same as that of the PSO-LDIW algorithm. However, the stability of the CDIW-PSO algorithm during the initial 22.9% of the search is dependent upon on the value of $z(t)$, whereas PSO-LDIW will never adhere to the stability criterion during this initial phase. Based on the above analysis, the CDIW-PSO algorithm will exhibit overall stable behaviour.

5.2.8 Particle Swarm Optimization with Natural Exponent Inertia Weight

The PSO-NEIW algorithm will exhibit stability when $\frac{t}{T} > 0.02603$, i.e., after approximately 2.6% of the search has completed. Thus, the PSO-NEIW algorithm will exhibit overall stability.

5.2.9 Particle Swarm Optimization with Oscillating Inertia Weight

By rearranging Equation (3.12), it can be shown that the PSO-OIW algorithm will exhibit stable behaviour when

$$\cos\left(\frac{17\pi t}{3750}\right) < 0.61800,$$

which, assuming $T = 5000$, accounts for 2670 of the 3750 iterations during the oscillation phase. Additionally, the final quarter of the search process employs an inertia weight of 0.3, which leads to trivially stable behaviour. Thus, the algorithm is expected to exhibit stability.

5.2.10 Particle Swarm Optimization with Sugeno Inertia Weight

The PSO-SIW algorithm will exhibit stability when

$$\frac{t}{T} > \frac{1073}{2927s + 5000},$$

and thus the stability depends largely on the employed value of s. For the purposes of this study, a value of $s = 2$ was used, indicating the algorithm will exhibit stability when

$$\frac{t}{T} > \frac{1073}{10854} = 0.09886,$$

i.e., after approximately 10% of the search has completed. Despite the initially unstable behaviour due to the initial inertia weight value of 1, the PSO-SIW algorithm is expected to exhibit stable behaviour.

5.2.11 Logarithm Decreasing Inertia Weight Particle Swarm Optimization

Given the parameters in Table 5.1, the LD-PSO algorithm will exhibit stable behaviour when $\frac{t}{T} > 0.02576$, i.e., after only approximately 2.6% of the search has completed. Therefore, the LD-PSO algorithm will exhibit stability.

5.2.12 Self-Regulating Particle Swarm Optimization

Given the parameters in Table 5.1, Equation (3.49b) simplifies to $\Delta\omega = 0.0001$. Note that in Equation (3.49a), the inertia weight increases for the best particle but decreases for all other particles. To continue with the analysis, the assumption that each particle is equally likely to be the best particle, and thereby increases its inertia weight accordingly, is made. Thus, given a swarm size of n_s particles, the inertia weight of each particle will exhibit an inertia weight decrease of $0.0001(n_s - 2)$ every n_s iterations or $\frac{0.0001*(n_s-2)}{n_s}$ each iteration, on average. Substitution of the resulting inertia weights into Equation (5.2), along with an assumed swarm size of 30 particles, leads to the necessary condition of

$$1227.86 < t < 11378.50$$

for stability. This indicates that the SRPSO algorithm should exhibit stability starting at iteration 1228, i.e., after approximately 24.6% of the search. Note that, if the SRPSO algorithm is employed for longer than 11378 iterations without leading to complete stagnation, it will once again exhibit divergent behaviour because the inertia weight will decrease below -0.16199.

5.2.13 Adaptive Particle Swarm Optimization Based on Velocity Information

As the ideal velocity of Xu [113] decreases over time, the employed value of $\omega(t)$ will also decrease over time, and thus will eventually settle within the stable range. Furthermore, given that the ideal velocity tends toward 0, by definition the average velocity will also tend toward 0, regardless of the inertia weight. Therefore, the APSO-VI algorithm is expected to exhibit stable behaviour.

5.2.14 Fine-Grained Inertia Weight Particle Swarm Optimization

Firstly, it is recognized that the distance term will always be positive in the FG-PSO algorithm. As a result,

$$0 < e^{-(d(\hat{\mathbf{y}}(t),\mathbf{y}_i(t))*\frac{t}{T})} \leq 1,$$

and using the substitution

$$c = e^{-(d(\hat{\mathbf{y}}(t),\mathbf{y}_i(t))*\frac{t}{T})},$$

Equation (3.36) can be expressed as

$$\omega_i(t+1) = \omega_i(t) - c(\omega_i(t) - 0.4),$$

where $c \in (0, 1]$. Therefore, as the distance between a particle and the global best decreases, the inertia weight will also decrease to a minimum of 0.4. An important note about the FG-PSO strategy is that the inertia weight never increases, and thus the inertia weight will always tend toward 0.4. This behaviour will likely cause the FG-PSO strategy to exhibit stable behaviour in the long run, as even large distances from the global best will cause a non-zero decrease in the inertia weight, given the asymptotic nature of the exponential term. However, the inertia weight is expected to decrease rather quickly, especially for the global best particle, which will have a distance of 0, and therefore the FG-PSO strategy is expected to suffer from premature convergence.

5.2.15 Double Exponential Self-Adaptive Inertia Weight Particle Swarm Optimization

The first observation about the DE-PSO algorithm is that $R_i(t)$ must be positive. Given that $R_i(t) > 0$, it follows that $0 < e^{-R_i(t)} \leq 1$ and, by using the substitution $c = e^{-R_i(t)}$, Equation (3.45a) can be expressed as

$$\omega_i(t) = e^{-c},$$

with $c \in (0, 1]$. This now provides a definite range for the inertia weight given by

$$0.36788 < \omega_i(t) < 1,$$

where $\omega_i(t)$ is minimized when $R_i(t)$ is 0. Conversely, the DE-PSO algorithm provides larger inertia weights when the distance from the global best is increased. The inertia weight provided by the DE-PSO algorithm is thus expected to be large initially due to the roaming behaviour of unconstrained particles [32], but is expected to decrease over time as particles tend toward the global best. Therefore, the DE-PSO algorithm will exhibit stable behaviour.

5.2.16 Particle Swarm Optimization with Rank-Based Inertia Weight

Assuming a swarm size of 30 particles, it follows from Equation (3.22) that the PSO-RBI algorithm will exhibit stable behaviour when $\frac{R_i(t)}{n_s} \geq 0.22920$. That is, when a particle is ranked within the worst 22.9% of the swarm, the particle will exhibit divergent behaviour. Therefore, more than one-fifth of the swarm will exhibit divergent behaviour at any given time, causing the PSO-RBI algorithm to exhibit overall divergent behaviour.

5.2.17 Adaptive Inertia Weight Particle Swarm Optimization

Given that the parametrization of AIWPSO is the same as seen in Chapter 4, the stability analysis is the same. To summarize the previous findings, the swarm may not be stable when more than 78.54% of the particles improve their personal best fitness in any given iteration. However, it is unreasonable to assume such a high rate of success will be maintained, thus stable behaviour is expected.

5.2.18 Improved Particle Swarm Optimization by Li and Tan

Li and Tan [67] provided no guidance for the selection of the α and β parameter values, thus the mid-point of the allowable range was used, namely $\alpha = \beta = 0.5$.

It then follows that

$$\omega_i(t) = 1 - \left| \frac{1 - c_i(t)}{3(d_i(t) + 1)} \right|,$$

which further simplifies the stability criterion such that

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

is required for stable behaviour to be exhibited. Alternatively, the stability criterion can be reformulated as

$$\begin{cases} 1.55328c - 2.55328 < d < -1.55328c + 0.55328, & \text{if } c < 1 \\ -1.55328c + 0.55328 < d < 1.55328c - 2.55328, & \text{if } c > 1 \end{cases}$$

to provide a more direct view of the relationship between c and d. As the search progresses, it is expected that both $c_i(t)$ and $d_i(t)$ will tend towards 0, and thus the IPSO-LT algorithm should exhibit overall stability.

5.3 Experimental Procedure

To examine the performance of the considered inertia weight control strategies, 50 independent runs were executed for each strategy on each of the 60 benchmark problems, described in Appendix A, using the parameters listed in Table 5.1. Note that the parameters employed in this section were identical to those used for the derivation of stability conditions. Each execution consisted of 5000 iterations and all strategies used an identical base PSO configuration, with the obvious exception of the inertia weight control strategy. The base configuration consisted of 30 particles arranged in a star topology and a synchronous update strategy was used. Particle positions were randomly initialized within the feasible range and velocities were initialized to the zero vector [30]. To ensure that the particle attractors remained inside the feasible region, personal best positions were only updated if a new position of superior fitness was found that was within the feasible search space. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. As with the stability analysis, all algorithms employed acceleration coefficients of $c_1 = c_2 = 1.496180$.

5.3.1 Performance Measures

The inertia weight control strategies were compared with respect to the following performance measures:

- **Accuracy** – the fitness of the global best particle after 5000 iterations.
- **Consistency** – the squared difference between the accuracy of each independent run and the average accuracy obtained across all runs.
- **Success Rate** – the percentage of the 50 independent runs that reached specified accuracy levels. A total of 1000 accuracy levels were considered, with the accuracy levels starting at the best accuracy obtained by all algorithms and logarithmically increasing towards the worst accuracy.

In addition to the aforementioned performance measures, the average particle movement [19, 20] given by Equation (4.2) was also employed. In contrast to the measures employed in Chapter 4, these measures were selected to provide a better focus on performance, rather than overall search behaviour.

5.3.2 Statistical Analysis

For each benchmark problem and performance measure, pairwise two-tailed Mann-Whitney U tests were performed at a significance level of 0.05 to identify performance differences. When the Mann-Whitney U test indicated that a difference existed among two strategies, the median performance measure values were used to assign wins and losses; the better performing strategy was awarded a win, while the inferior strategy was awarded a loss. The strategies were then assigned an overall rank based on the difference between the number of wins and losses, i.e., difference = wins - losses. Additionally, the best rank frequency (BRF), defined as the number of benchmark problems for which the strategy attained the highest rank, was recorded. Finally, the Bonferroni-Dunn post-hoc test was employed to generate critical difference plots that visually depict the average ranks with respect to the solution accuracy. Note that the Mann-Whitney U tests assessed the fine-grained, per-function performance differences while the Bonferroni-Dunn test was used to indicate differences in average rank across an entire set of problems.

5.4 Empirical Results and Discussion

This section presents the results of the experiments described in Section 5.3. Table 5.2 summarizes the results obtained across all benchmark problems using the aforementioned Mann-Whitney U statistical analysis procedure, while Table 5.3 presents the average and standard deviation of the ranks across all benchmark problems. Figure 5.1 presents the critical difference plot, graphically indicating the average rank of each algorithm with respect to the accuracy measure. Algorithms grouped with a line had no significant difference in average rank.

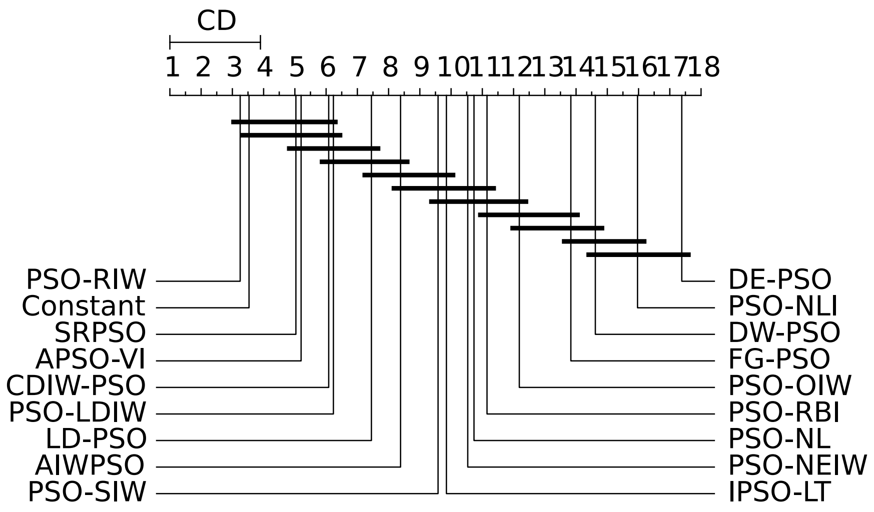


Figure 5.1: Critical difference plot using the Bonferroni-Dunn test on all benchmark problems.

Table 5.2: Summary of performance across all 60 benchmark problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses across all benchmark problems. The maximum possible value for the Diff column is 1020.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	143	8	7	96	7	16	129	7	10
APSO-VI	471	4	15	174	3	17	276	4	17
CDIW-PSO	313	6	5	134	5	14	98	8	8
Constant	630	2	23	**387**	**1**	27	**364**	**1**	22
DE-PSO	-874	18	1	-362	18	14	-322	18	11
DW-PSO	-565	16	1	-247	16	14	-275	15	9
FG-PSO	-534	15	2	-226	15	14	-279	16	9
IPSO-LT	-28	10	1	56	8	14	-160	12	9
LD-PSO	303	7	1	48	9	14	146	6	9
PSO-NL	-221	13	1	-66	12	15	-184	13	7
PSO-NLI	-697	17	1	-292	17	14	-304	17	7
PSO-NEIW	-87	11	1	-51	11	14	-92	11	8
PSO-OIW	-327	14	1	-157	14	14	-245	14	7
PSO-RIW	**637**	**1**	11	349	2	19	352	2	18
PSO-RBI	-97	12	1	-71	13	14	-81	10	7
SRPSO	494	3	3	155	4	14	288	3	10
PSO-SIW	2	9	2	-50	10	15	20	9	10
PSO-LDIW	439	5	3	117	6	14	276	4	9

The first observation from Table 5.2 is that the random inertia weight strategy (i.e., PSO-RIW) attained the best rank for the solution accuracy with a difference score of 637. However, on only 11 (18.3%) of the 60 problems did the random strategy attain the best fitness, while a constant inertia weight strategy attained the best accuracy on 23 (36.7%) problems, and attained a difference score of 630. Furthermore, the APSO-VI strategy demonstrated the best performance on 15 (25.0%) problems, i.e., four more than the random strategy, but ranked fourth overall with a difference score of 471. Of the examined strategies, the DE-PSO algorithm depicted the worst overall accuracy with a difference score of -874. Considering the average ranks over all benchmark functions presented in Table 5.3, the constant and random strategies attained average ranks of 3.000 and 3.033 with standard deviations (SDs) of 2.456 and 2.484, respectively, and therefore demonstrated nearly indistinguishable accuracy overall. This is further evidenced by the critical difference plot in Figure 5.1, which indicated that the average rank was not significantly different among the top six algorithms.

Table 5.3: Average rank and standard deviation (SD) across all benchmark problems.

	Accuracy		Success Rate		Consistency	
Strategy	Average	SD	Average	SD	Average	SD
AIWPSO	7.700	4.806	5.667	5.059	7.000	5.042
APSO-VI	5.183	4.094	3.923	3.542	5.183	4.678
CDIW-PSO	5.767	2.375	4.256	3.126	7.050	4.272
Constant	3.000	2.456	2.103	2.245	4.900	5.184
DE-PSO	17.200	2.910	11.795	8.192	11.267	6.996
DW-PSO	14.467	2.807	10.154	7.066	10.700	6.091
FG-PSO	13.783	3.683	9.795	6.921	10.683	5.685
IPSO-LT	9.317	4.710	5.179	3.684	10.133	6.074
LD-PSO	6.950	2.896	4.949	3.783	6.317	4.164
PSO-NL	11.183	2.920	7.436	5.510	10.133	4.918
PSO-NLI	15.733	3.399	10.846	7.524	11.333	6.337
PSO-NEIW	10.350	2.193	7.205	4.964	9.600	4.439
PSO-OIW	12.083	3.614	8.282	6.017	10.750	5.389
PSO-RIW	3.033	2.484	2.154	1.565	3.850	4.149
PSO-RBI	10.833	2.631	7.282	5.477	9.317	4.579
SRPSO	4.917	3.055	3.923	3.467	4.817	3.864
PSO-SIW	9.483	2.534	6.615	4.881	7.600	4.203
PSO-LDIW	6.117	3.669	4.513	4.322	4.900	3.516

These results clearly indicate that when considering solution accuracy for an arbitrary problem, a constant or random inertia weight is preferable. Furthermore, the results show that none of the adaptive inertia weight strategies performed as well as the simpler constant and random strategies when considering the fine-grained, per-function analysis of the Mann-Whitney U tests. However, when the overall average rank was considered via the Bonferroni-Dunn test, far less discrepancies in accuracy were noted.

When considering the success rate, the constant inertia weight strategy attained both the best overall rank, with a difference score of 387, and the largest number of problems for which it obtained the best performance, with 27 (45.0%) of such problems. As with the accuracy measure, the random strategy was very close in performance to the constant strategy, obtaining the second highest rank, with a difference score of 349, and the best performance on 19 (31.7%) of the problems. A noteworthy observation is that the difference scores, relative to the success rate, were significantly lower than with the accuracy measure, indicating that there were less discrepancies in performance, i.e., the various strategies had a more similar level of performance. When considering the average ranks, the constant strategy attained the best average rank of 2.103 with a standard deviation of 2.245, while the random strategy had a slightly higher average rank of 2.154 coupled with a lower standard deviation of 1.565. This indicates that the random strategy was much more consistent in terms of the success rate. The worst overall success rate was observed when using the DE-PSO strategy, with a difference score of -382.

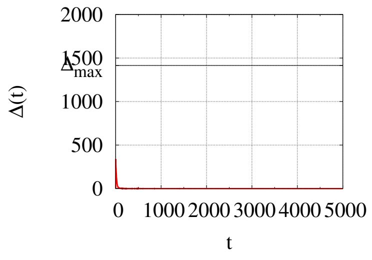
In terms of consistency, the constant inertia weight strategy had the lowest overall deviations from the average accuracy, with a difference score of 364, while the second best strategy, the random strategy, attained a difference score of 352. Furthermore, the constant strategy had the best consistency on 22 (36.7%) of the problems, while the random strategy was most consistent on 18 (30.0%). Despite having better performance overall, the average rank of the constant strategy, 4.900 with a SD of 5.184, was significantly higher than the average rank of the random strategy, at 3.850 with a SD of 4.149. As with the previous two measures, the DE-PSO strategy showed the worst overall performance with a difference score of -322.

To provide empirical evidence in support of the stability analyses, Figures 5.2 and 5.3 depict the average particle movement values over time on two problem instances, namely f_1 and f_{17}, which are representative of the overall behaviour of each algorithm.

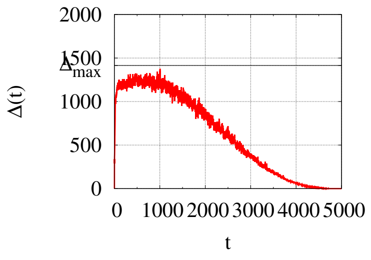
An immediate observation was that the only algorithm to exhibit an average particle movement above Δ_{max} after 5000 iterations was the PSO-RBI algorithm. Recall that, of all the examined strategies, PSO-RBI was the only strategy that was not expected to exhibit stable behaviour. Furthermore, recall that there were three algorithms expected to demonstrate immediate convergence of particle positions, namely the PSO-NLI, DW-PSO, and FG-PSO algorithms. The average particle step sizes for these algorithms are shown in Figures 5.2j and 5.3j for PSO-NLI, Figures 5.2f and 5.3f for DW-PSO, and Figures 5.2g and 5.3g for FG-PSO – each of which indicates nearly stagnant particles after only a few iterations. Additionally, recall that both PSO-NEIW and LD-PSO were expected to exhibit stable behaviour after approximately 2.6% of the search had completed. Considering Figures 5.2l, 5.2i, 5.3l, and 5.3i, which visualize the particle step sizes for the PSO-NEIW and LD-PSO algorithms, it is evidenced that both algorithms exhibited divergent behaviour initially, but depicted rapid decreases in particle movement very early in the search.

Further examination of the particle movement plots in Figures 5.2 and 5.3 highlighted a correlation between particle movement and performance. Specifically, the six worst performing algorithms, namely DE-PSO, PSO-NLI, DW-PSO, FG-PSO, PSO-OIW, and PSO-NL, each demonstrated prohibitively low particle movement values after only a few iterations. Thus, the worst performing strategies exhibited extremely premature convergence, and thereby had insufficient particle movement to perform an effective search. Conversely, the two best performing strategies, namely the PSO-RIW and constant strategies, maintained a slightly higher level of particle activity than the worst performing strategies. However, both the PSO-RIW and constant strategies depicted higher initial levels of movement followed by a noticeably more gradual decline. Finally, the mid-performing algorithms, e.g., LD-PSO, typically depicted large initial movements, suggesting initially divergent behaviour, followed by a rapid decline in movement levels, suggesting stagnation. It is thus concluded that, for the mid-performing strategies, the particles rapidly move outwards from their starting locations, finding moderately good solutions along the way, but eventually stagnate without locating any further promising areas.

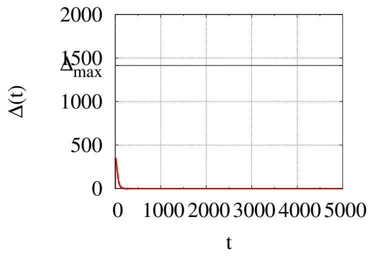
In summary, the constant and random (PSO-RIW) strategies attained the best ranks



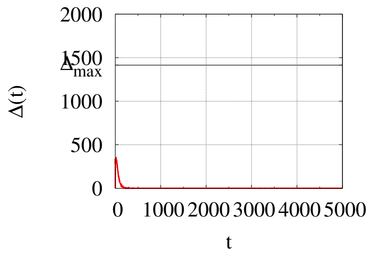
(a) AIWPSO



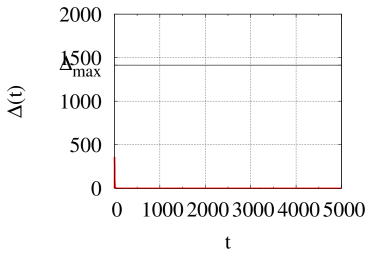
(b) APSO-VI



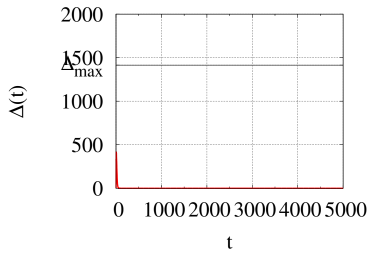
(c) CDIW-PSO



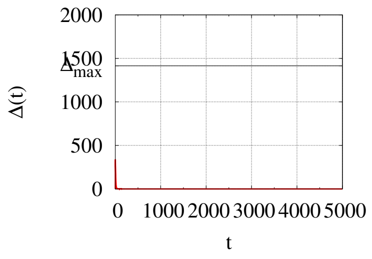
(d) Constant



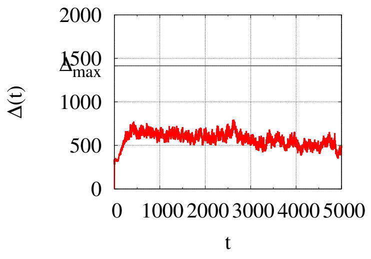
(e) DE-PSO



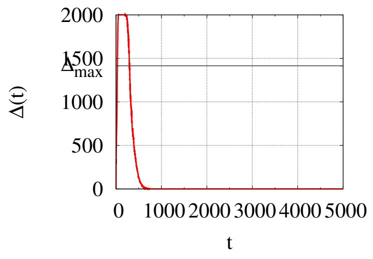
(f) DW-PSO



(g) FG-PSO



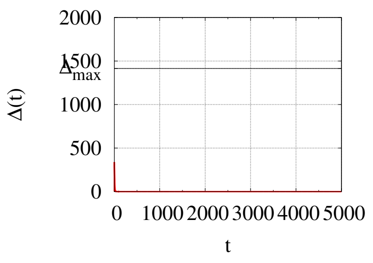
(h) IPSO-LT



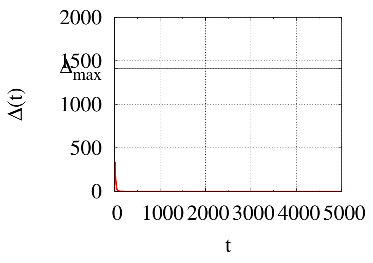
(i) LD-PSO

Figure 5.2: Average particle movement on function f_1 with $\Delta_{max} = 1414.214$ (part I: a–i).

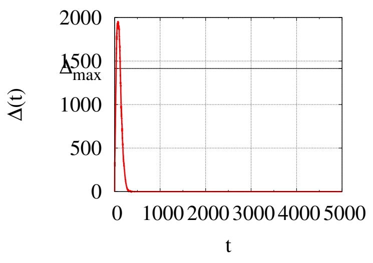
for all performance measures when aggregated across all 60 benchmark problems. Moreover, the constant and random control strategies performed relatively similar for all performance measures, and both showed a significant improvement over the next (third) best strategy. Therefore, it can be concluded that, when faced with the optimization of an arbitrary function, the use of an inertia weight control strategy is not recommended because the constant and random strategies both lead to consistently better results. Furthermore, the results presented here highlight the drastic misrepresentations of the various algorithms' performance in the literature, which stems from the limited analyses when they were proposed. Finally, an examination of the average particle movement over



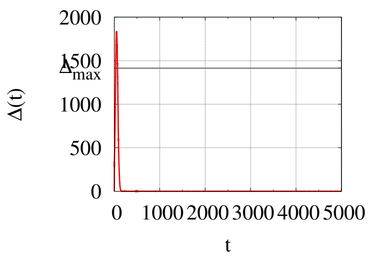
(j) PSO-NLI



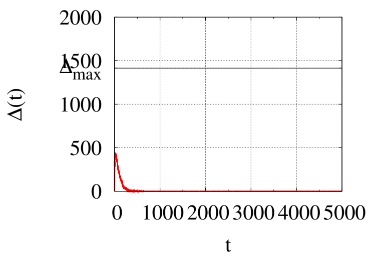
(k) PSO-NL



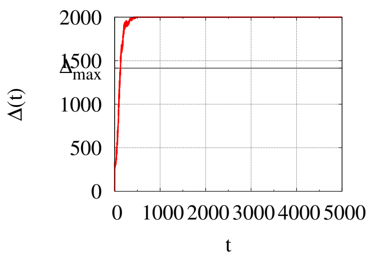
(l) PSO-NEIW



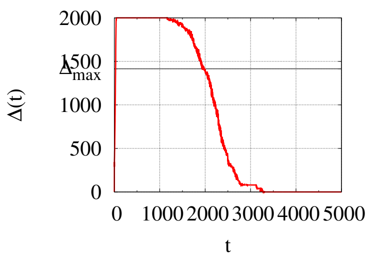
(m) PSO-OIW



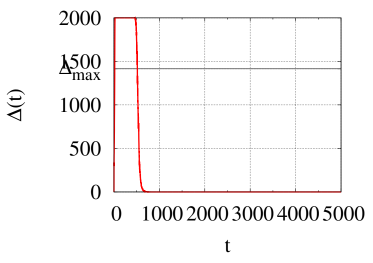
(n) PSO-RIW



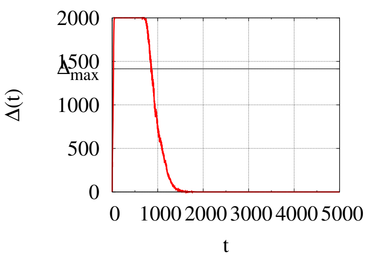
(o) PSO-RBI



(p) SRPSO



(q) PSO-SIW

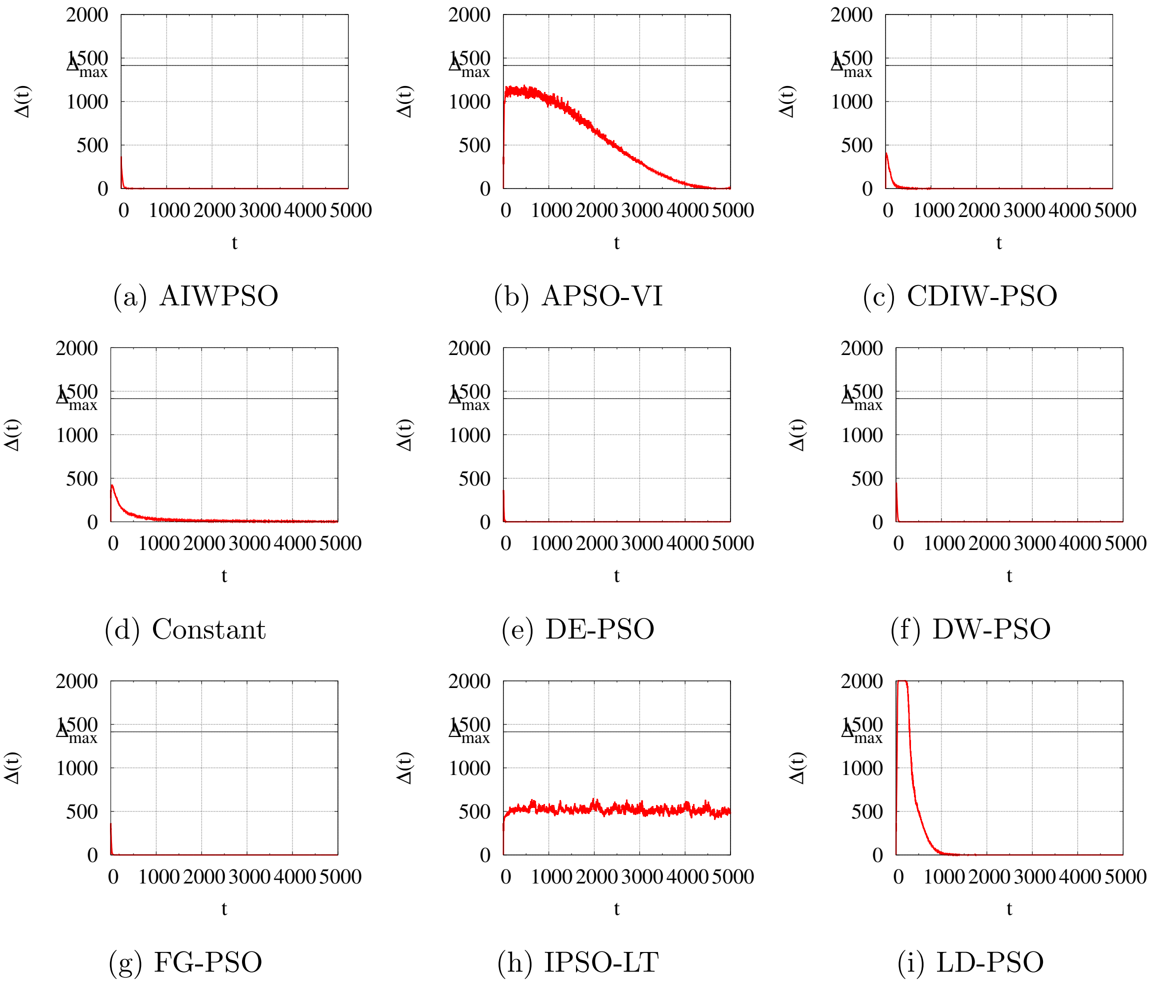


(r) PSO-LDIW

Figure 5.2: Average particle movement on function f_1 with $\Delta_{max} = 1414.214$ (part II: j–r).

time highlighted a correlation between the level of particle movement and performance.

In the following sections, the performance of the inertia weight control strategies is analysed with respect to various characteristics of the benchmark problems. The performance is examined with respect to unimodal, multimodal, separable, non-separable, and composition problems to ascertain whether such characteristics of the problem affect the performance.



(a) AIWPSO (b) APSO-VI (c) CDIW-PSO

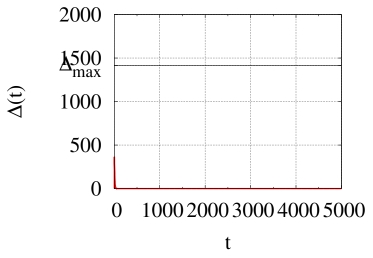
(d) Constant (e) DE-PSO (f) DW-PSO

(g) FG-PSO (h) IPSO-LT (i) LD-PSO

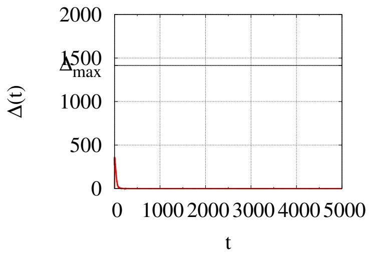
Figure 5.3: Average particle movement on function f_{17} with $\Delta_{max} = 1414.214$ (part I: a–i).

5.4.1 Unimodal Problems

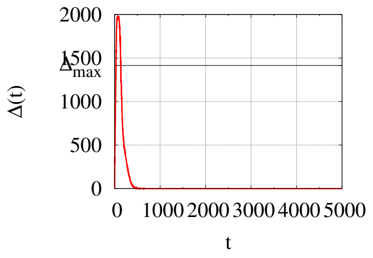
Table 5.4 depicts the overall rank and best rank frequency of the examined strategies across the 19 unimodal problems, while Figure 5.4 shows the critical difference plot for the accuracy measure. For all three performance measures, the constant strategy attained the best rank using the Mann-Whitney U tests, followed by the random strategy. The constant strategy had difference scores of 291, 180, and 255 for the accuracy, success rate, and consistency measures, respectively. However, the critical difference plot in Figure 5.4 depicts insignificant differences in average rank for the accuracy measure across the



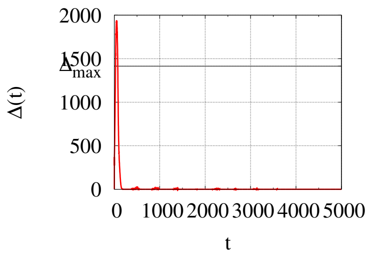
(j) PSO-NLI



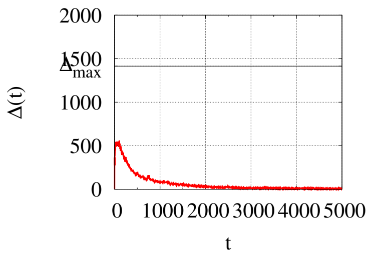
(k) PSO-NL



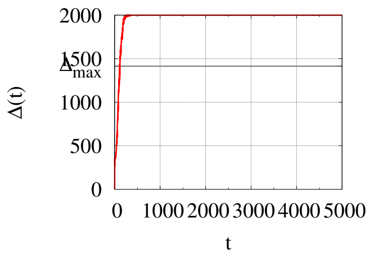
(l) PSO-NEIW



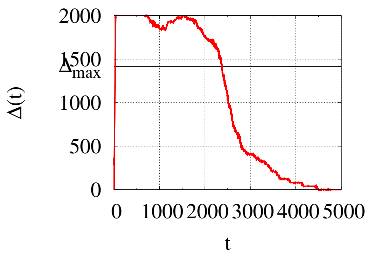
(m) PSO-OIW



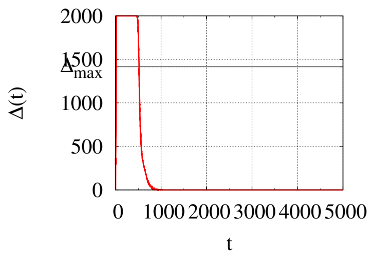
(n) PSO-RIW



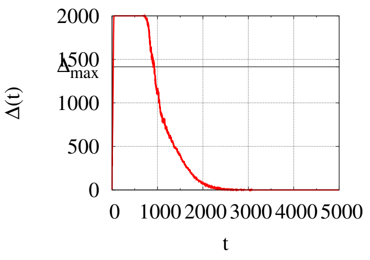
(o) PSO-RBI



(p) SRPSO



(q) PSO-SIW



(r) PSO-LDIW

Figure 5.3: Average particle movement on function f_{17} with $\Delta_{max} = 1414.214$ (part II: j–r).

top six performing algorithms, despite the finer-grained differences noted by the Mann-Whitney U tests. Additionally, the constant strategy had the largest number of problems for which it attained the best rank with respect to each performance measure, with 12 (63.2%), 14 (73.7%), and 13 (68.4%) problems, respectively. The worst overall strategy for all performance measures was DE-PSO, indicating that it not only showed the worst accuracy, but also the lowest consistency.

Table 5.4: Summary of performance on the 19 unimodal problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses. The maximum possible value for the Diff column is 323.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	89	8	2	73	3	6	93	5	3
APSO-VI	113	6	1	21	6	4	92	6	1
CDIW-PSO	116	5	1	19	8	4	60	8	1
Constant	**291**	**1**	12	**180**	**1**	14	**255**	**1**	13
DE-PSO	-302	18	0	-96	18	4	-203	18	1
DW-PSO	-212	16	0	-82	16	4	-149	15	1
FG-PSO	-202	15	0	-76	15	4	-154	16	1
IPSO-LT	-42	11	0	-12	9	4	-96	13	2
LD-PSO	112	7	0	21	6	4	74	7	1
PSO-NL	-101	13	0	-50	13	4	-82	12	1
PSO-NLI	-256	17	0	-91	17	4	-174	17	1
PSO-NEIW	-58	12	0	-28	10	4	-50	11	1
PSO-OIW	-121	14	0	-68	14	4	-124	14	1
PSO-RIW	262	2	5	152	2	7	227	2	5
PSO-RBI	-17	9	0	-28	10	4	-30	10	1
SRPSO	187	3	0	57	4	4	147	3	1
PSO-SIW	-32	10	0	-30	12	4	-18	9	1
PSO-LDIW	163	4	0	50	5	4	134	4	1

5.4.2 Multimodal Problems

A summary of the results for the 41 multimodal problems are presented in Table 5.5, while Figure 5.5 presents the critical difference plot for the accuracy measure. With respect to accuracy, the random strategy had the best overall performance with a difference score of 375. However, the random strategy had the best accuracy on only six (14.6%) problems, outnumbered by both APSO-VI, with 14 (34.1%) problems, and the constant strategy, with 11 (26.8%) problems. A noteworthy observation was that the constant

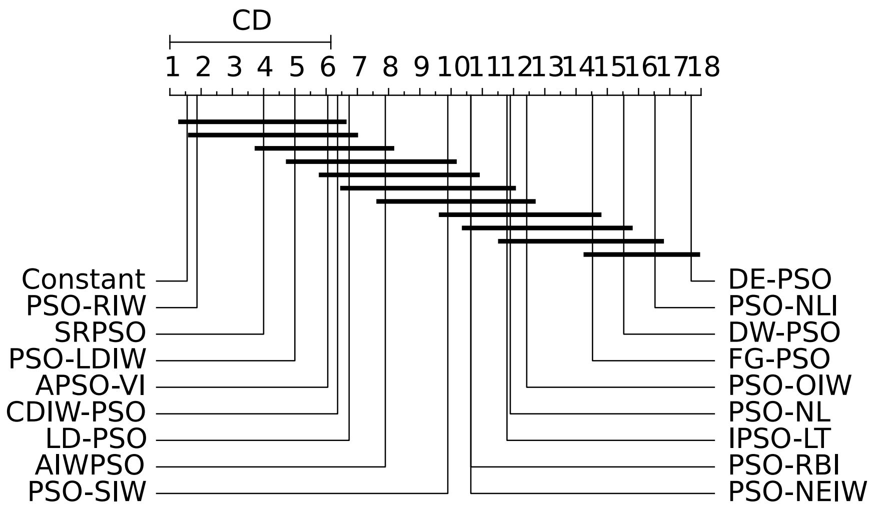


Figure 5.4: Critical difference plot using the Bonferroni-Dunn test on the 19 unimodal benchmark problems.

inertia weight ranked third best on multimodal problems, with a difference score of 339, while the second best accuracy was attained by the APSO-VI strategy, with a difference score of 358. Thus, the constant inertia weight had degraded accuracy when faced with multimodal problems. The DE-PSO algorithm again had the worst performance, with a difference score of -572.

Despite the degraded performance relative to the accuracy measure, the constant inertia weight strategy attained the highest rank for the success rate measure, with a difference score of 207. This indicates that, while the constant strategy did not attain the best accuracy, it did provide convergence to more accurate solutions, in general. Furthermore, the constant strategy depicted the best success rate on 25 (61.0%) of the multimodal problems. However, it should be noted that the success rate was insignificantly different among the strategies on many problems, as even the worst performing strategy, namely DE-PSO, displayed the highest rank on 17 (41.5%) of problems.

The APSO-VI strategy had the most consistent results on multimodal problems, with a difference score of 184, followed by the PSO-LDIW strategy with a score of 142. The APSO-VI algorithm also had the largest number of problems, 16 (39.0%), for which it attained the highest rank. Note that the random and constant strategies attained ranks of fourth and fifth respectively, and that the consistency measure on multimodal problems was one of only two scenarios in which the constant or random strategy did not attain the highest rank. The least consistent results were attained by the PSO-NLI strategy, which attained a difference score of -130.

Table 5.5: Summary of performance on the 41 multimodal problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses. The maximum possible value for the Diff column is 697.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	54	8	5	23	9	20	36	9	7
APSO-VI	358	2	14	153	3	23	**184**	**1**	16
CDIW-PSO	197	6	4	115	4	17	38	7	7
Constant	339	3	11	**207**	**1**	25	109	5	9
DE-PSO	-572	18	1	-266	18	17	-119	14	10
DW-PSO	-353	16	1	-165	16	17	-126	17	8
FG-PSO	-332	15	2	-150	15	17	-125	16	8
IPSO-LT	14	10	1	68	6	17	-64	12	7
LD-PSO	191	7	1	27	8	17	72	6	8
PSO-NL	-120	13	1	-16	10	18	-102	13	6
PSO-NLI	-441	17	1	-201	17	17	-130	18	6
PSO-NEIW	-29	11	1	-23	12	17	-42	10	6
PSO-OIW	-206	14	1	-89	14	17	-121	15	6
PSO-RIW	**375**	**1**	6	197	2	22	125	4	13
PSO-RBI	-80	12	1	-43	13	17	-51	11	6
SRPSO	307	4	3	98	5	17	141	3	9
PSO-SIW	34	9	2	-20	11	18	38	7	9
PSO-LDIW	276	5	3	67	7	17	142	2	8

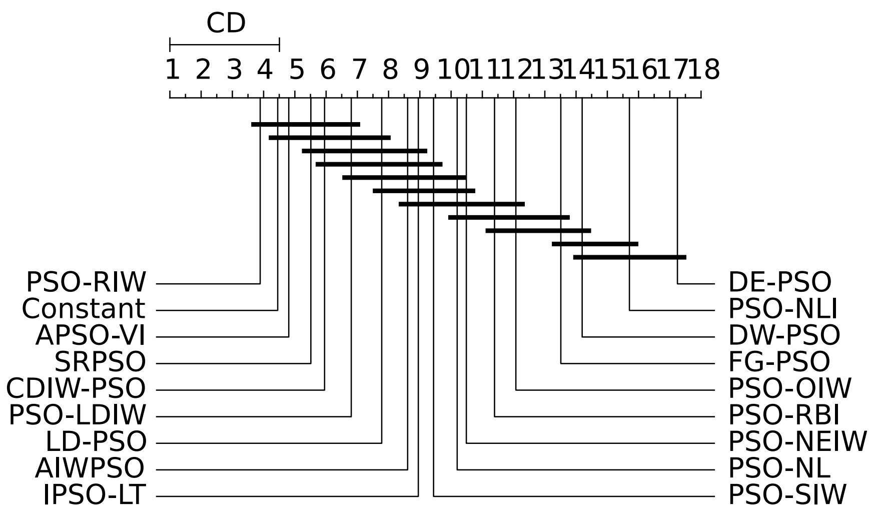


Figure 5.5: Critical difference plot using the Bonferroni-Dunn test on the 41 multimodal benchmark problems.

5.4.3 Separable Problems

The performance on the 21 separable problems is summarized in Table 5.6. Figure 5.6 depicts the critical differences with respect to the accuracy measure. For all three performance measures, the constant strategy showed the best performance with difference scores of 244, 134, and 185 for the accuracy, success rate, and consistency measures, respectively. Furthermore, the constant strategy also had the largest number of functions for which it attained the highest rank with nine (24.9%), 12 (57.1%), and nine (24.9%), respectively. The worst performance on separable problems was depicted by the DE-PSO strategy, with difference scores of -320, -94, and -162, respectively.

5.4.4 Non-Separable Problems

Table 5.7 depicts the performance of the various strategies on the 39 non-separable problems, while Figure 5.7 presents the critical differences for the accuracy measure. Considering the accuracy measure, the random strategy attained the most accurate solutions, with a difference score of 394. Despite having the highest rank overall, the random strategy attained the best performance on only six (15.4%) of the problems, while the constant strategy had the highest rank on 14 (35.9%) problems, and APSO-VI performed

Table 5.6: Summary of performance on 21 separable problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses. The maximum possible value for the Diff column is 357.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	94	6	3	70	3	10	87	6	5
APSO-VI	158	4	5	46	4	10	122	3	7
CDIW-PSO	90	7	1	20	7	7	38	8	2
Constant	**244**	**1**	9	**134**	**1**	12	**185**	**1**	9
DE-PSO	-320	18	0	-94	18	7	-162	18	2
DW-PSO	-195	16	0	-66	16	7	-125	15	3
FG-PSO	-184	15	0	-56	15	7	-135	16	2
IPSO-LT	-18	9	0	-3	8	7	-36	10	2
LD-PSO	82	8	0	-5	9	7	49	7	2
PSO-NL	-78	13	0	-29	11	7	-81	13	2
PSO-NLI	-242	17	0	-72	17	7	-146	17	2
PSO-NEIW	-52	12	0	-30	12	7	-51	12	2
PSO-OIW	-105	14	0	-44	14	7	-108	14	2
PSO-RIW	243	2	5	123	2	10	179	2	7
PSO-RBI	-21	10	0	-23	10	7	-48	11	2
SRPSO	179	3	0	39	5	7	116	5	2
PSO-SIW	-29	11	0	-30	12	7	-14	9	2
PSO-LDIW	153	5	0	28	6	7	117	4	2

best on 10 (25.6%) problems. Thus, despite attaining the highest accuracy on more than double the number of problems, the constant strategy performed slightly worse, with a difference score of 386, than the random strategy on non-separable problems. The worst accuracy was attained by the DE-PSO with a difference score of -554.

The observations for the success rate and consistency measures were the same: the best performance was noted for the constant strategy, with difference scores of 253 and 179, respectively, while the worst performance was attained by the DE-PSO strategy,

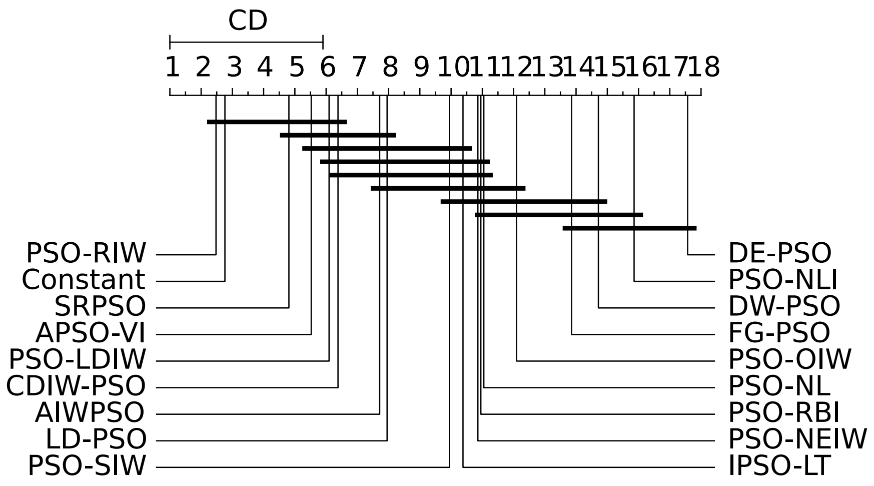


Figure 5.6: Critical difference plot using the Bonferroni-Dunn test on the 21 separable benchmark problems.

with difference scores of -268 and -160, respectively. Additionally, the constant strategy had the largest number of functions for which it attained the highest rank relative to both measures, with 27 (69.2%) and 13 (33.3%) problems, respectively.

5.4.5 Composition Problems

The final type of problem that was considered in this study were composition problems. The results for the 11 composition problems are presented in Table 5.8. For both the accuracy and success rate measures, the constant strategy attained the highest rank overall, with difference scores of 73 and 118, respectively. For both measures, the constant inertia weight also attained the best performance on four (36.4%) and six (54.5%) problems, respectively. An additional observation was that the difference scores for the accuracy measure were lower than those of the success rate, which indicates that there was significantly less deviation among the accuracy levels of the various strategies relative to the other problem classes. To further support this observation, Figure 5.8 presents the critical difference plot on the composition problems. This plot indicates that very minute differences in average rank were observed among the various strategies when considering the accuracy measure. For both measures, the DE-PSO algorithm depicted the worst overall performance.

Table 5.7: Summary of performance on the 39 non-separable problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses. The maximum possible value for the Diff column is 663.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	49	8	4	26	9	16	42	8	5
APSO-VI	313	4	10	128	3	17	154	5	10
CDIW-PSO	223	6	4	114	5	14	60	7	6
Constant	386	2	14	**253**	**1**	27	**179**	**1**	13
DE-PSO	-554	18	1	-268	18	14	-160	18	9
DW-PSO	-370	16	1	-181	16	14	-150	16	6
FG-PSO	-350	15	2	-170	15	14	-144	15	7
IPSO-LT	-10	10	1	59	7	14	-124	13	7
LD-PSO	221	7	1	53	8	14	97	6	7
PSO-NL	-143	13	1	-37	12	15	-103	12	5
PSO-NLI	-455	17	1	-220	17	14	-158	17	5
PSO-NEIW	-35	11	1	-21	11	14	-41	11	5
PSO-OIW	-222	14	1	-113	14	14	-137	14	5
PSO-RIW	**394**	**1**	6	226	2	19	173	2	11
PSO-RBI	-76	12	1	-48	13	14	-33	10	5
SRPSO	315	3	3	116	4	14	172	3	8
PSO-SIW	31	9	2	-20	10	15	34	9	8
PSO-LDIW	286	5	3	89	6	14	159	4	7

When considering the consistency measure for composition problems, a few unexpected results arose. Firstly, the most consistent strategy was DE-PSO, with a difference score of 74 and the highest rank on six (54.5%) problems. This is one of only two scenarios in which the DE-PSO did not exhibit the worst performance. The second unexpected result was the rank of the constant and random strategies at 15 and 17, respectively. Note that the rank of 17 for the random strategy was a tie with IPSO-LT, and thus constituted the worst overall consistency, and that the rank of 15 for the constant strategy

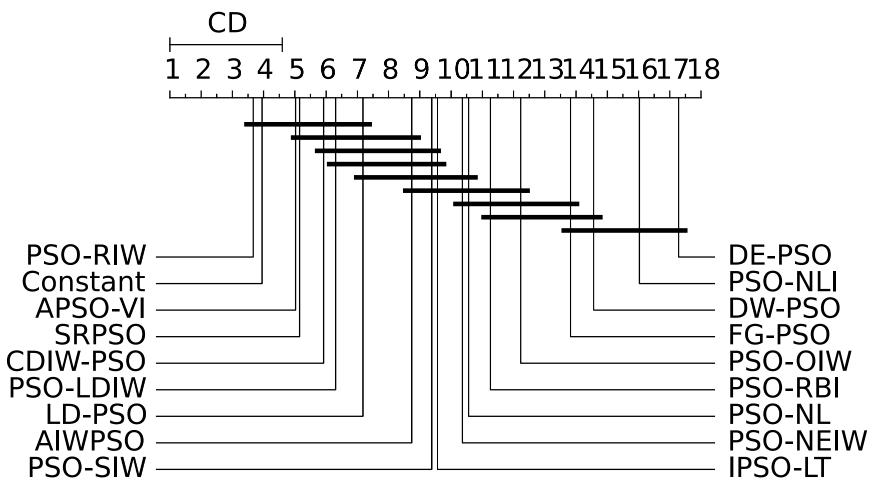


Figure 5.7: Critical difference plot using the Bonferroni-Dunn test on the 39 non-separable benchmark problems.

was a tie with PSO-NEIW, and thus constituted the penultimate consistency. The low rank attained by the random and constant strategies was quite unexpected given their performance on all other problem classes. This shows that despite their top performance, the random and constant strategies are extremely unstable on composition problems.

5.4.6 Summary of Results

In summary, the previous sections provided an empirical investigation into 18 inertia weight control strategies and further drilled down into the results by looking at five prominent characteristics of the benchmark problems, namely unimodal, multimodal, separable, non-separable, and composition to ascertain whether such characteristics have an impact on performance. A further examination of particle movement levels highlighted a correlation between particle movement and performance. Specifically, the best performing strategies depicted moderate initial movement levels, followed by a gradual decrease in movement. Conversely, the remaining strategies depicted either prohibitively low particle movements or rapid divergence, followed by rapid stagnation. A summary of the best and worst performing strategies for each problem type is given in Table 5.9. Thus, Table 5.9 can be taken as a recommendation of which strategy to employ based on the type of function and performance measure being examined. Note that,

Table 5.8: Summary of performance on the 11 composition problems. Bold entries indicate the strategies with the highest rank based on the difference of wins and losses. The maximum possible value for the Diff column is 187.

	Accuracy			Success Rate			Consistency		
Strategy	Diff	Rank	BRF	Diff	Rank	BRF	Diff	Rank	BRF
AIWPSO	-2	10	1	-3	11	1	0	7	2
APSO-VI	67	2	2	69	4	2	-6	10	3
CDIW-PSO	63	4	2	84	2	0	-21	13	3
Constant	**73**	**1**	4	**118**	**1**	6	-30	15	3
DE-PSO	-134	18	0	-143	18	0	**74**	**1**	6
DW-PSO	-65	16	0	-80	16	0	34	4	3
FG-PSO	-64	15	0	-69	15	0	42	3	3
IPSO-LT	42	6	0	47	5	0	-35	17	2
LD-PSO	31	8	0	10	8	0	-25	14	2
PSO-NL	-6	12	0	11	7	1	-17	12	2
PSO-NLI	-89	17	0	-106	17	0	56	2	2
PSO-NEIW	-4	11	0	0	10	0	-30	15	2
PSO-OIW	-37	14	0	-42	14	0	14	5	2
PSO-RIW	65	3	1	76	3	0	-35	17	2
PSO-RBI	-22	13	0	-20	13	0	7	6	2
SRPSO	47	5	0	41	6	0	-5	9	2
PSO-SIW	9	9	1	-7	12	1	-12	11	3
PSO-LDIW	33	7	0	6	9	0	-4	8	2

the worst performance for the consistency measure on the composition problems was a tie between IPSO-LT and the random strategy. In general, the problem type had only minimal effects on the overall results, as the same general trend was observed, with only a few exceptions, regardless of the problem type. Specifically, the solutions were most accurate when using a constant or random inertia weight strategy, the constant weight strategy always had the most solutions converging to the specified accuracy levels and was the most consistent with respect to accuracy for three of the five problem classes.

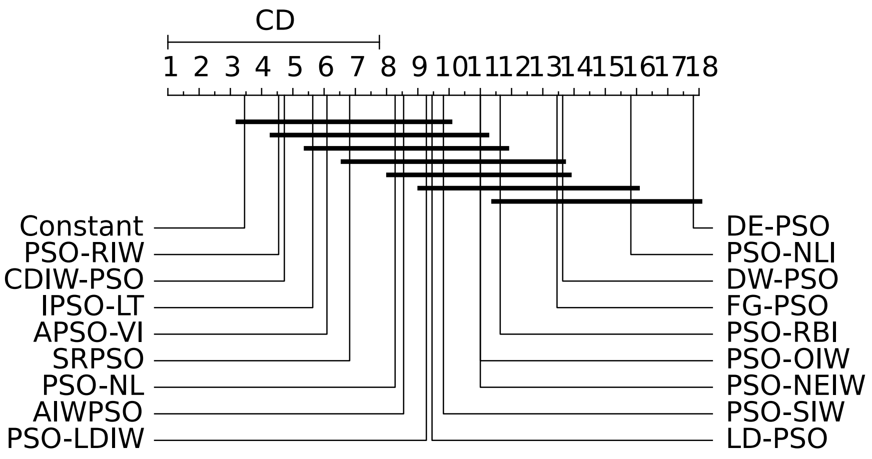


Figure 5.8: Critical difference plot using the Bonferroni-Dunn test on the 11 composition benchmark problems.

The remaining two problem types, namely multimodal and composition, had the most consistent results from the APSO-VI and DE-PSO strategies, respectively. In fact, it was only when considering the consistency on the multimodal and composition problem types that the DE-PSO strategy attained anything but the worst overall performance. Apart from the two aforementioned scenarios, the DE-PSO algorithm performed the worst with regards to all performance measures on all problem types.

Table 5.9: The best and worst performing strategy by function type.

	Accuracy		Success Rate		Consistency	
Problem	Best	Worst	Best	Worst	Best	Worst
All	PSO-RIW	DE-PSO	Constant	DE-PSO	Constant	DE-PSO
U	Constant	DE-PSO	Constant	DE-PSO	Constant	DE-PSO
M	PSO-RIW	DE-PSO	Constant	DE-PSO	APSO-VI	PSO-NLI
S	Constant	DE-PSO	Constant	DE-PSO	Constant	DE-PSO
NS	PSO-RIW	DE-PSO	Constant	DE-PSO	Constant	DE-PSO
C	Constant	DE-PSO	Constant	DE-PSO	DE-PSO	PSO-RIW

5.5 Summary

The primary purpose of this chapter was to analyse the performance of a number of inertia weight control strategies for the PSO algorithm on a comprehensive set of benchmark problems. This chapter analysed the stability implications of various inertia weight control strategies, deriving exact conditions for order-2 stability to be exhibited, where possible. Finally, a suite of 60 benchmark functions was employed to empirically examine the performance of the strategies relative to three performance measures. The benchmark problems encompassed a plethora of different characteristics, allowing the performance to be correlated with problem type.

All examined strategies were given equivalent PSO configurations, with the obvious exception of the way in which the inertia weight was controlled. Results of the empirical analysis indicated that despite their respective reported successes in the literature, the only inertia weight control strategy that performed on par with a constant inertia weight was the random weight strategy. Therefore, given an arbitrary optimization problem, it is preferable to have a constant inertia weight. If a non-static inertia is desired, then randomly selecting the inertia weight each iteration is the only worthwhile strategy of those examined. Furthermore, an examination of the average particle movement over time highlighted a correlation between the level of particle movement and performance. These results depict a grim state for the field of adaptive inertia weight strategies. The results clearly show a problem with the way in which authors examine their proposed strategies, as there is a tendency for newly proposed strategies to be compared with a small, seemingly arbitrarily chosen set of existing strategies and benchmark problems. However, it should be explicitly stated that the findings presented in this chapter are likely to change if the algorithmic configuration, such as the values of the acceleration coefficients or neighbourhood topology, are changed.

It is also noteworthy that similar studies have found that adaptive parameter strategies perform worse than constant or random control parameter values for both ant colony optimization [88] and tabu search [78]. Specifically, it was reported that the failure of the adaptive tabu search could be explained by the lack of adaptation to the local characteristics of the search space [78]. Given that many of the examined inertia weight control strategies base their adaptation mechanisms on a characterization of the search behaviour, rather than properties of the search space itself, it is likely their respective failures can be explained in the same manner as the previous findings for tabu search. Thus, proposing new inertia weight control strategies that adapt based on the characteristics of the local search space may lead to an improvement in performance.

This chapter further demonstrated that the PSO algorithm is sensitive to the values of the control parameters. In the following chapter, the regions in parameter space that lead to good performance are investigated.

Chapter 6

Investigating Optimal Parameter Regions

The analysis of inertia weight control studies in Chapter 5 clearly reinforced that the PSO algorithm is sensitive to the values of its control parameters. However, there is limited information about the correlation between the values of the control parameters and the performance of PSO. Therefore, to identify the regions of parameter space that lead to good performance, this chapter investigates the performance of PSO using 1012 distinct parameter configurations over a set of 22 benchmark problems. These 22 problems represent the base functions (i.e., the non-shifted, non-rotated, and non-composition problems) found in Appendix A, with the exception of f_{17} and f_{18}, which were omitted due to their dependencies on matrix operations. Further details about the experimental design are provided in Section 6.2. The overall experimental results are presented and discussed in Section 6.3, while Section 6.4 discusses the experimental results with regards to modality and separability. Finally, a summary of findings are presented in Section 6.5.

6.1 Motivation

Despite the reported successes of many SAPSO algorithms, a large majority of these algorithms have recently been shown to exhibit lacklustre performance [41, 42, 44, 109].

One plausible explanation for the poor performance of SAPSO algorithms is that the primary search performance (i.e., finding a solution to a continuous optimization problem) is almost entirely dependent upon the identification of well-performing parameters, which is itself a continuous optimization problem. Thus, a SAPSO algorithm employed to solve a continuous optimization problem must solve two such problems concurrently. To further complicate matters, failure to identify a reasonable parameter configuration will almost certainly lead to poor solutions for the primary search problem. Given that it is of utmost importance for a SAPSO algorithm to have an effective parameter search, any reduction in the complexity of finding appropriate parameter configurations would likely result in much better performance for SAPSO algorithms.

While many studies have examined the performance of PSO parameters from an empirical standpoint [9, 52, 71, 106], there is no general consensus on which parameter configurations lead to the best performance. Most of the empirical studies examined a fixed set of parameter configurations and determined those that performed best over a set of benchmark problems, thereby providing a recommended parametrization. Moreover, previous studies commonly used only a single neighbourhood topology (the original global-best topology) and, therefore, it is unknown whether the results hold when a different topology is employed. In contrast, this study does not focus primarily on the identification of the best-performing parameter configurations. Rather, this study aims to identify the regions of parameter space, for both the global-best and local-best topologies, that lead to superior performance over a variety of benchmark problems. This information, while also useful for general purpose parameter selection, can be used to bias the (parameter) search mechanisms of SAPSO algorithms, thereby improving their overall searching capabilities.

6.2 Experimental setup

To identify the regions of parameter space that lead to good performance, a total of 1012 parameter configurations were examined on a set of 22 benchmark functions, as summarized in Table 6.1. All of the benchmark functions were evaluated in 30 dimensions. The parameter configurations were constructed as sampled points, $(c_1 + c_2, \omega)$ where $c_1 = c_2$, taken every 0.1 units within the ranges

$$\omega \in [-1.1, 1.1] \text{ and } c_1 + c_2 \in [0.1, 4.4].$$

Each experiment made use of synchronous updates and ran for 5000 iterations with a swarm size of 30. Experiments were repeated 30 times using both the global best (gbest) and local-best (lbest) topologies. For the lbest topology, a neighbourhood size of three was used, as depicted in Figure 2.1b. To prevent infeasible attractors, personal best positions were only updated if the new position was feasible and had a better fitness than the previous personal best position. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. Particles were initialized uniformly within the feasible region and velocities were initialized to the zero vector [30].

Results were analysed using the following statistical analysis procedure. For each benchmark problem and topology, a Kruskal-Wallis test was performed to first determine if any significant differences existed among the fitness values obtained by using each of the parameter configurations. If the Kruskal-Wallis test indicated that a significant difference existed, pairwise Mann-Whitney U tests were then performed to identify the individual differences. When the Mann-Whitney U test indicated that a difference in performance existed, the average fitness values were used to assign wins and losses; the parameter configuration that lead to better performance was awarded a win, while the inferior configuration was awarded a loss. Finally, the parameter configurations were ranked based on the difference between the number of wins and losses. Both the Kruskal-Wallis and Mann-Whitney U tests were performed at a significance level of 0.05.

6.3 Overall Results

Table 6.2 presents the 10 best parameter configurations, as determined by the difference score aggregated across all 22 benchmark problems. The first key observation was that the gbest topology performed best with larger acceleration coefficients (i.e., values of c_1 and c_2), while the lbest topology performed best with slightly smaller values for these coefficients. This is further evidenced by Figure 6.1, which plots the overall rank

Table 6.1: Characteristics of the benchmark functions. 'Equation' specifies the equation number of the function (see Appendix A). 'Modality' specifies the modality ('U' for unimodal, 'M' for multimodal). 'Separability' denotes the separability ('S' for separable, 'NS' for non-separable).

Function	Name	Equation	Modality	Separability
f_1	Absolute Value	(A.1)	U	S
f_2	Ackley	(A.2)	M	NS
f_3	Alpine	(A.3)	M	S
f_4	Egg Holder	(A.4)	M	NS
f_5	Elliptic	(A.5)	U	S
f_6	Griewank	(A.6)	M	NS
f_7	HyperEllipsoid	(A.7)	U	S
f_8	Michalewicz	(A.8)	M	S
f_9	Norwegian	(A.9)	M	NS
f_{10}	Quadric	(A.10)	U	NS
f_{11}	Quartic	(A.11)	U	S
f_{12}	Rastrigin	(A.12)	M	S
f_{13}	Rosenbrock	(A.13)	M	NS
f_{14}	Salomon	(A.14)	M	NS
f_{15}	Schaffer 6	(A.15)	M	NS
f_{16}	Schwefel 1.2	(A.16)	U	NS
f_{19}	Schwefel 2.21	(A.19)	U	S
f_{20}	Schwefel 2.22	(A.20)	U	S
f_{21}	Shubert	(A.21)	M	NS
f_{22}	Spherical	(A.22)	U	S
f_{23}	Step	(A.23)	M	S
f_{24}	Vincent	(A.24)	M	S

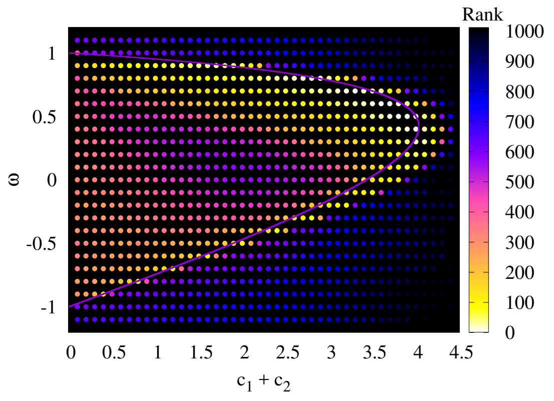
attained by each parameter configuration for both topologies. As seen in Figure 6.1, the best performing parameter configurations (indicated by light-coloured points) for the gbest topology, shown in Figure 6.1a, tended to be clustered closer to the top right

Table 6.2: The ten best parameter configurations by overall rank across all benchmark functions.

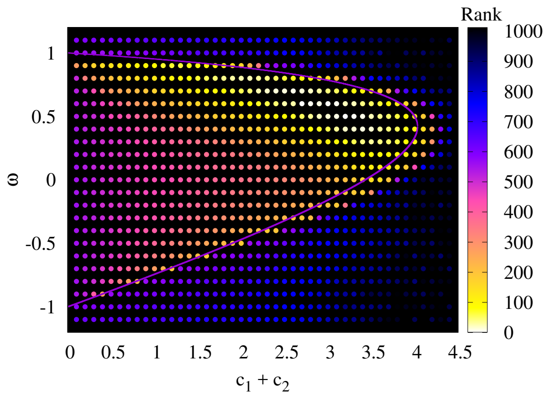
Global Best Topology				Local Best Topology		
ω	$c_1 = c_2$	Average Rank (SD)	**Overall Rank**	ω	$c_1 = c_2$	Average Rank (SD)
0.5	1.90	18.727 (14.149)	**1**	0.5	1.60	44.227 (25.064)
0.6	1.80	20.091 (14.068)	**2**	0.6	1.35	43.273 (35.158)
0.4	2.00	20.682 (15.683)	**3**	0.6	1.50	45.000 (29.552)
0.5	1.95	18.955 (14.113)	**4**	0.5	1.65	44.955 (26.561)
0.7	1.65	19.909 (12.577)	**5**	0.6	1.55	39.818 (26.388)
0.6	1.85	19.909 (14.491)	**6**	0.6	1.40	46.727 (32.445)
0.6	1.75	21.227 (18.452)	**7**	0.7	1.30	44.091 (19.914)
0.7	1.60	21.455 (14.222)	**8**	0.5	1.70	43.773 (32.433)
0.5	1.85	23.182 (21.830)	**9**	0.5	1.55	51.000 (34.630)
0.7	1.70	21.273 (14.945)	**10**	0.6	1.45	47.364 (28.065)

section of the stable region than their lbest counterparts, shown in Figure 6.1b.

Figure 6.2 visualizes the 100 best parameter configurations (determined by aggregate rank), and further demonstrates that many of the best parameter configurations in the gbest topology had larger acceleration coefficients. However, it is also noted that the best parameters for the gbest topology had a much stronger tendency to be near the boundary of the stable region. Specifically, Figure 6.2a shows a number of the best parameter configurations were situated near the lower boundary of the stable region for the gbest topology, while no such configurations were present in Figure 6.2b for the lbest topology. Additionally, 30 of the top 100 parameter configurations for the gbest topology violated Poli's stability criterion, while none of the best configurations for the lbest violated the criterion. This result is not completely unexpected given that there is an observable discrepancy between the theoretical and empirical stable regions for traditional benchmark problems [20]. Nonetheless, it is noteworthy that when using the gbest topology, strict adherence to the stability criteria does not necessarily translate into superior performance. Rather, selecting parameter configurations that are outside the theoretically stable region, but are within reasonably close proximity to the apex, may still lead to relatively good performance. However, when employing the lbest topology,



(a) Global best topology.



(b) Local best topology.

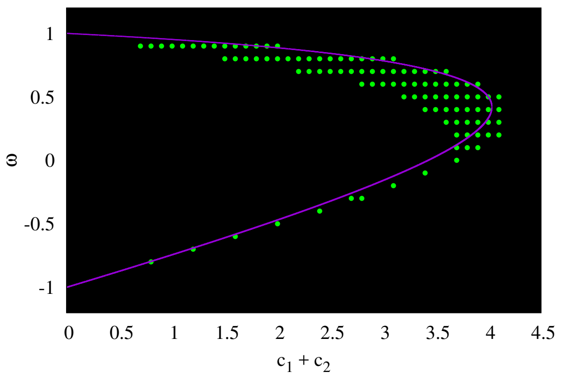
Figure 6.1: Plot of the overall rank across all 22 benchmark problems.

adherence to the stability criterion is of greater benefit as none of the 100 best parameter configurations reside outside of the theoretical stable region.

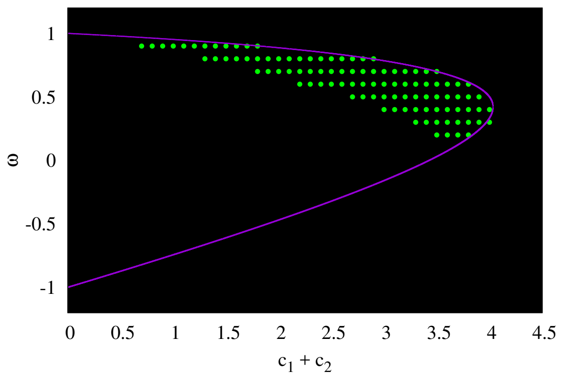
To quantify the observed differences in rank between the two topologies, the average absolute difference in rank between corresponding parameter configurations in the gbest and lbest topologies was computed. More formally, the average difference was calculated as

$$\frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}|R_g(p)-R_l(p)|, \tag{6.1}$$

where p is a specific parameter configuration from the set of examined configurations $\mathcal{P}$, $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$, and the overall ranks for the PSO algorithms using parameter configuration p are denoted by $R_g(p)$ and $R_l(p)$ for the gbest and lbest topologies, respectively. Using Equation (6.1), the average difference in rank between the two topologies was 75.733 with a standard deviation of 77.558. The distribution of the difference values is shown in Figure 6.3. The average difference in rank indicates that the ranks were notably different between corresponding parameter configurations among the two topologies. The difference in rank provides a quantification of the effect that the topology had on the relative performance of parameter configurations. A small difference in rank indicates that topology had minimal effects on the overall performance when using the same parameter configuration. Conversely, a large difference in rank indicates that the change in topology caused a more noticeable effect on the relative performance when



(a) Global best topology.



(b) Local best topology.

Figure 6.2: Plot of the best 100 parameter configurations by overall rank.

the parameter configurations were employed by PSO. Therefore, a large difference in rank suggests that the areas of the parameter space that lead to good performance were different among the two topologies.

An average difference of 75.733 provides strong evidence to support that the topology employed is an important factor when selecting control parameter values for the PSO algorithm. Similarly, if the PSO control parameters are not being tuned for a specific problem, then examining both the gbest and lbest topologies for the selected parametrization can be an effective way of improving performance. This result also suggests that comparing gbest and lbest PSO results using the same parameter configuration may be unfair, in general, given that the relative ranks are drastically different among corresponding parameter configurations. Specifically, a naive comparison using the same parameter configuration for both topologies will inherently create a bias towards the topology that leads to better performance using that parameter configuration. Thus, a more fair comparison would be to examine the performance of PSO with each topology over a set of parameters, then perform the comparison using the best respective parameter configuration(s). In this scenario, the relative rankings are preserved, thereby mitigating the bias to some degree and leading to a more fair comparison of performance.

A further observation from Figure 6.1 was made regarding the worst performing parameter configurations within the stable region. When using the gbest topology, the worst performance, indicated by darker-coloured points, was observed for parameters

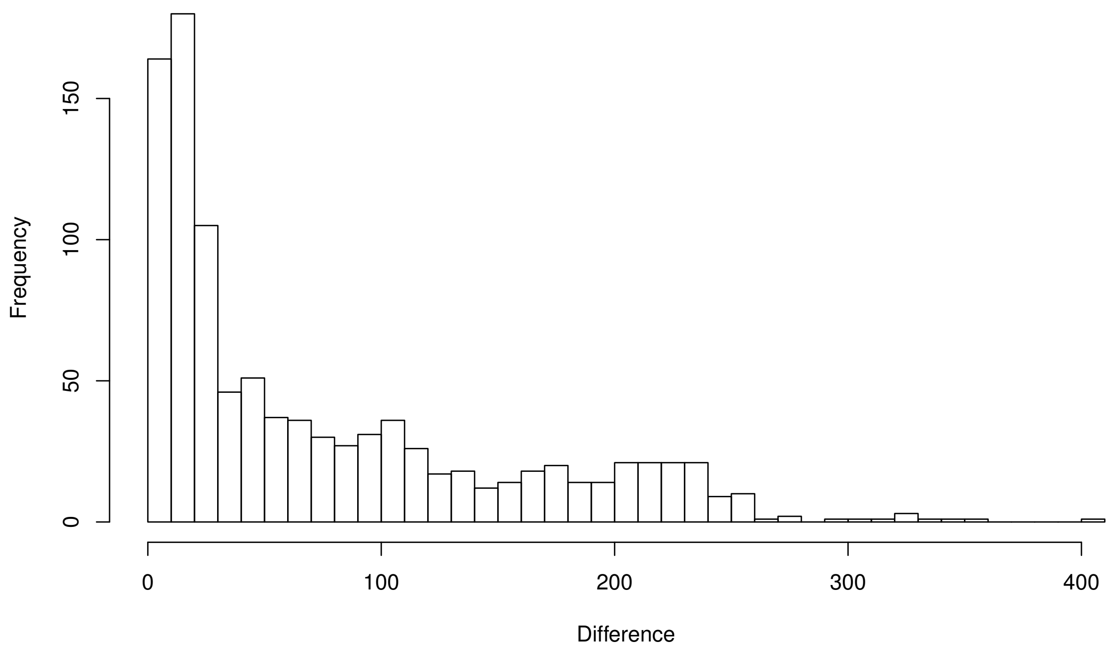


Figure 6.3: Distribution of the absolute difference in rank between corresponding parameter configurations using the gbest and lbest topologies.

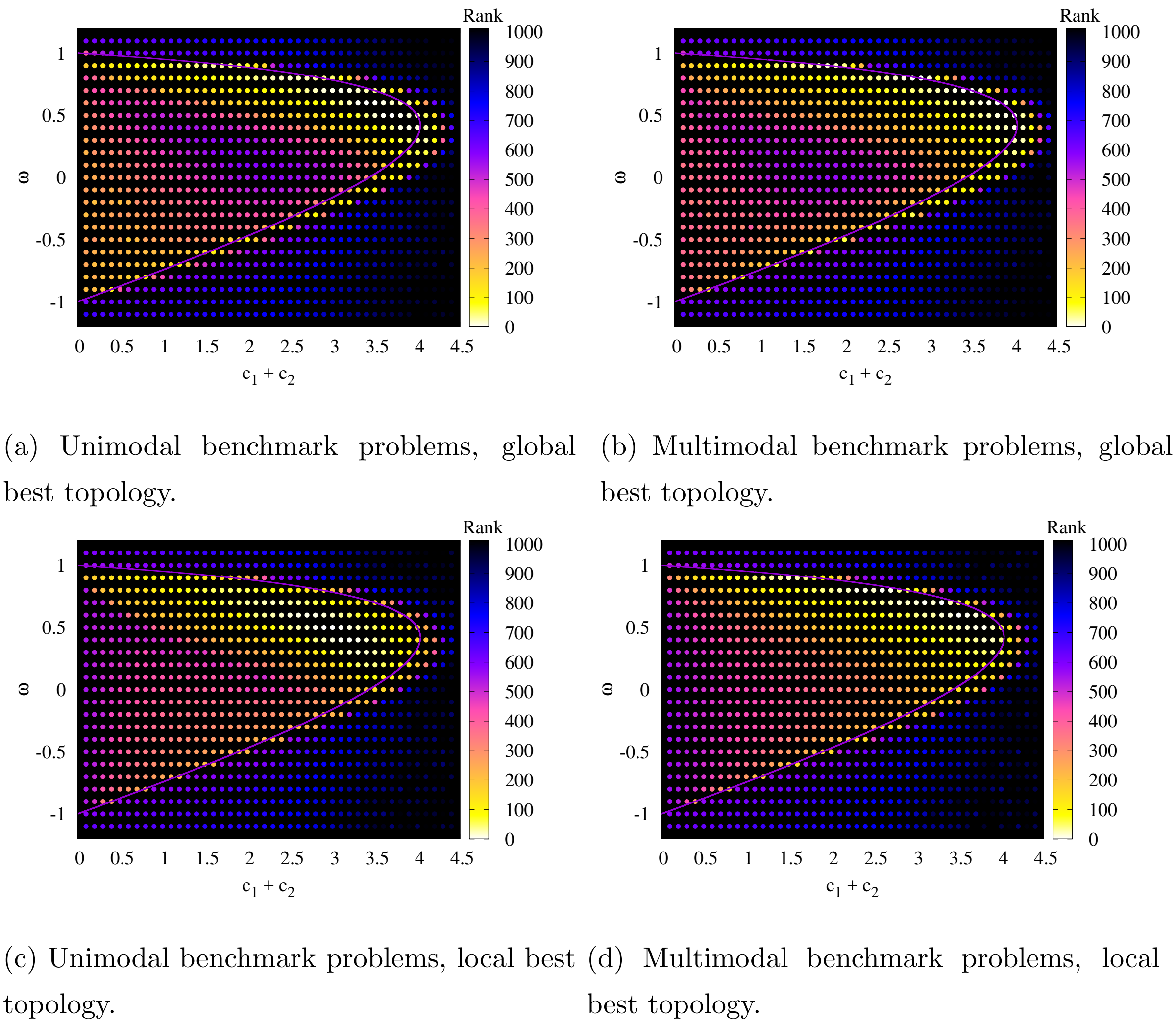
near the centre of the region (i.e., $c_1 + c_2 \in [1, 3]$ and $\omega \in [-0.5, 0.5]$). However, for the lbest topology, the worst performance was observed when small values for the acceleration coefficients (i.e., $c_1 + c_2 \in [0, 1]$) were used, largely irrespective of the inertia weight.

Another observation, from Table 6.2, was that both the average and standard deviation of per-function ranks were significantly lower for the gbest topology. This indicates that the performance of the best parameter configurations was more stable under the gbest topology. More concretely, this indicates that the gbest topology has a less profound dependence on parametrization in the sense that well-performing parameters are more likely to perform better across a variety of problems than when using the lbest topology.

The following section examines whether the modality and separability of the problems have an effect on which areas of the parameter space lead to the best performance.

6.4 Results by Environment Type

Figure 6.4 shows the rank for each parameter configuration across the 9 unimodal and 13 multimodal problems, respectively, for both topologies. Additionally, Figure 6.5 visualizes the rank for each parameter configuration across the 12 separable and 10 non-separable problems, respectively. In each of the plots, the striking similarities to the respective overall ranks in Figure 6.1 were noted. That is, the regions where the best and worst performance are observed remained largely unchanged for both topologies, irrespective of the modality and separability of the problem.



(a) Unimodal benchmark problems, global best topology.

(b) Multimodal benchmark problems, global best topology.

(c) Unimodal benchmark problems, local best topology.

(d) Multimodal benchmark problems, local best topology.

Figure 6.4: Plot of the aggregate rank based on modality.

For each problem type and parameter configuration, the average difference between the overall rank and the rank on the specified problem type was computed in an analogous fashion to the comparison between topologies provided by Equation (6.1). Table 6.3 presents the average difference in rank and corresponding standard deviation based on problem type for each of the topologies. It is noteworthy that, in all cases, the average difference between overall rank and each problem type was less than one-third of the difference in rank between the two topologies. In other words, the topology had a much greater influence on the relative performance of an arbitrary parameter configuration than the modality and separability of the problem.

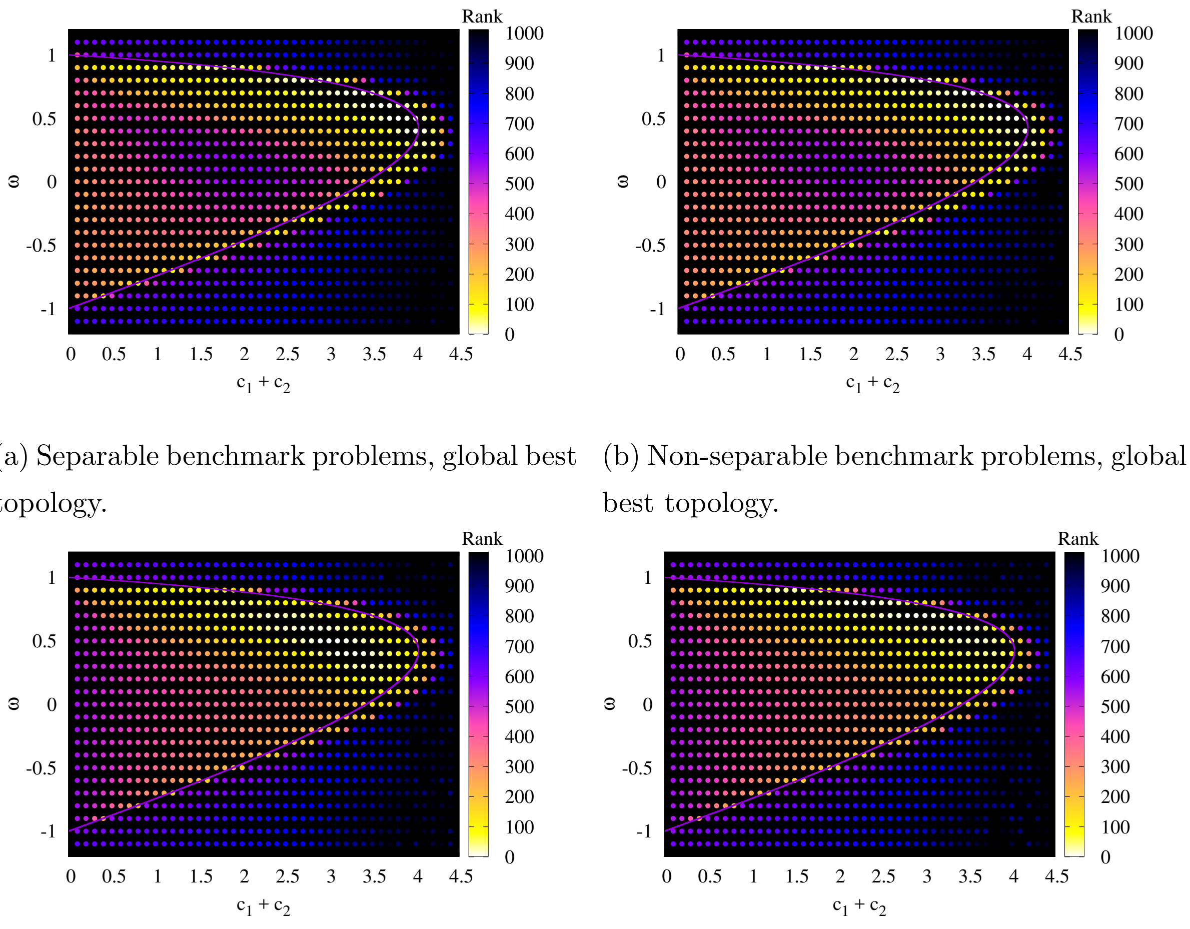
As further analysis, a Mann-Whitney U test was executed, at a significance level of 0.05, using a vector of the pairwise difference values (referred to as the 'difference vector') for the gbest and lbest topologies on each problem type to ascertain whether the effect of the problem type was consistent between the two topologies. In all cases, the Mann-Whitney U tests indicated that the difference vectors were insignificantly different, suggesting that the modality and separability of a problem has the same relative influence on the performance attained using various PSO parameter configurations, irrespective of whether the gbest or lbest topology is employed.

Table 6.3: Average difference in rank, relative to the overall rank, by problem type.

	Global Best Topology		Local Best Topology	
Problem Type	Mean	SD	Mean	SD
Unimodal	24.790	30.317	24.052	25.413
Multimodal	22.271	30.275	15.116	15.821
Separable	9.349	13.638	8.958	10.579
Non-separable	11.689	19.310	11.298	13.390

6.5 Summary

In summary, topology does play a significant role in determining the regions of parameter space that perform well. This is evidenced by Figure 6.1, which clearly depicted that the regions leading to the best and worst performance were noticeably different among the



(a) Separable benchmark problems, global best topology.

(b) Non-separable benchmark problems, global best topology.

(c) Separable benchmark problems, local best topology.

(d) Non-separable benchmark problems, local best topology.

Figure 6.5: Plot of the aggregate rank based on separability.

gbest and lbest topologies. Figure 6.2 provided additional evidence to support that the regions of parameter space that lead to the best performance were different among the two topologies. Moreover, the rank of corresponding parameter configurations differed by nearly 76 between the two topologies. These results suggest that comparing the performance of a gbest and lbest PSO using the same parameter configuration may be unfair.

When the modality and separability were examined, it was shown that neither of these problem characteristics had a significant influence on the regions of parameter space where good performance is attained. Figures 6.4 and 6.5, which show the ranks based on the modality and separability of problems, depicted striking similarities to the respective plots of the overall rank shown in Figure 6.1. The results also showed that the performance differences among the various problem types was consistent across both topologies. That is, the difference in rank among corresponding points in the gbest and lbest topologies was insignificantly different for all problem types, suggesting that the performance implications of the modality and separability of the problem were independent of the topology employed.

The following chapter expands on this investigation to examine scenarios where the cognitive and social acceleration coefficients are not equal using the global best topology. Further, the next chapter investigates whether the optimal region for PSO parameters remains fixed over time.

Chapter 7

Optimal Parameter Regions and the Time-Dependence of Control Parameter Values

As discussed in Chapter 6, there have been a number of studies that examined the performance of various PSO parameter configurations. Despite these studies, there is no general consensus as to which parameter configurations lead to the best performance. Moreover, two important questions remain unanswered. Specifically, it is unknown whether the optimal parameter regions depend on time. Additionally, the previous chapter did not address whether the results were also applicable if the values of the cognitive and social acceleration coefficients were not equal.

This chapter provides answers to both questions presented above, namely how the best parameter values change over time, and what regions of parameter space lead to the best performance when the values of the cognitive and social control parameters are not equal. This chapter contributes to the understanding of PSO by concluding that the balance between the social and cognitive acceleration coefficients has a significant impact on the areas in parameter space that lead to good performance. Moreover, this chapter shows that the optimal regions in parameter space shift over time, thus providing direct evidence in support of SAPSO algorithms.

The remainder of this chapter is structured as follows. Section 7.1 describes the experimental procedures, the results of which are presented in Section 7.2. Finally, Section 7.3 provides a summary of the findings.

7.1 Experimental Setup and Statistical Analysis

To identify the regions of parameter space that lead to good performance, a total of 3036 parameter configurations were examined on a set of 22 benchmark problems, as summarized in Table 6.1. All of the benchmark functions were evaluated in 30 dimensions. Further descriptions of the benchmark functions can be found in Appendix A. Each experiment made use of synchronous updates [93] and ran for 5000 iterations with a swarm size of 30. Experiments were repeated 30 times using the global best (star) topology. To prevent infeasible attractors, personal best positions were only updated if the new position was both feasible and had a better objective function value than the previous personal best position. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. Particles were initialized uniformly within the feasible region and velocities were initialized to the zero vector [30].

Experiments were grouped into three categories based on the ratio between the cognitive and social control parameter values to ascertain the effects of imbalanced acceleration coefficients. The experiments were labelled 'equal', where $c_1 = c_2$, 'cognitive', where $c_1 = 3c_2$, and 'social', where $c_2 = 3c_1$. For each experiment type, the parameter configurations were constructed as sampled points, (C, ω), taken every 0.1 units within the ranges

$$C \in [0.1, 4.4] \text{ and } \omega \in [-1.1, 1.1].$$

Note that, while negative inertia weights are not traditionally employed in the PSO algorithm, recent results have shown that negative inertia weights can lead to stable behaviour, and thus are not necessarily unreasonable [6, 21]. The values assigned to the control parameters c_1 and c_2 were then calculated based on the type of experiment. The control parameter values were taken as $c_1 = c_2 = \frac{C}{2}$ for experiments labelled 'equal', $c_1 = \frac{3C}{4}$ and $c_2 = \frac{C}{4}$ for experiments labelled 'cognitive', and $c_1 = \frac{C}{4}$ and $c_2 = \frac{3C}{4}$ for experiments labelled 'social'. For the cognitive and social experiments, a multiplier value of three was chosen to provide an environment where either the cognitive or social acceleration coefficient would dominate the other, without resorting to a cognitive-only or social-only model. Each experiment group thus consisted of 1012 parameter configurations, leading to a combined total of 3036 parameter configurations examined in this study.

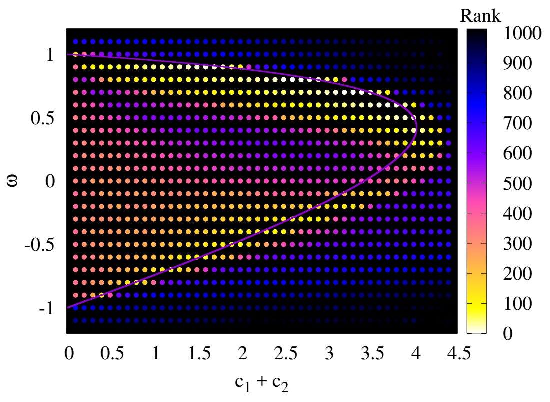
Results were analysed using the following statistical analysis procedure. For each benchmark problem, a Kruskal-Wallis test was performed to first determine if any significant differences existed among the fitness values obtained by using each of the parameter configurations. If the Kruskal-Wallis test indicated that a significant difference existed, pairwise Mann-Whitney U tests were then performed to identify the individual differences. When the Mann-Whitney U test indicated that a difference in performance existed, the median fitness values were used to assign wins and losses; the parameter configuration that lead to better performance was awarded a win, while the inferior configuration was awarded a loss. Finally, the parameter configurations were ranked based on the difference between the number of wins and losses. In this context, a lower rank corresponds to superior performance. Both the Kruskal-Wallis and Mann-Whitney U tests were performed at a significance level of 0.05.

7.2 Identifying Optimal Parameter Regions

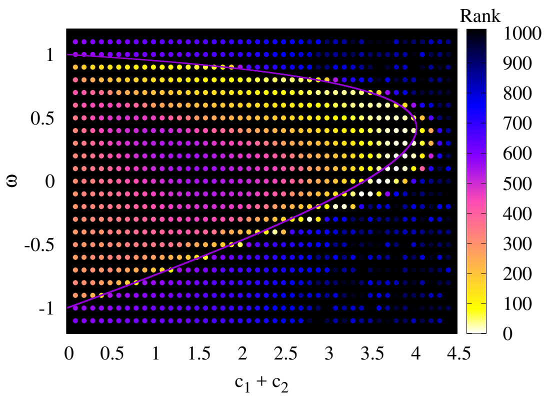
This section presents the results of the experiments described in Section 7.1. First, the overall performance of the parameter configurations is examined in Section 7.2.1, followed by an examination of the time-dependence in Section 7.2.2.

7.2.1 Overall Performance

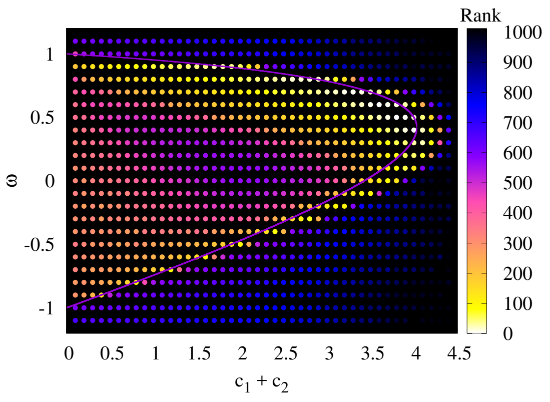
Figure 7.1 depicts the overall rank of each parameter configuration for each experiment after 5000 iterations. While each of the experiments depicted a tendency for parameters near the boundary of the theoretically stable region to perform best, there are observable differences regarding where the best parameters lie. Such observations are reinforced by Figure 7.2, which visualizes the 100 best parameter configurations by overall rank after 5000 iterations for each experiment type. For the cognitive experiments, a majority of the best parameter configurations were clustered along the upper boundary. However, there was also a notable cluster of points along the lower boundary. For the social experiments, the best parameters were clustered very strongly around the apex, with a slight preference for positive values of ω. For the equal experiments, the best parameters were mostly clustered around the apex, with a slightly more pronounced preference for positive values of ω than the social configurations.



(a) Cognitive configurations.



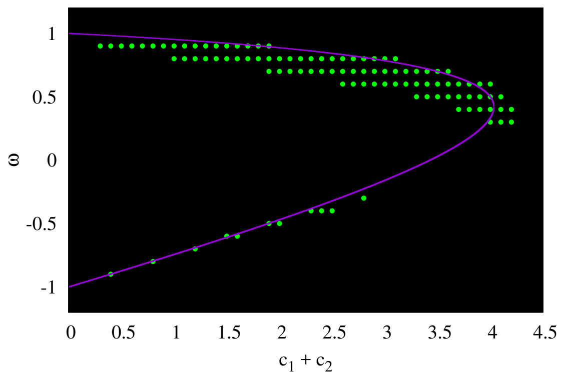
(b) Social configurations.



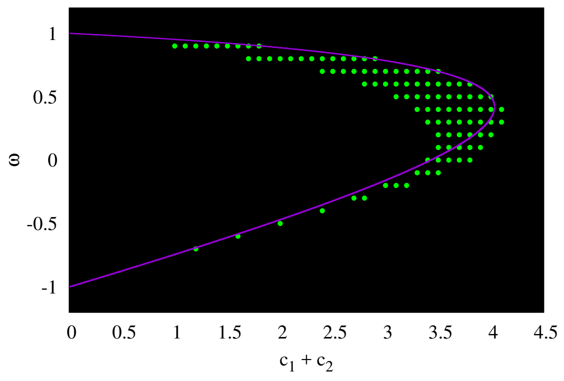
(c) Equal configurations.

Figure 7.1: Overall rank after 5000 iterations based on experiment type.

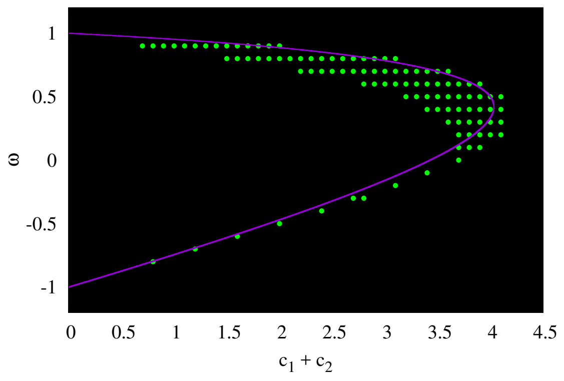
Figure 7.1 shows that the regions leading to the worst performance were significantly different among the various configurations. For each of the experiment types, it is clear that there was a tendency for parameters that lie outside the theoretically stable region to perform worse than those within the region. However, the exact areas that lead to the absolute worst performance were notably different. For the cognitive experiments, the



(a) Cognitive configurations.



(b) Social configurations.



(c) Equal configurations.

Figure 7.2: 100 best parameter configurations after 5000 iterations by overall rank.

worst parameters were mostly dependent upon the value of ω. That is, configurations where $|\omega| \approx 1$ (albeit more prominently with $\omega \approx -1$) tended to perform the worst, largely irrespective of the values of c_1 and c_2. For the social configurations, the worst performance was scattered throughout the extreme ends of the examined parameter space. Examining the equal configurations, the worst performance was again clustered around the extreme ends of the examined parameter space. However, it was noted that, for parameter configurations that lie outside the stable region, there was a definite correlation between distance to the stable region and performance. Specifically, it is clear that performance degradation was proportional to the distance from the stable region. It was also noted that, for each experiment type, there was a cluster within the stable region that lead to relatively poor performance.

Table 7.1 provides a summary of the fitness attained by the best performing parameter configuration for each benchmark problem. In the event of a tie, the parameter configuration that lead to the lowest median fitness was selected. In the cases of f_{23} and f_{24}, no definitive best configuration could be chosen given that there were multiple parameter configurations that always lead to the optimal fitness being attained. For f_{23}, there were 69 configurations that always lead to the optimal fitness, while there were three such parameter configurations for f_{24}. From Table 7.1, it is clear that the best parameter values to employ are problem specific. Furthermore, there is no clear configuration type that outperformed the others. However, when the best configurations are correlated with modality, a few observations can be made. For seven of the functions, six of which were unimodal, an equal configuration lead to the best performance. For five of the functions, all of which were multimodal, a cognitive configuration lead to the best performance. For eight of the functions, five of which were multimodal, a social configuration lead to the best performance. In other words, the best performance was attained for six out of nine unimodal problems when $c_1 = c_2$, and in no instance did a cognitive configuration lead to the best performance on a unimodal problem. In contrast, parameter configurations that had $c_1 = c_2$ attained the best performance on only one multimodal problem, whereas the cognitive and social configurations each lead to the best performance on five multimodal problems. A further observation regarding Table 7.1 was made when the best parameters were correlated with the stability criterion given in Equation (2.5). For five of the benchmark problems, all of which were multimodal, the best parameter configuration violated the stability criterion. However, it should be noted that violating the stability criterion does not necessarily imply divergence, but rather that stability can not be guaranteed.

Table 7.2 presents the 10 best parameter configurations determined by aggregate rank across all benchmark problems. Of the best 10 parameter configurations, four had $c_1 = c_2$, while six were categorized as social configurations; none of the best 10 parameter configurations were categorized as cognitive configurations. In fact, the best cognitive configuration had a rank of 19. Moreover, of the best 100 parameter configurations, 54 were categorized as social, 36 were categorized as equal, while only nine were cognitive

Table 7.1: Summary of the fitness attained by the best parameter configuration for each benchmark function. A * indicates that multiple parameter configurations were found that always lead to the optimal fitness.

Function	ω	c_1	c_2	Median	Average	Standard Deviation
f_1	0.4	2.000	2.000	3.17E-048	2.94E-042	1.24E-0.041
f_2	0.7	2.550	0.850	6.66E-015	5.01E-002	2.74E-001
f_3	0.7	2.625	0.875	1.07E-014	1.69E-014	3.15E-014
f_4	-0.1	0.875	2.625	-1.57E+004	-1.59E+004	1.25E+003
f_5	0.4	1.950	1.950	2.76E-100	9.64E-094	3.81E-093
f_6	0.5	0.975	2.925	3.70E-003	1.01E-002	1.81E-002
f_7	0.4	1.950	1.950	9.96E-105	1.13E-099	5.99E-099
f_8	-0.5	1.500	0.500	-2.70E+001	-2.69E+001	8.34E-001
f_9	0.6	1.850	1.850	-7.85E-001	-7.88E-001	6.88E-003
f_{10}	0.5	1.750	1.750	6.69E-014	2.12E-013	5.46E-013
f_{11}	0.5	1.750	1.750	1.11E-179	7.52E-170	0.00E+000
f_{12}	-0.1	0.900	2.700	1.39E+001	1.51E+001	7.93E+000
f_{13}	0.8	0.500	1.500	2.47E+000	3.47E+000	3.55E+000
f_{14}	0.8	2.025	0.675	3.00E-001	3.20E-001	5.51E-002
f_{15}	0.1	0.950	2.850	4.51E+000	4.56E+000	6.36E-001
f_{16}	0.2	0.900	2.700	2.60E-014	3.10E-013	8.45E-013
f_{19}	0.0	0.850	2.550	1.02E-007	1.41E-007	1.57E-007
f_{20}	0.4	2.000	2.000	8.21E-048	9.40E-036	5.15E-035
f_{21}	0.8	2.025	0.675	-1.89E+034	-2.02E+034	9.27E+033
f_{22}	0.4	1.950	1.950	1.31E-106	1.08E-100	5.82E-100
f_{23}	*	*	*	0.00E+000	0.00E+000	0.00E+000
f_{24}	*	*	*	-3.00E+001	-3.00E+001	0.00E+000

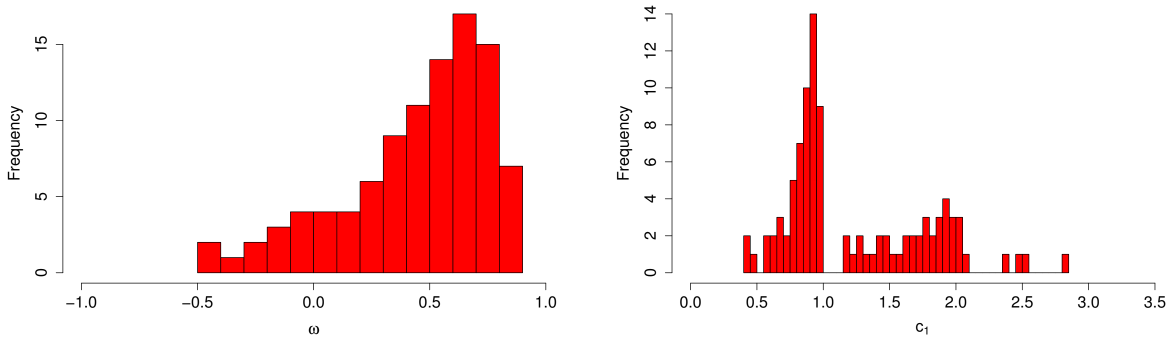
configurations. This result is not surprising given that the strength of the PSO algorithm lies in its social aspect. Three of the best 10 configurations employed a negative inertia weight, indicating a preference to switch directions rather than resist directional changes. Despite the stable region defined by Equation (2.5) containing negative inertia weight values, they are rarely, if ever, used in practice. However, these results suggest that negative inertia weights may not necessarily be detrimental to the search process. A further two of the best 10 configurations employed no inertia at all (i.e., $\omega = 0$). Given that half of the best 10 parameter configurations employed non-positive inertia weight values, it can be concluded that it is not always beneficial for particles to remain on their current trajectory. However, it should be noted that all five of the parameter configurations that had non-positive inertia weights were also categorized as social configurations, while four of the five configurations with positive inertia weight values had equal values for the social and cognitive coefficients. This suggests that having a high social influence may, in part, make having a positive inertia weight value unnecessary. Nonetheless, it is generally not recommended to employ a negative inertia weight, especially for cognitive parameter configurations.

Table 7.2: The 10 best parameter configurations by overall rank across all benchmark problems.

Overall Rank	ω	c_1	c_2	Average Rank	Rank SD
1	0.1	0.950	2.850	67.955	43.951
2	-0.1	0.875	2.625	64.864	54.961
3	0.0	0.900	2.700	66.727	49.891
4	0.0	0.925	2.775	67.364	45.390
5	0.6	1.800	1.800	68.591	58.467
6	0.5	1.900	1.900	67.773	58.557
7	0.7	1.650	1.650	67.500	47.914
8	-0.2	0.800	2.400	76.227	61.082
9	-0.3	0.700	2.100	75.227	50.191
10	0.6	1.850	1.850	67.864	55.555

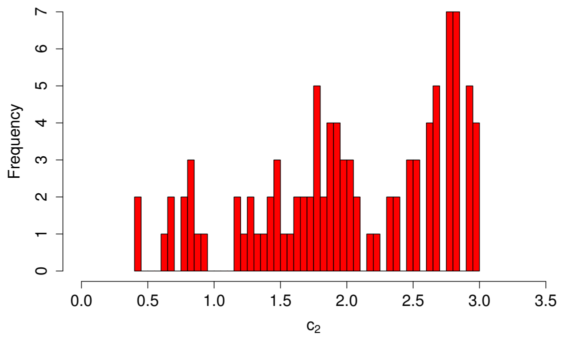
Figure 7.3 shows the distribution of the values for each control parameter over the 100 best-performing parameter configurations. From Figure 7.3a, it is evident that the most frequent well-performing inertia weight was 0.7. There was a very skewed distribution, showing a steady increase in frequency for inertia weight values between -0.5 and 0.7, after which the frequency declined. For the distribution of the cognitive acceleration

coefficient, depicted in Figure 7.3b, a value of approximately 0.9 frequently lead to good performance. Furthermore, there was a noticeable preference for cognitive acceleration coefficients between 0.7 and 1.0. Regarding the distribution of the social acceleration coefficient, larger values were preferred, in general. The highest frequency was observed with social coefficients of approximately 2.75. There is also a notable cluster of good performance when the value of the social acceleration coefficient was between 1.6 and 2.0. These results further demonstrate that, in general, it is preferable to have a larger value for the social coefficient than the cognitive coefficient, despite these two parameter values being traditionally taken as equal in the literature.



(a) Inertia weight.

(b) Cognitive acceleration coefficient.



(c) Social acceleration coefficient.

Figure 7.3: Distribution of the values of the best 100 parameter configurations after 5000 iterations.

Figure 7.4 presents the overall rank of each parameter configuration based on modality. It is evident from Figures 7.4a, 7.4b, 7.4e, and 7.4f that the effects of modality were minimal when equal and social parameter configurations were employed. However, it is notable that for both social and equal parameter configurations, the region that lead to the best performance shifted toward the boundaries of the stable region when faced with multimodal problems. That is, larger acceleration coefficients in the best parameter configurations tended to lead to better performance on multimodal problems. Figures 7.4c and 7.4d indicate that modality had a noticeable effect on the relative performance of cognitive parameter coefficients, specifically the regions that lead to poor performance. While the region leading to the best performance, namely the top boundary of the stable region, did not change significantly, the central region where poor performance was observed (i.e., the darker region) was much larger when faced with unimodal problems. When faced with multimodal problems, the performance of cognitive configurations within the stable region tended to improve as the values of the acceleration coefficients increased. However, when faced with unimodal problems, the cognitive coefficients had two notable regions that lead to good performance, namely the bottom left and top right regions of the stable region, indicating that either a negative inertia weight and small acceleration coefficients, or a large inertia weight and large acceleration coefficients, lead to the best performance.

Figure 7.5 shows the distribution of the values for each control parameter over the 100 best-performing parameter configurations based on modality. When considering the inertia weight, shown in Figures 7.5a and 7.5b, it is evident that unimodal problems had a much smaller range in which the inertia weight can lie while leading to good performance. Notably, multimodal problems had a greater tolerance for negative inertia weights, as evidenced by the longer left tail. Furthermore, Figure 7.5a depicts a higher peak with a larger density surrounding it, suggesting that the deviation of the best inertia weight was much smaller for unimodal problems. When considering the cognitive acceleration coefficient in Figures 7.5c and 7.5d, there is a notable tendency for values between 0.5 and 1 to perform well. When faced with multimodal problems, larger values of the cognitive coefficient were acceptable than when faced with unimodal problems, as evidenced by the longer right tail in Figure 7.5d. Examining the distribution of the social acceleration coefficient, shown in Figures 7.5e and 7.5f, it is evident that larger values (i.e., between 2.5 and 3) had a greater tendency to perform better on unimodal problems, while multimodal problems had a weaker dependence on the value of the social coefficient. However, the best performance on multimodal problems was still obtained within the same region of $[2.5, 3]$. These results suggest that it is preferable to have larger values for the social acceleration coefficient than the cognitive acceleration coefficient, regardless of the modality of the problem.

Table 7.3 presents the best parameter configurations for each environment type, as determined by aggregate rank. For unimodal problems, the best parameter configurations were within the stable region, while the overall best parameter configuration for multimodal problems violated the criterion provided in Equation (2.5). However, it is noteworthy that the average rank was lower for the best parameter configurations in unimodal environments, i.e., where the best parameter configurations were theoretically stable. Specifically, the average rank and standard deviation were much lower for unimodal environments, indicating that the observed best parameter configuration performed well consistently across the unimodal problems. Reinforcing what was observed in Table 7.1, the best configuration for unimodal environments had $c_1 = c_2$, while the best configuration for multimodal environments had a social configuration.

Table 7.3: Best parameter configuration by environment type.

Environment	ω	c_1	c_2	Average Rank	Rank SD
Overall	0.1	0.950	2.850	67.955	43.951
Unimodal	0.5	1.850	1.850	20.333	18.635
Multimodal	-0.1	0.875	2.625	43.846	39.497

7.2.2 The Time-Dependence of Control Parameter Values

To assess the dependence of the optimal parameter region on time, Figures 7.6 to 7.8 present the overall rank of each parameter configuration at various iterations. A key observation is that the region containing the best performing parameter configurations shifted over time. Specifically, as time passed, there was a noticeable preference for larger values of $c_1 + c_2$. Similarly, there was an improvement in the relative performance of parameter sets that lie just outside the stable region, but near the apex. This is evidenced by the emergence of a greater number of light-coloured points outside the stable region, but still near the apex, as the number of iterations increased. This clearly indicates that larger social and cognitive acceleration coefficients were preferred later in the search process. However, this also suggests that the dependence on the stability criterion was weaker as the search progressed.

To further assess the relative performance of parameters over time, Figure 7.9 depicts the best 100 parameter configurations (based on aggregate rank) at various iterations for each of the configuration types. The most important observation is that the best parameter configurations were noticeably different at each of the examined time intervals. Specifically, there was a tendency for parameters to shift towards the right over time, implying that larger values of the social and cognitive parameters were preferred later in the search process. In contrast, the results suggest that the inertia weight value was less dependent on time than the acceleration coefficients, as the optimal parameter region shifted much more horizontally than vertically. This result provides direct evidence against dynamically reducing the inertia weight over time, and provides further evidence in support of the findings in Chapter 5, where it was found that linearly decreasing inertia weight strategies performed worse than a constant inertia weight. An additional, noteworthy observation is that there was an inherent relationship between performance, time, and adherence to the stability criterion. Adherence to the stability criterion was less important as time passed; parameter configurations that violated the stability criterion, but still performed relatively well, were more frequent as the search progressed. This is likely a result of the larger variances in particle positions resulting from parameter configurations that are near, or even outside, the stable region [9]. A large variance corresponds to larger particle step sizes, which means that complete stagnation is less likely. Therefore, parameter configurations with larger variances may be preferred later in the search due to their prevention (or delaying) of stagnation. The observed time-dependence of control parameter values further emphasizes the importance of developing efficient SAPSO algorithms.

Another observation from Figure 7.9 reinforces that the region containing the best parameters was noticeably different based on the balance between the social and cognitive parameters. For parameter configurations that are considered social, the optimal region formed a cluster that closely resembles the apex of the stable region. However, when cognitive parameter configurations were employed, the best parameter configurations formed two distinct clusters. Regarding the balanced parameter sets, the region containing the best configuration was somewhat a mix of the two regions described above. For all three parameter schemes, there was a visible preference for positive values of the inertia weight. This observation was least prominent for the social configuration, likely as a result of the increased influence of the global best position. To illustrate this point, consider a cognitive parameter configuration, where the movement direction of a particle is most prominently influenced by its own personal best position. Thus, it is reasonable to assume a high degree of correlation between the direction of subsequent particle movements. That is, if a particle is moving in one direction and finds a new personal best solution, it is likely to continue in the same direction. Therefore, a negative inertia weight would be rather detrimental to a cognitive parameter configuration. Conversely, a social solution takes a high degree of influence from the remainder of the swarm, and thus is subject to have less correlation between the direction of subsequent movements.

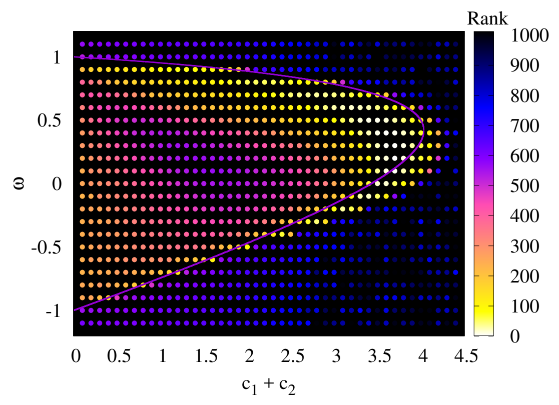
Figures 7.10 to 7.12 show the distribution of the 100 control parameter values that lead to the best performance at various time intervals. Considering the distribution of the inertia weight values, shown in Figure 7.10, there was little change in the distribution after 1000 iterations, suggesting that the best inertia weight values to employ were within the range $[0.4, 0.8]$ but did not, in general, change as the search progresses. This provides further evidence to suggest that decreasing inertia weight strategies are suboptimal. For each of the iterations examined, the most frequent well-performing cognitive control parameter values, shown in Figure 7.11, were within the range of $[0.5, 1.0]$. However, the value of the cognitive control parameter that most frequently lead to the best performance showed a slight increase as the search progresses. After 500 iterations, cognitive acceleration values near 0.8 occurred most frequently in the 100 best parameter configurations, while the most frequent well-performing cognitive acceleration coefficient increased to approximately 0.9 after 5000 iterations. The distribution of the best-performing social acceleration coefficients, shown in Figure 7.12, depicts a similar trend to the cognitive acceleration coefficient. After 500 iterations, values of the social acceleration coefficient that were approximately 2.6 lead to good performance most frequently, while a social acceleration value of approximately 2.8 tended to most frequently lead to good performance after 5000 iterations. In general, the best performance was observed when a social acceleration coefficient, within the range of $[2.5, 3.0]$, was employed, further suggesting that parameter configurations favouring the social influence tended to perform better than those that favour a cognitive influence. Based on these results, a general guideline for selecting PSO control parameter values, which should lead to reasonable performance regardless of the number of iterations, is to set $\omega \in [0.4, 0.8], c_1 \in [0.5, 1.0]$, and $c_2 \in [2.5, 3.0]$.

7.3 Summary

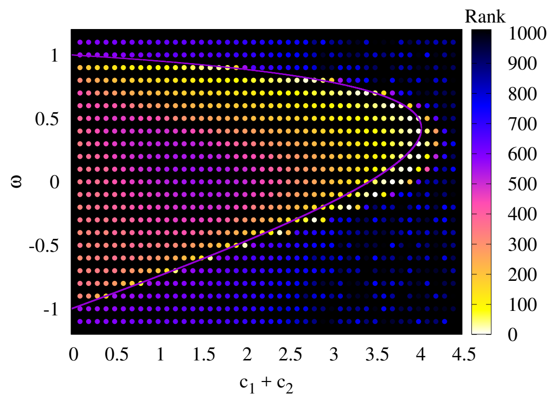
This chapter provided an empirical investigation into the relative performance of PSO parameter configurations. The overall objective was to identify the regions of parameter space that lead to the best performance for the global best topology. Specifically, two important questions regarding the parametrization of a global-best PSO were addressed. Firstly, the question of where the optimal parameter configurations reside when the respective values of the acceleration coefficients were different was examined. Secondly, this study examined the question of whether the optimal parameters to employ are dependent on time. To investigate these questions, a total of 3036 parameter configurations were examined on a set of 22 benchmark functions. The modality of the benchmark problems was examined to ascertain whether the optimal regions of parameter space were dependent on the modality of the problem. Furthermore, the results were correlated with the best-known stability criterion to determine whether particle stability had any effect on the performance of the PSO algorithm.

To address the first question, the experiments were divided into three categories based on the ratio between the social and cognitive acceleration coefficients (i.e., equal, larger social, and larger cognitive). The results indicated that the regions leading to the best performance were notably different among each of the different types of parameter configurations. That is, the balance between social and cognitive coefficients does have a significant effect on the regions of parameter space that lead to optimal performance. To address the second question, the performance of the parameter configurations was captured at various points throughout the search. Results indicated that the optimal values for the acceleration coefficients increase as the search progresses, irrespective of the balance between the social and cognitive coefficients. Specifically, this indicates that the optimal parameters are, in fact, time-dependent, thereby providing further justification for SAPSO algorithms that can alter the values of their control parameters over time. Despite the observed dependence on time, a general recommendation for selecting values for the PSO control parameters is to set $\omega \in [0.4, 0.8], c_1 \in [0.5, 1.0]$, and $c_2 \in [2.5, 3.0]$. It is noted that the findings in Chapter 6 suggest the results may be different if these experiments were repeated using the local best topology.

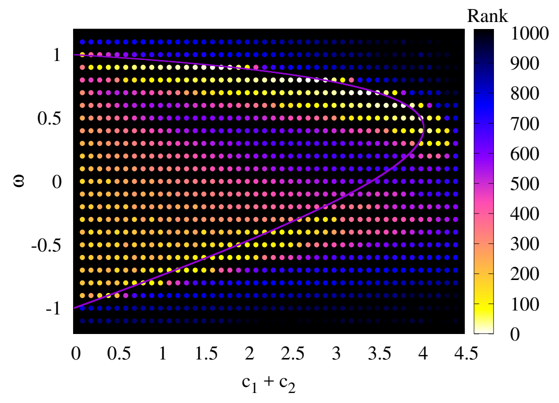
Examining Figure 7.9, it is evident that certain regions of parameter space lead to better performance. In the next chapter, a new region for sampling PSO parameter values is proposed based on the findings in this chapter. Moreover, a new PSO algorithm, which adapts its control parameters over time using the proposed region, is also designed.



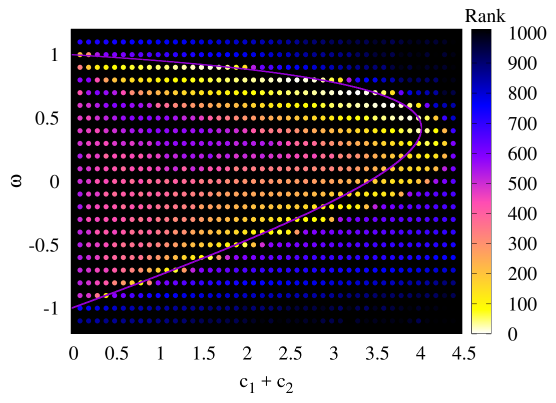
(a) Social configurations, unimodal problems.



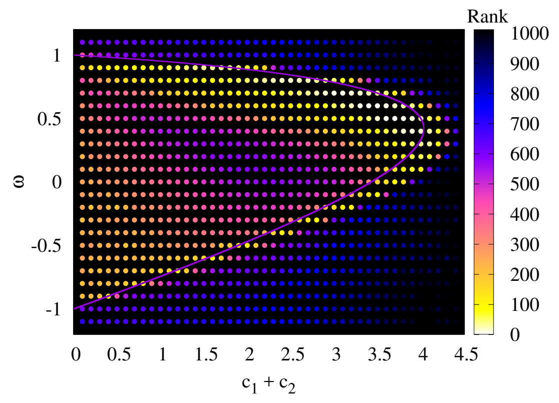
(b) Social configurations, multimodal problems.



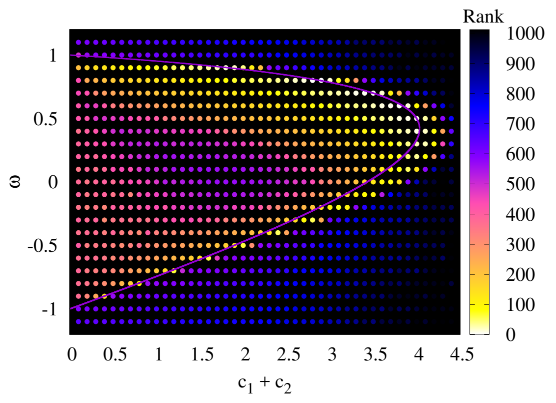
(c) Cognitive configurations, unimodal problems.



(d) Cognitive configurations, multimodal problems.

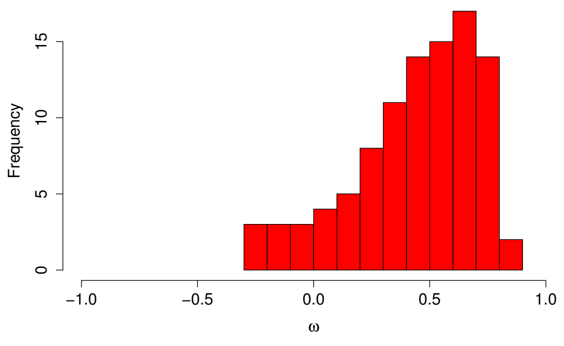


(e) Equal configurations, unimodal problems.

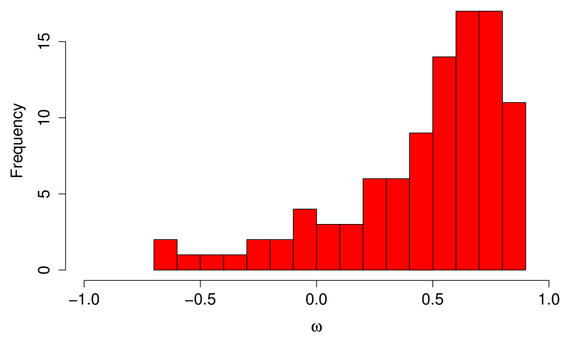


(f) Equal configurations, multimodal problems.

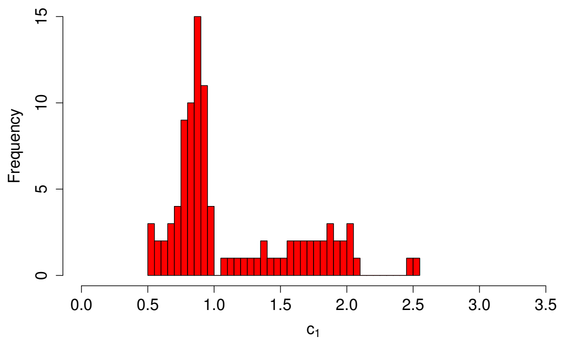
Figure 7.4: Overall rank after 5000 iterations based on modality.



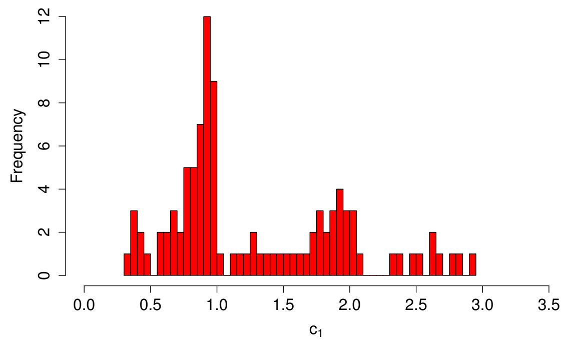
(a) Inertia weight, unimodal problems.



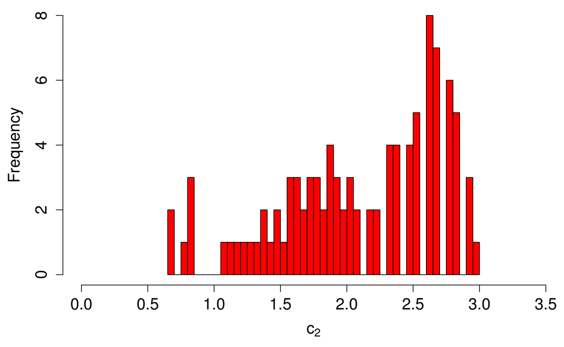
(b) Inertia weight, multimodal problems.



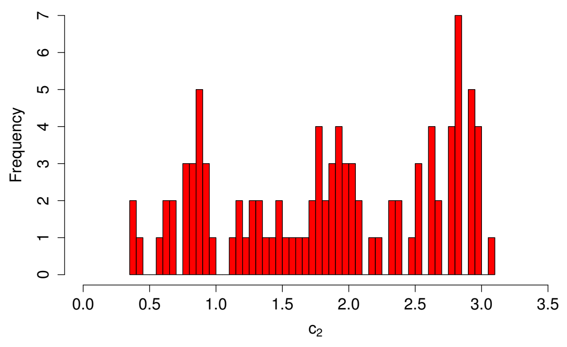
(c) Cognitive acceleration coefficient, unimodal problems.



(d) Cognitive acceleration coefficient, multimodal problems.

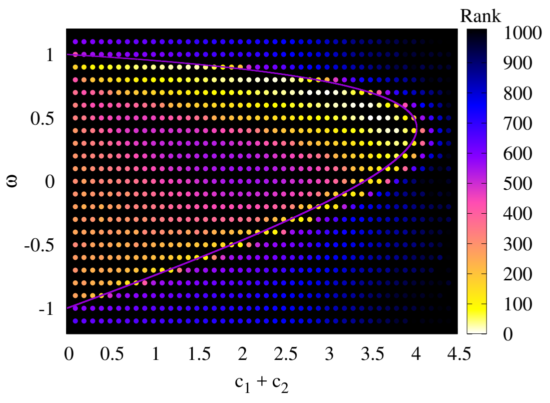


(e) Social acceleration coefficient, unimodal problems.

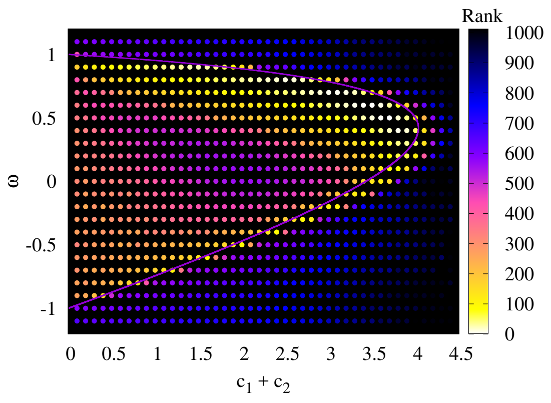


(f) Social acceleration coefficient, multimodal problems.

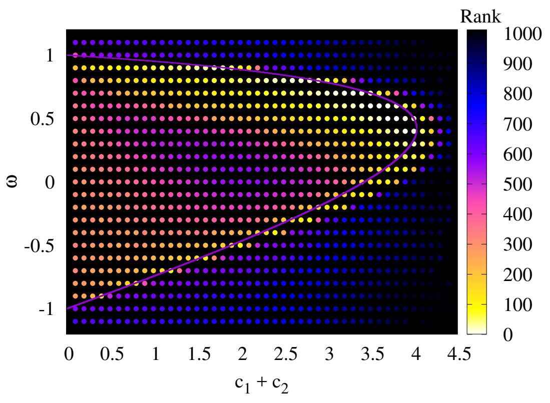
Figure 7.5: Distribution of the values of the best 100 parameter configurations after 5000 iterations based on modality.



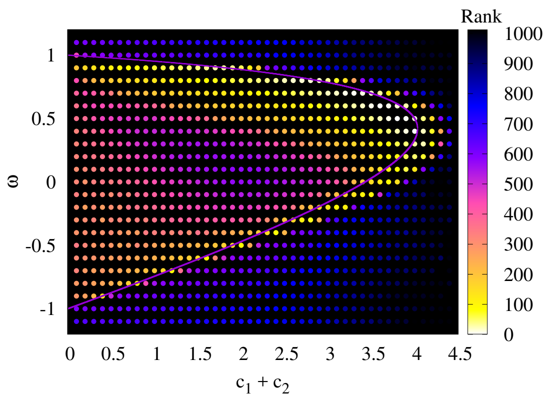
(a) 500 Iterations.



(b) 1000 Iterations.

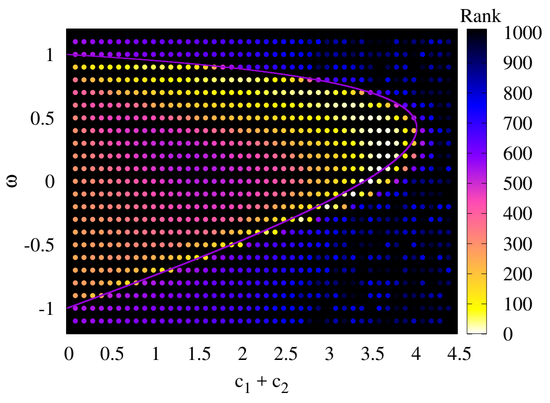


(c) 2500 Iterations.

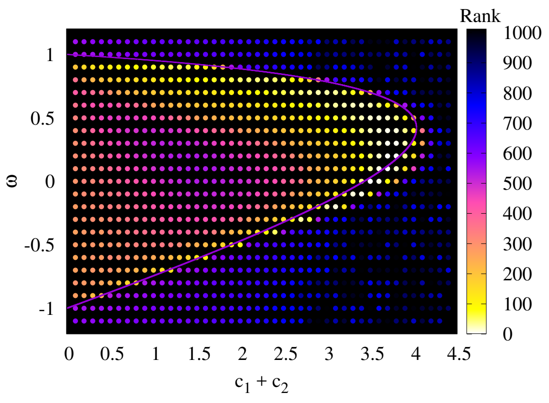


(d) 5000 Iterations.

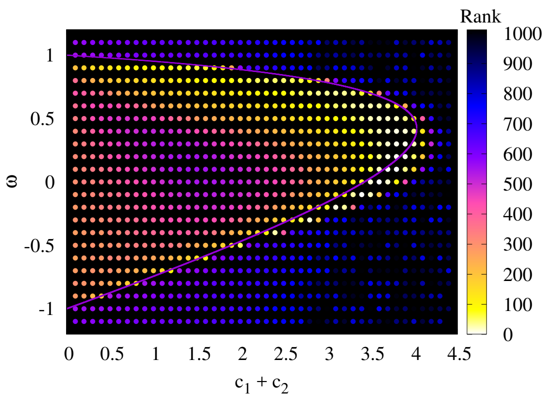
Figure 7.6: Overall rank at various iterations with $c_1 = c_2$.



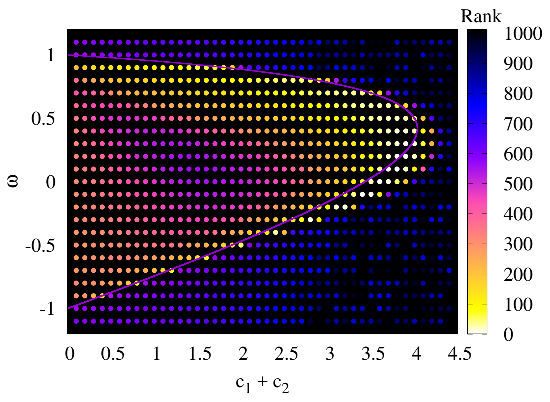
(a) 500 Iterations.



(b) 1000 Iterations.

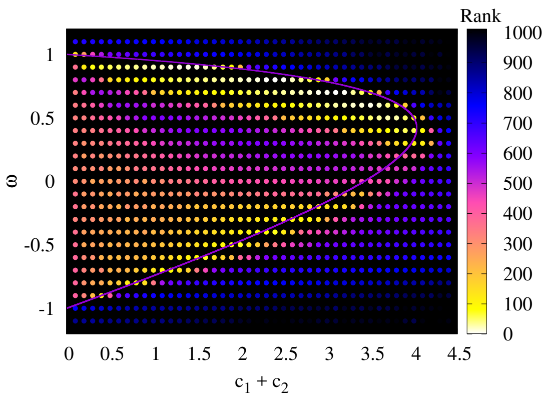


(c) 2500 Iterations.

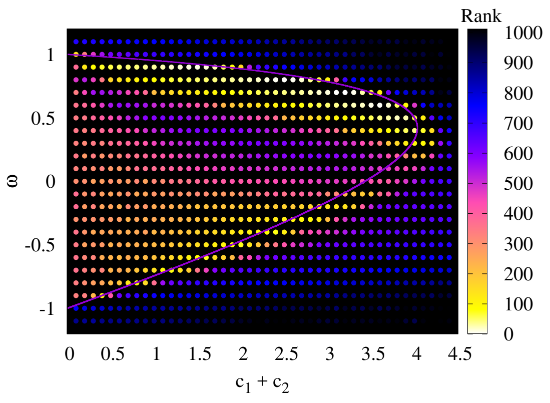


(d) 5000 Iterations.

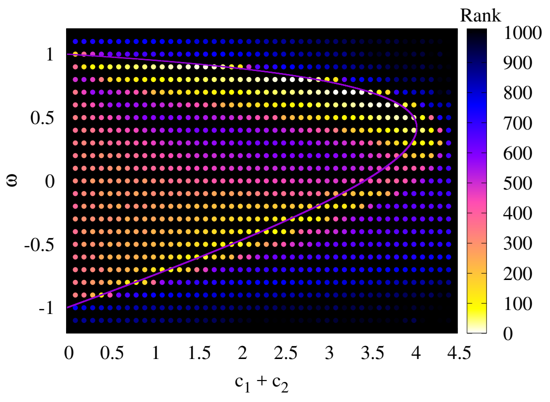
Figure 7.7: Overall rank at various iterations with a social configuration.



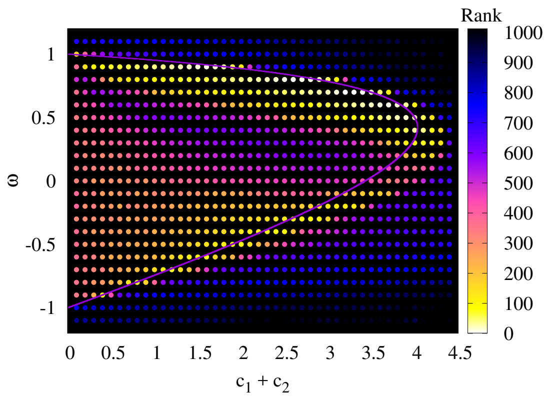
(a) 500 Iterations.



(b) 1000 Iterations.

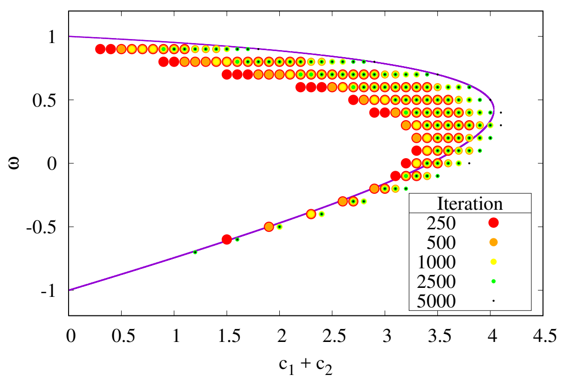


(c) 2500 Iterations.

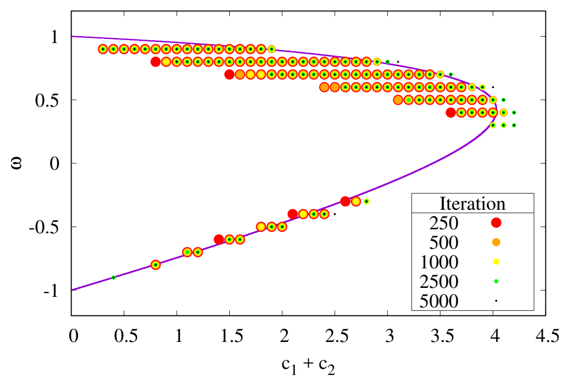


(d) 5000 Iterations.

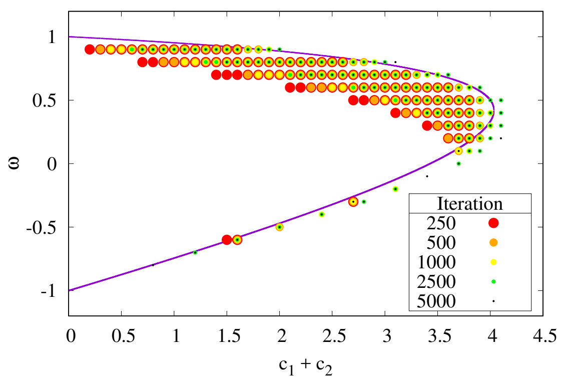
Figure 7.8: Overall rank at various iterations with cognitive configurations.



(a) Social configurations.

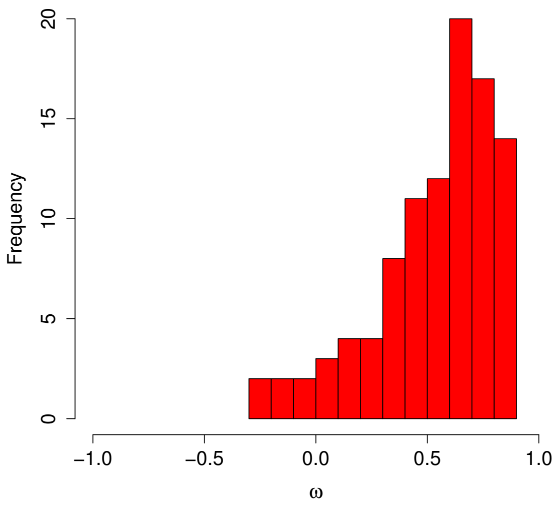


(b) Cognitive configurations.

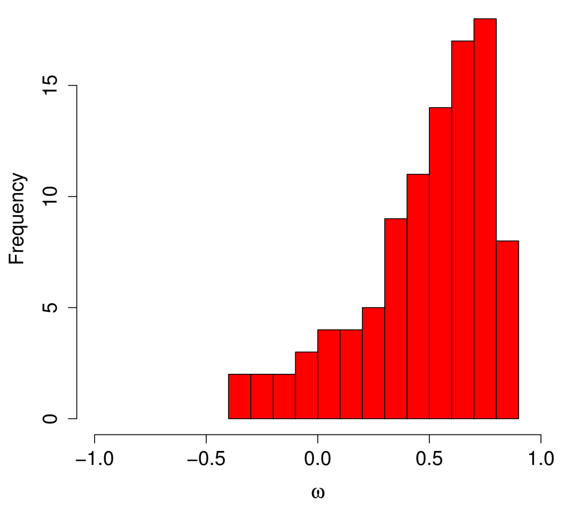


(c) Equal configurations.

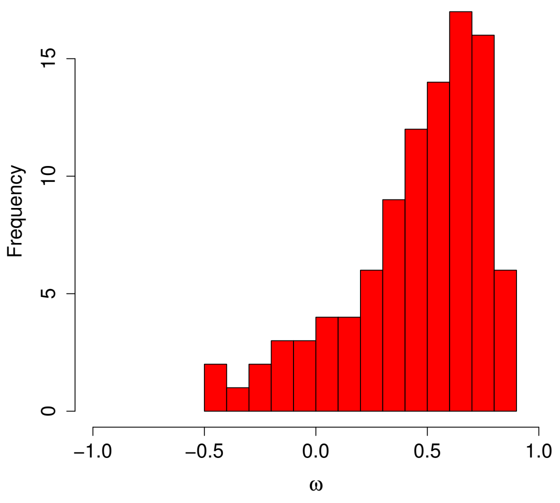
Figure 7.9: 100 best parameter configurations at various iterations.



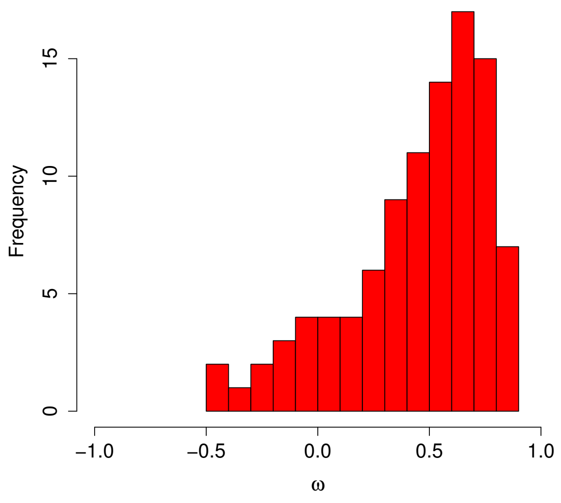
(a) 500 Iterations.



(b) 1000 Iterations.

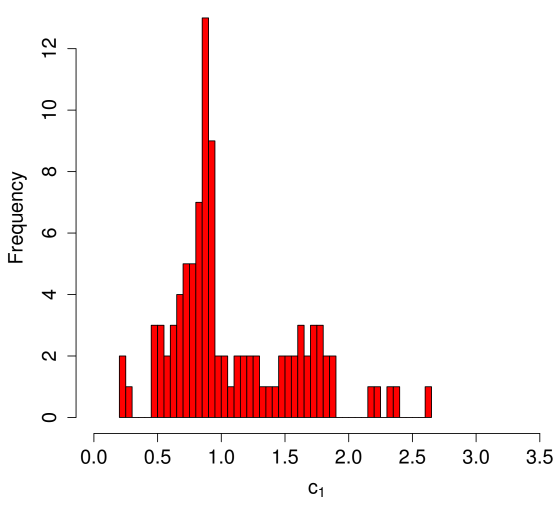


(c) 2500 Iterations.

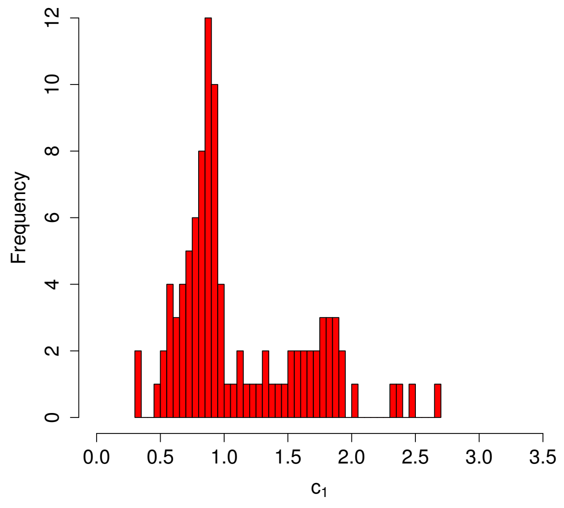


(d) 5000 Iterations.

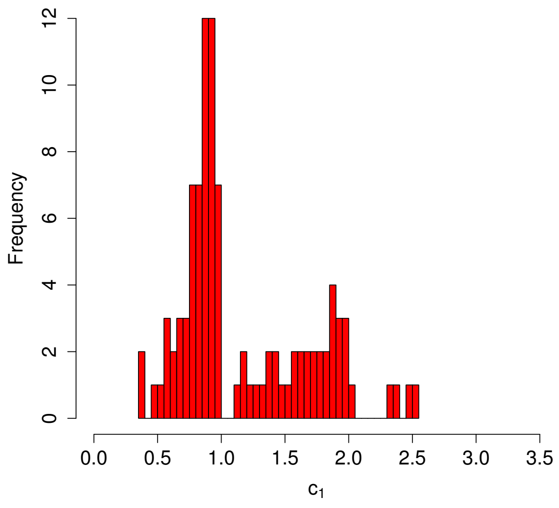
Figure 7.10: Distribution of the inertia weight values of the best 100 parameter configurations at various iterations.



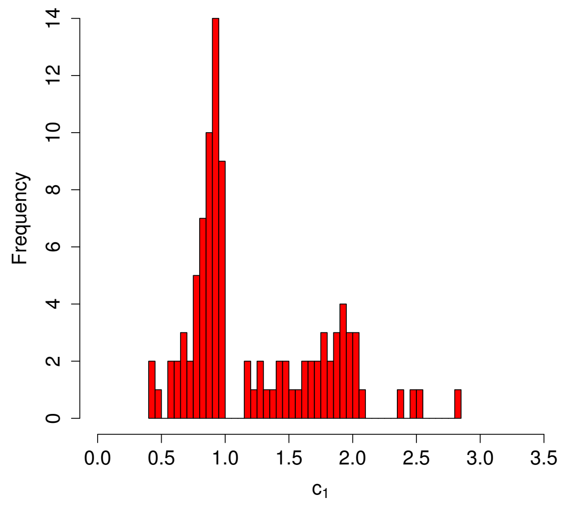
(a) 500 Iterations.



(b) 1000 Iterations.

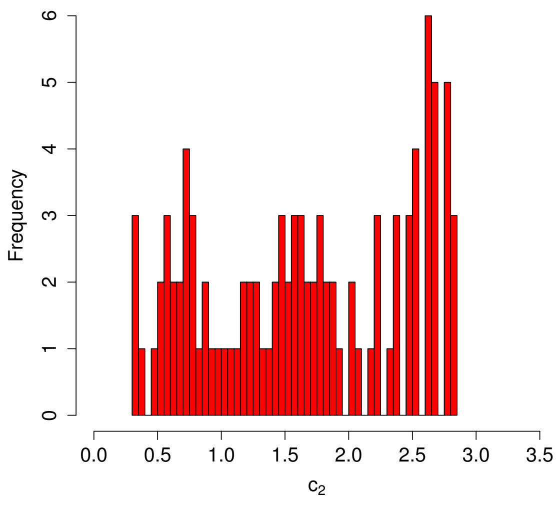


(c) 2500 Iterations.

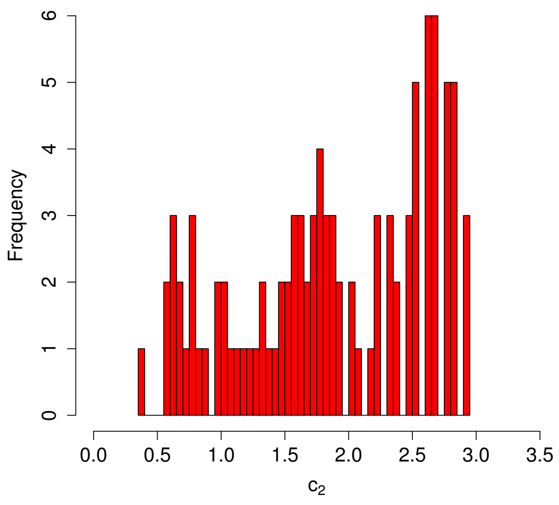


(d) 5000 Iterations.

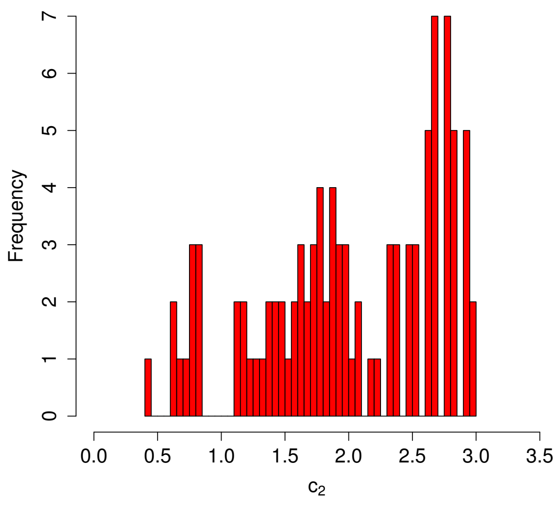
Figure 7.11: Distribution of the cognitive acceleration values of the best 100 parameter configurations at various iterations.



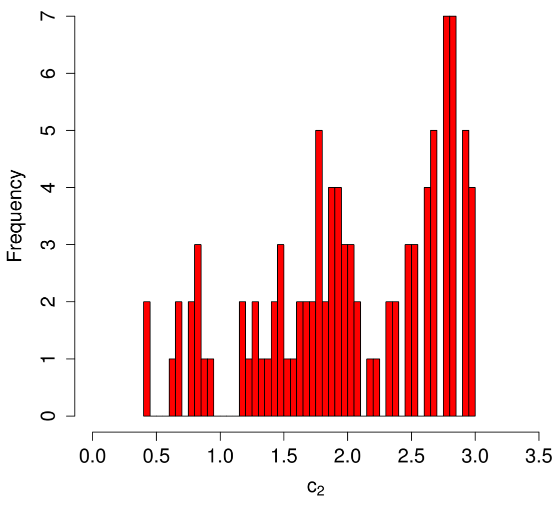
(a) 500 Iterations.



(b) 1000 Iterations.



(c) 2500 Iterations.



(d) 5000 Iterations.

Figure 7.12: Distribution of the social acceleration values of the best 100 parameter configurations at various iterations.

Chapter 8

An Adaptive Particle Swarm Optimization Algorithm Based on Optimal Parameter Regions

This chapter proposes a new region for the selection of PSO parameters based on the results in Chapters 6 and 7. Furthermore, a PSO variant that randomly samples control parameter values from the proposed region is also introduced. Two different strategies dictating when the control parameter values are updated are also examined. The proposed PSO variant is then compared to the standard PSO employing 14 different parameter configurations suggested in the literature across a suite of 60 benchmark problems. Furthermore, various landscape characteristics are examined to ascertain the effects of the environment type on the results.

The remainder of this chapter is structured as follows. Section 8.1 provides an overview of studies that provide recommendations on the setting of PSO parameter configurations. In Section 8.2, a new region for parameter selection is constructed and an algorithm exploiting this region is proposed. Section 8.3 describes the experimental design used to compare the performance of the proposed algorithm, the results of which are presented in 8.4. Finally, Section 8.5 provides a summary of this chapter's findings.

8.1 Parameter Recommendations

This section reviews a number of studies that have examined and suggested new PSO parameter configurations. Each of these studies were selected based on either the prevalence of their suggested parameters in the literature, or for their wide variety of parameter settings examined. Therefore, the studies and parameter settings described below represent what a practitioner would find when searching for suggested parametrizations of the PSO algorithm.

One of the most popular parametrizations for PSO, where $\omega = 0.7298$ and $c_1 = c_2 = 1.49618$, is attributed to the early work of Eberhart and Shi [28], where the constriction factor PSO [22] and inertia weight PSO [98] were compared. Though it was concluded that the constriction factor PSO was superior, the two variations are functionally equivalent. This conclusion was likely due to the inertia model employing a decreasing inertia weight strategy, which was shown in Chapter 5 to be detrimental to PSO performance.

Carlisle and Dozier [12] examined various parameter configurations of the constriction factor PSO, with the goal of creating a general purpose, "off-the-shelf" PSO variant. Carlisle and Dozier [12] discovered the best performing parameters of those examined were $\phi_1 = 2.8$, $\phi_2 = 1.3$, and $\phi = 4.1$, which translates to $\omega = 0.729$, $c_1 = 2.0412$, and $c_2 = 0.9477$ in the inertia weight model.

Trelea [105] used dynamic system theory to analyse the theoretical behaviours of various PSO control parameter values. The authors empirically examined a number of parameter sets[1], where $\omega = 0.6$ and $c_1 = c_2 = 1.7$ were selected as the best performing parameter values. The aforementioned parameter set was reported to outperform the best known parameters at that time.

Clerc [23] examined the stagnation behaviour of PSO and found the most promising parameters, with regards to stagnation behaviour, to be $\omega = 0.721$, and $c_1 + c_2 = 2.386$. For the purposes of this study, it is assumed that $c_1 = c_2 = 1.193$.

Jiang *et al.* [51] examined the relationship between the speed of convergence toward a fixed location and particle trajectories, which lead to the derivation of a new theoretical model of particle stability. Based on their derived model and selection guidelines, a set

[1]The exact number of parameter sets examined was not explicitly given.

of parameters, $\omega = 0.715$ and $c_1 = c_2 = 1.7$, was suggested.

Zhang *et al.* [121] empirically examined the dynamic characteristics of PSO, ultimately leading to a new parameter selection technique. Using their proposed technique, Zhang *et al.* [121] proposed three sets of parameters, each of which performed similarly. The three suggested parameter settings were as follows:

- $\omega = 0.724$ and $c_1 = c_2 = 1.468$
- $\omega = 0.785$ and $c_1 = c_2 = 1.331$
- $\omega = 0.837$ and $c_1 = c_2 = 1.255$

Liu [71] derived an order-2 stable region for PSO using a weak stagnation assumption. This region was subsequently used as the basis for parameter selection. Liu [71] then empirically examined 14 parameter configurations, of which $\omega = 0.42$ and $c_1 = c_2 = 1.55$ lead to the best performance.

Bonyadi and Michalewicz [9] examined the impact of PSO parameters on the movement pattern of particles. Bonyadi and Michalewicz [9] then empirically examined 11 parameter sets (six of which they proposed) and found that setting $\omega = 0.711897$ and $c_1 = c_2 = 1.711897$ lead to the overall best performance.

The two best performing parameter configurations found in Chapter 6 were $\omega = 0.5, c_1 = c_2 = 1.90$ and $\omega = 0.6, c_1 = c_2 = 1.80$. However, this study was limited to parameter configurations in which $c_1 = c_2$. In the expanded study of Chapter 7, the best two parameter configurations were found to be $\omega = 0.1, c_1 = 0.95, c_2 = 2.85$ and $\omega = -0.1, c_1 = 0.875, c_2 = 2.625$.

8.2 Proposed Adaptive Particle Swarm Optimization Algorithm

Chapter 7 found that the best performing values for the control parameters, with respect to solution accuracy, tended to be clustered near the boundaries of the stable region, as visualized in Figure 8.1 alongside Poli's stability criterion. Figure 8.1 shows the best

100 parameter configurations for each experiment type, namely 'Equal' where $c_1 = c_2$, 'Cognitive' where $c_1 = 3c_2$, and 'Social' where $3c_1 = c_2$.

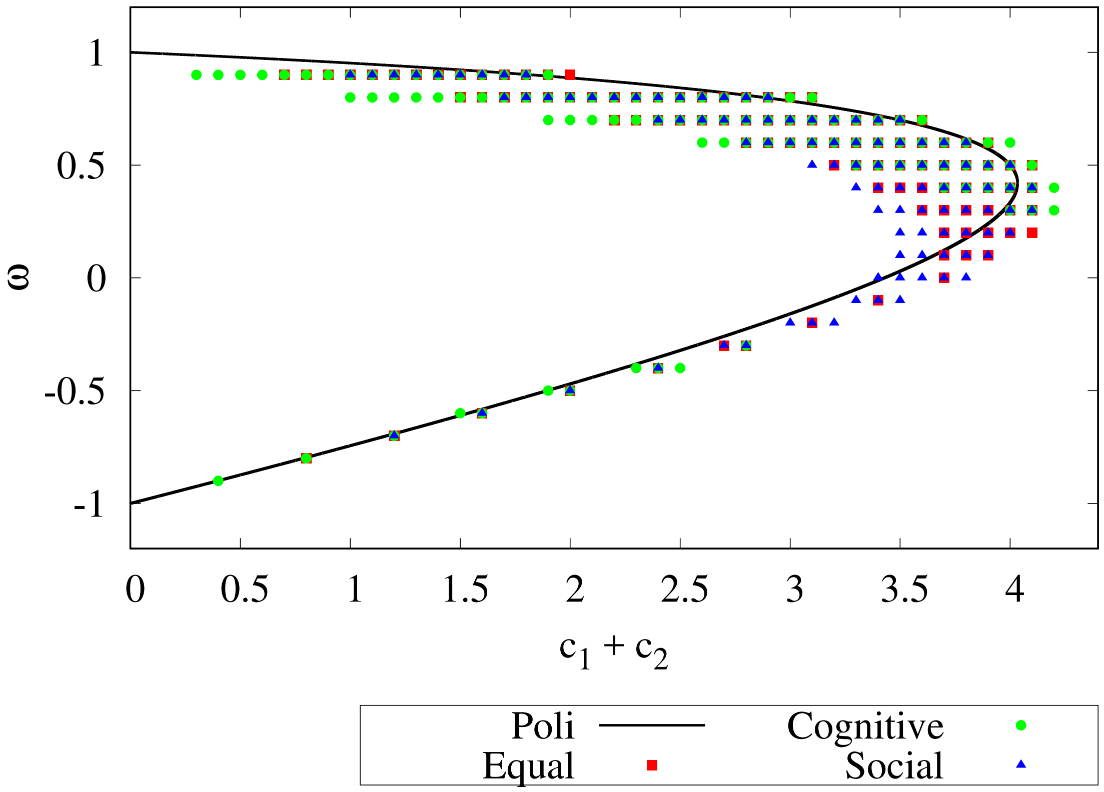


Figure 8.1: Visualization of the best performing parameter values from Chapter 7. 'Equal', 'Cognitive', and 'Social' represent the experiments with differing ratios between the cognitive and social acceleration coefficients.

It is evident from Figure 8.1 that there is a relatively small region where the performance is superior in terms of accuracy (assuming a fixed computational budget). By manipulating the coefficients from Equation (2.5) and reversing the inequality such that

$$c_1 + c_2 > \frac{22 - 30\omega^2}{7 - 5\omega}, \tag{8.1}$$

a reasonable lower bound for parameter selection can be attained. The proposed region, from which to select PSO control parameter values, is then constructed by taking Equation (2.5) as an upper bound and Equation (8.1) as a lower bound. The proposed region, constrained by

$$\frac{22 - 30\omega^2}{7 - 5\omega} < c_1 + c_2 < \frac{24 - 24\omega^2}{7 - 5\omega}, \tag{8.2}$$

is visualized in Figure 8.2 along with the best found parameters from Chapter 7. Note that the right side of Equation (8.2) is simply Equation (2.5) with the numerator expanded.

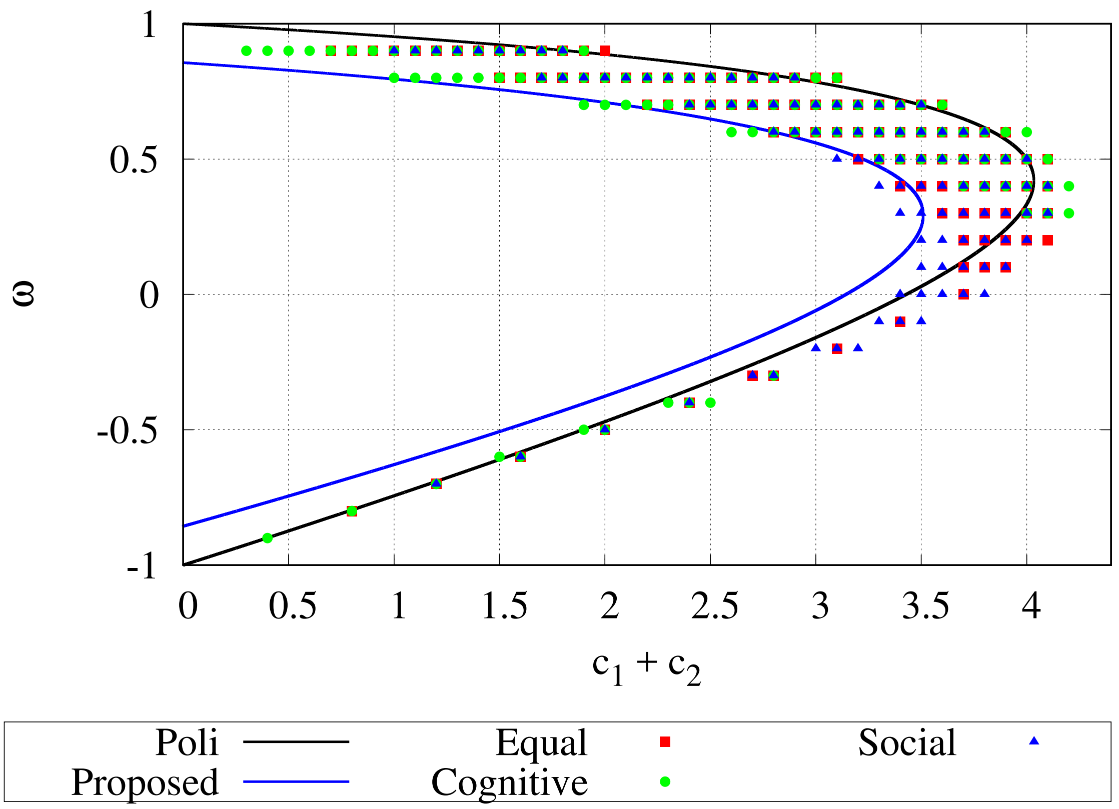


Figure 8.2: Proposed lower bound containing the best performing parameter values.

The proposed algorithm, namely particle swarm optimization with improved random constants (PSO-iRC), initializes each particle with a unique set of control parameter values by uniformly sampling the region constrained by Equation (8.2). The control parameter values are then re-sampled uniformly from this region throughout the search process according to one of two strategies:

- PSO-iRC-pk, where new control parameter values are selected for each particle after every k iterations.

- PSO-iRC-sk, where new control parameter values are selected for a particle if its personal best position has stagnated (i.e., no better personal best position has been found) for k iterations.

Note that, for both of the proposed variants, there is no need to specify values for the traditional PSO control parameters, namely ω, c_1, and c_2. Rather, only the frequency of parameter value updates (i.e., the value of k) must be specified. Therefore, the proposed algorithm leads to an overall reduction in the number of parameters required for the PSO algorithm.

8.3 Experimental Setup

This section describes the experiments that examine the performance of PSO-iRC relative to the standard PSO. The various PSO parameter configurations introduced in Section 8.1, which are used for comparison, are summarized in Table 8.1. For both PSO-iRC-pk and PSO-iRC-sk, the examined values of k were $\{1, 5, 10, 25\}$. All examined PSO variants consisted of 30 particles arranged in a global-best topology and used a synchronous update strategy. Initial particle positions were randomly sampled within the feasible bounds of the search space, while particle velocities were initially set to the zero vector [30]. To prevent infeasible attractors, a particle's personal best position was only updated if a new position had a better objective function value and was within the feasible bounds of the search space. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. The objective fitness value averaged over 50 independent runs, each consisting of 5000 iterations, was taken as the measure of performance for each algorithm. As described in Appendix A, a suite of 60 minimization problems were used in this study. All functions were optimized in 30 dimensions.

Statistical analysis of results was done by way of Friedman's test for multiple comparisons among all methods [35, 36], as recommended by Derrac *et al.* [26]. Furthermore, Shaffer's post-hoc procedure [97] was performed as a means to identify the pairwise

Table 8.1: Control parameter values

ID	ω	c_1	c_2	Reference
PSO-1	0.7298	1.49618	1.49618	[28]
PSO-2	0.729	2.0412	0.9477	[12]
PSO-3	0.6	1.7	1.7	[105]
PSO-4	0.721	1.193	1.193	[23]
PSO-5	0.715	1.7	1.7	[51]
PSO-6	0.724	1.468	1.468	[121]
PSO-7	0.785	1.331	1.331	[121]
PSO-8	0.837	1.255	1.255	[121]
PSO-9	0.42	1.55	1.55	[71]
PSO-10	0.711897	1.711897	1.711897	[9]
PSO-11	0.5	1.90	1.90	[43] (Ch. 6)
PSO-12	0.6	1.80	1.80	[43] (Ch. 6)
PSO-13	0.1	0.950	2.850	[45] (Ch. 7)
PSO-14	-0.1	0.875	2.625	[45] (Ch. 7)

comparisons that produced significant differences. Finally, critical difference plots were generated, whereby algorithms that are to the left of the plot demonstrated superior performance, and algorithms that are grouped by a line were found to have insignificantly different performance.

8.4 Results and Discussion

Figure 8.3 presents the critical difference plot when considering performance across all 60 benchmark problems. The first important observation is that the best overall performance was insignificantly different among 15 of the 22 configurations when considering all benchmark problems. Notably, five of the eight variants of PSO-iRC were among the top-performing group, suggesting that PSO-iRC is an effective solution for setting PSO control parameter values. Furthermore, the relative positioning of the PSO-iRC-pk variants (i.e., the PSO-iRC-pk variants were all further left than the PSO-iRC-sk variants) suggests that periodic parameter updates lead to better performance than updating parameter values only when the personal best position stagnates.

From the large group sizes shown in Figure 8.3 (i.e., the large number of algorithms that are grouped with a line), it can be concluded that the performance of various PSO control parameter values was problem-dependent, thus causing no particular configuration to outperform all others. In general, the best performing parameters, despite the insignificant differences noted for the top 15 configurations, were those employed by PSO-2 and PSO-7, while the worst performing parameters were those employed by PSO-9 and PSO-4. Considering the PSO-iRC variants, the best parameter update strategy was to generate new parameters every five iterations. To ascertain the effects of various landscape characteristics on the performance of PSO-iRC, the following subsections present the results for various environment types.

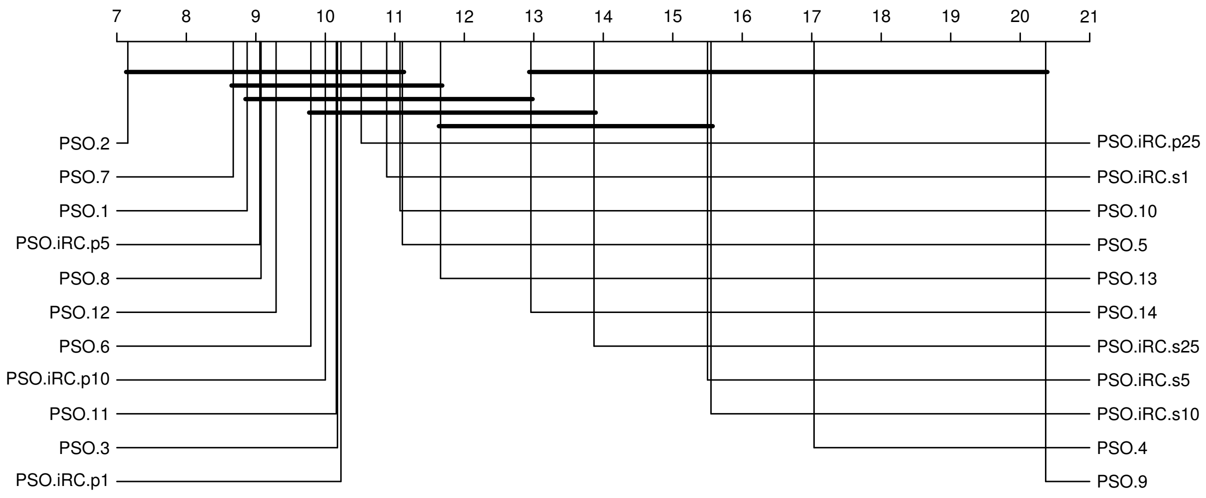


Figure 8.3: Critical difference plot, all functions.

8.4.1 Modality

The critical difference plot with regards to the 19 unimodal problems is shown in Figure 8.4. Three of the variants of PSO-iRC were among the top performing group, all of which were PSO-iRC-pk. For unimodal functions, the best performance was attained by PSO-3 and PSO-1 while the worst performance was again demonstrated by PSO-9, followed by PSO-4.

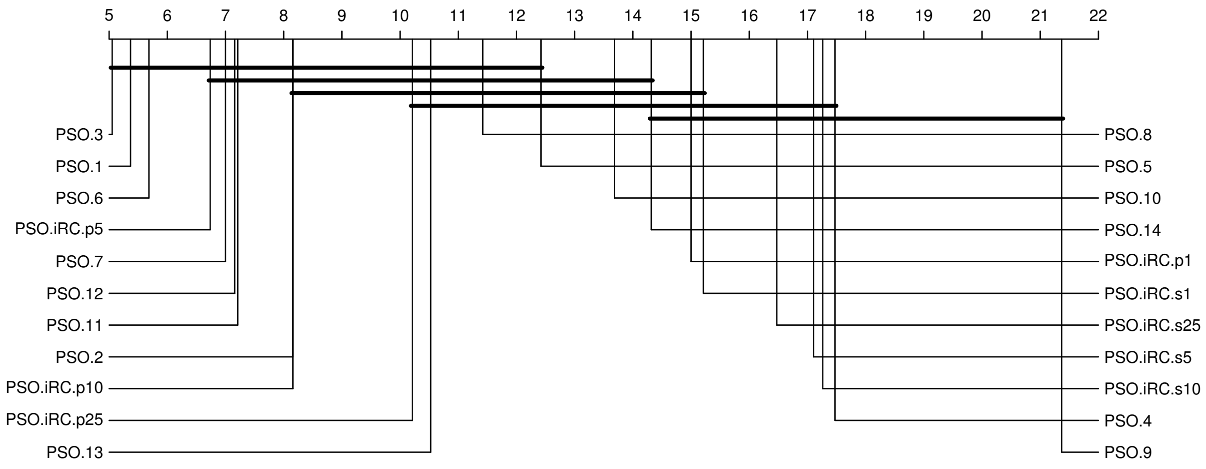


Figure 8.4: Critical difference plot, unimodal functions.

Considering the critical difference plot for the 41 multimodal problems, as shown in Figure 8.5, five of the eight PSO-iRC variants shared the best performance. In contrast to the unimodal problems, where only PSO-iRC-pk variants showed the best performance, PSO-iRC-s1 was also among the top performing configurations. For multimodal problems, the best performance was demonstrated by PSO-2, while the worst performance was shown by PSO-9.

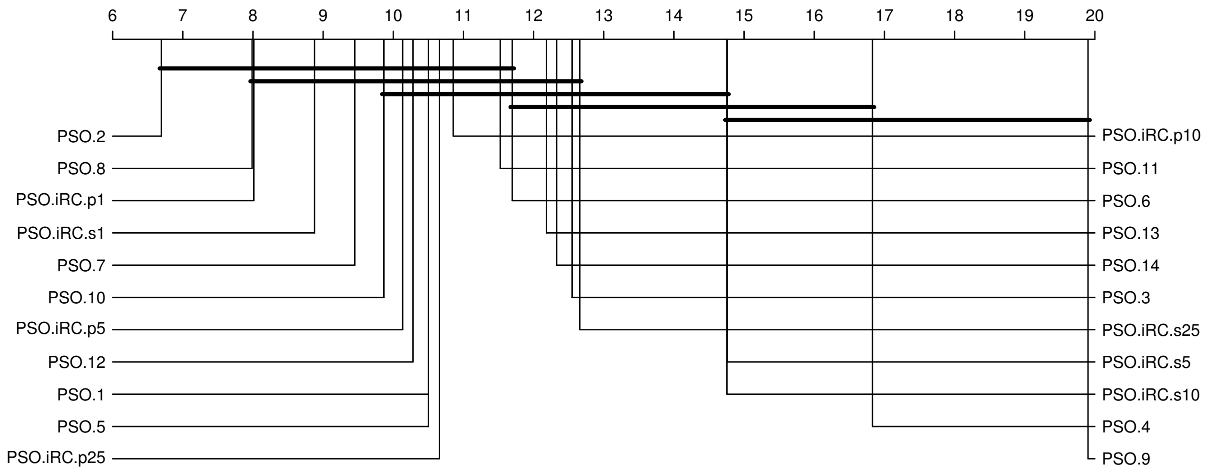


Figure 8.5: Critical difference plot, multimodal functions.

8.4.2 Separability

Considering the critical differences across the 21 separable problems, as shown in Figure 8.6, it is noted that there was an insignificant difference in performance across 20 of the 22 examined configurations. The best performance was attained by PSO-2 and PSO-iRC-p5, while the worst performance was noted for PSO-9 and PSO-4.

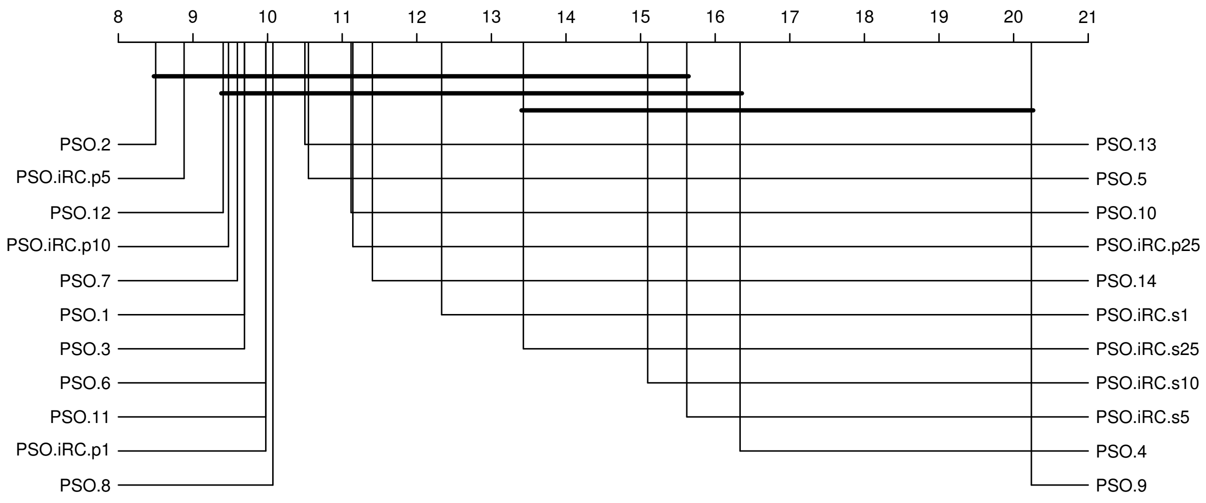


Figure 8.6: Critical difference plot, separable functions.

Figure 8.7 shows the critical difference plot when considering the 39 non-separable problems. Of the 15 best performing algorithms, four were variants of PSO-iRC. When considering non-separable environments, PSO-2 lead to the best performance, while PSO-9 lead to the worst performance.

8.4.3 Composition Functions

Figure 8.8 presents the critical difference plot when considering the 11 composition problems. Figure 8.8 shows that the best performance was shared among 14 of the 22 configurations, eight of which were the PSO-iRC configurations. Thus, all of the PSO-iRC variants, independent of the parameter switching mechanism, were highly effective on the composition problems, which can be considered the most challenging type of problem examined. An additional noteworthy observation was that the best performing PSO-iRC variant was PSO-iRC-s1. This is in contrast to every other type of environment,

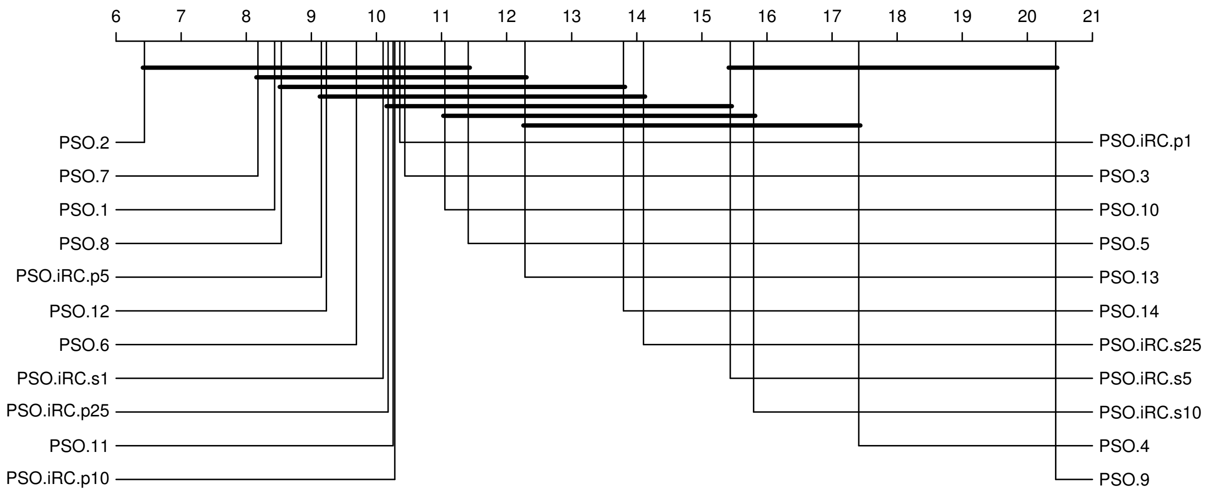


Figure 8.7: Critical difference plot, non-separable functions.

where PSO-iRC-pk had shown better performance. For composition problems, PSO-2 was again the best performing parameter configuration, while PSO-13 lead to the worst performance.

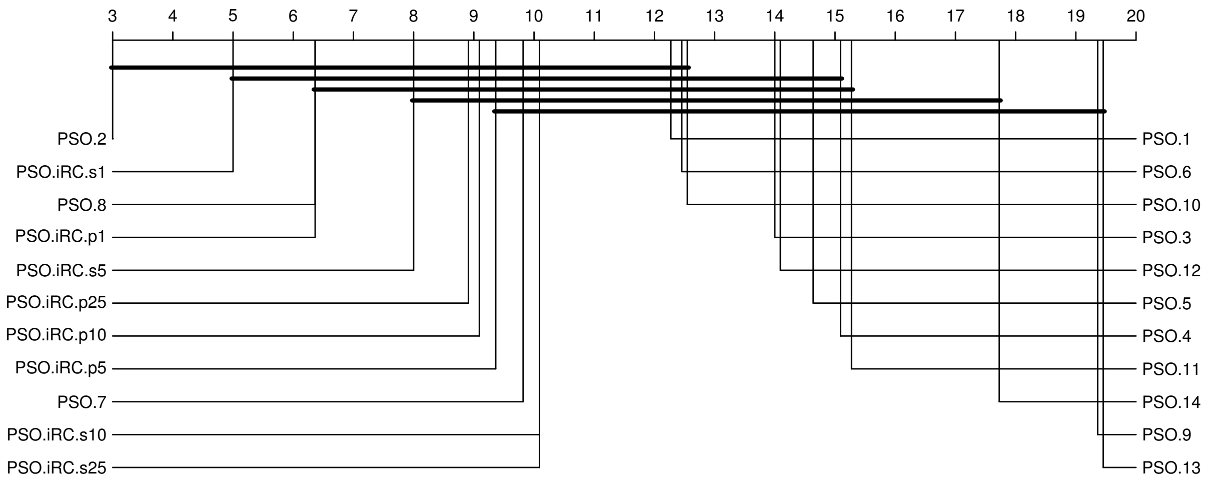


Figure 8.8: Critical difference plot, composition functions.

8.5 Summary

This chapter presented a novel PSO variant, namely PSO-iRC, based on random sampling of control parameter values from a region in parameter space known to contain promising configurations. Two variants of PSO-iRC, which differed in when the values of the control parameters were updated, were proposed. The two variants of PSO-iRC were compared to 14 different parametrizations of PSO, as suggested by the literature, across a suite of 60 benchmark problems. The results were also examined based on various landscape characteristics to ascertain the effects of different environment types.

Firstly, it is evident that PSO-iRC-pk is an effective solution for setting PSO parameters throughout the search. For nearly all environment types, PSO-iRC-p1, PSO-iRC-p5, and PSO-iRC-p10 were among the best performing configurations. However, it is noted that the best performance was shared among a fair number of configurations, irrespective of the environment type considered. This result reinforces that parametrization of the PSO algorithm is highly problem dependent, and therefore the performance differences of the well-performing parameter settings were not very drastic when considered across a number of problems. Nonetheless, PSO-iRC-pk demonstrated consistently good performance, irrespective of the environment type, and further benefits from the reduction in the number of control parameters.

Another notable observation was that the performance of PSO-2, despite being insignificantly different from a number of other configurations, was generally superior. Therefore, if a static set of parameters are desired, then it is recommended to employ the parameters suggested by Carlisle and Dozier [12]. Conversely, the worst performance was consistently demonstrated by PSO-9 and PSO-4, thus it is recommended to avoid these two parameter configurations when employing PSO.

The next chapter examines the short-term vs. long-term performance of PSO parameter configurations. Predictive models are then trained to predict the (relative) long-term performance of parameter configurations. Additionally, a parameter-free PSO algorithm is proposed based on these predictive models.

Chapter 9

A Parameter-Free Particle Swarm Optimization Algorithm using Performance Classifiers

This chapter presents an investigation into the short-term versus long-term performance of various PSO parameter configurations. While Chapter 7 provided evidence that the best PSO parameters to employ are time-dependent, this investigation provides a more in-depth examination of a small set of parameters, thereby providing a more concrete quantification of the performance degradation observed with specific parameter configurations over time. Given that the short-term performance is not necessarily indicative of long-term performance, this poses a problem for adaptive algorithms, which rely on real-time information to decide upon parameter configurations. To predict long-term performance of PSO parameter configurations, this chapter proposes using machine learning techniques to build predictive models based upon two easily-observable landscape characteristics. Finally, using the predictive models as a basis, this chapter also proposes a parameter-free PSO algorithm.

The remainder of this chapter is structured as follows. Section 9.1 investigates short-term and long-term performance of PSO parameter configurations. Section 9.2 builds classification models for predicting the long-term performance of PSO parameter configurations, while Section 9.3 proposes a parameter-free algorithm based upon these models.

Finally, Section 9.4 provides a summary of this chapter.

9.1 Investigating Short-Term Versus Long-Term Performance

This section presents an investigation of whether short-term PSO performance is indicative of long-term performance. Section 9.1.1 describes the investigation procedure, while Section 9.1.2 presents a discussion of the results.

9.1.1 Experimental Setup

To investigate the short-term versus long-term performance of various PSO configurations, this section uses the experimental data from the 14 parameter configurations identified in Chapter 8 (see Table 8.1). The objective fitness value was taken at iterations 10, 50, 100, 250, 1000, 2500, and 5000. For each benchmark problem and iteration, pairwise two-tailed Mann-Whitney U tests were performed at a significance level of 0.05 to identify performance differences. When the Mann-Whitney U test indicated that a difference existed among two strategies, the median performance measure values were used to assign wins and losses; the better performing strategy was awarded a win, while the inferior strategy was awarded a loss. The strategies were then assigned a rank based on the difference between the number of wins and losses. Additionally, the best rank frequency, defined as the number of benchmark problems for which the strategy attained the highest rank, was recorded.

9.1.2 Results

Table 9.1 presents the average rank across all 60 benchmark problems at various iterations. A few noteworthy observations can be made from this data. Firstly, PSO-4 depicted the best average rank at iterations 10 and 100, but the second worst rank after 5000 iterations. Similarly, PSO-9 demonstrated the best average rank at 50 iterations, but the worst average rank after 5000 iterations. Clearly, short-term performance was not indicative of long-term performance for these parameter configurations. While less pronounced, there are also examples where the long-term performance was better than the short-term performance. As an example, consider PSO-13, which demonstrated an average rank between 11 and 13 for the first 100 iterations, but improved to an average rank of 7.233 after 5000 iterations. Similarly, PSO-12 attained an average rank between 8 and 10 for the first 100 iterations, but attained an average rank of 5.100 after 5000 iterations. Notably, PSO-2 demonstrated consistent good performance throughout the search, and attained the best average rank for all iterations after 250.

Table 9.1: Average rank at various iterations across all problems

	Iteration							
Configuration	10	50	100	250	500	1000	2500	5000
PSO-1	6.183	5.683	5.583	4.850	4.017	4.433	4.633	4.883
PSO-2	1.450	2.817	2.117	**1.833**	**2.050**	**2.850**	**3.683**	**4.400**
PSO-3	7.883	4.783	3.817	2.700	3.083	4.050	5.083	5.700
PSO-4	**1.083**	1.283	**1.133**	2.683	5.800	8.217	9.700	10.217
PSO-5	9.183	10.667	11.083	11.583	11.083	9.233	7.283	6.583
PSO-6	5.233	4.117	3.883	3.233	3.483	4.333	5.617	5.617
PSO-7	4.333	4.817	5.300	5.050	4.350	4.350	4.617	4.767
PSO-8	4.317	7.833	8.617	9.200	9.100	7.250	6.100	5.433
PSO-9	1.700	**1.217**	2.417	6.533	9.783	11.500	12.467	12.733
PSO-10	9.317	11.017	11.350	11.900	11.333	9.383	7.750	6.867
PSO-11	10.783	9.533	8.733	7.367	6.133	5.900	5.383	5.667
PSO-12	9.567	8.800	8.467	7.567	5.833	5.150	4.700	5.100
PSO-13	12.450	12.717	11.467	10.283	8.833	7.683	7.183	7.233
PSO-14	12.683	13.750	13.917	13.883	12.933	10.583	8.783	8.050

As an alternative to the average rank, Table 9.2 presents the best rank frequency across all problems at various iterations. This table more clearly shows the striking difference in short-term and long-term performance. In fact, this table shows that short-term performance was dominated by only three configurations, namely PSO-2, PSO-4, and PSO-9, which were the only configurations to attain the best rank on any benchmark problem at iterations 10, 50, and 100. An example of misleading performance can be found by examining the results for PSO-4, which demonstrated the best rank on 55, 43, and 52 problems at iterations 10, 50, and 100, respectively. However, after 5000 iterations, PSO-4 only attained the best rank on one problem. A similar observation can be made when examining the results for PSO-9. Conversely, there are numerous examples where short-term performance (i.e., less than 1000 iterations) was relatively poor, but long-term performance was relatively good (e.g., PSO-3, PSO-7, and PSO-8).

Table 9.2: Best rank frequency at various iterations across all problems

	Iteration							
PSO	10	50	100	250	500	1000	2500	5000
PSO-1	0	0	0	0	5	7	7	9
PSO-2	41	4	22	**41**	**32**	**32**	**23**	**23**
PSO-3	0	0	0	4	13	16	15	16
PSO-4	**55**	43	**52**	26	12	4	2	1
PSO-5	0	0	0	0	0	3	6	7
PSO-6	0	0	0	1	9	10	6	9
PSO-7	0	0	0	0	4	9	8	11
PSO-8	0	0	0	0	0	4	6	12
PSO-9	35	**47**	22	6	3	1	1	1
PSO-10	0	0	0	0	0	3	6	8
PSO-11	0	0	0	0	7	8	9	8
PSO-12	0	0	0	0	7	7	8	5
PSO-13	0	0	0	0	1	7	6	4
PSO-14	0	0	0	0	0	5	9	10

Figure 9.1 presents the average fitness over the first 1000 iterations on selected benchmark problems for PSO-2, PSO-4, PSO-9, PSO-11, and PSO-14. Note that f_1 and f_{22} are unimodal problems, f_4, f_6, and f_{13} are multimodal problems, and f_{37} is a composition problem. Note that PSO-9, which had the best average rank after 50 iterations, but the worst average rank after 5000 iterations, depicted very rapid fitness stagnation and generally did not improve after 200 iterations. Conversely, PSO-14, which attained the third from worst rank after 5000 iterations, depicted the slowest fitness stagnation of

the five examined configurations. This indicates that both premature convergence and delayed convergence can pose issues for performance.

In conjunction with the results from Chapter 7, this section provided strong evidence against the real-time adaptation of control parameter values based solely upon their short-term performance. Specifically, the results clearly indicated that the best parameter configurations identified early in the optimization process can lead to the worst long-term performance. Therefore, an adaptive algorithm that naively selects parameter configurations based on their early performance is likely to lead to both sub-optimal performance and early convergence. To mitigate this issue, the next section builds predictive models that can accurately predict the long-term performance of PSO parameter configurations based upon two easily-observable landscape characteristics.

9.2 Predicting the Long-Term Performance of Particle Swarm Optimization Parameter Configurations

This section describes the training process used to build predictive models for classifying the long-term performance of PSO parameter configurations. Section 9.2.1 describes the training data and methodology, while Section 9.2.2 describes the algorithms used to build the predictive models. Finally, Section 9.2.3 discusses the accuracy of the trained classifiers.

9.2.1 Training Data

To account for various landscape types, two simple fitness landscape metrics were used to discriminate the environment type. The fitness-distance correlation (FDC) [77] measures the correlation between the fitness of a solution and its distance to the nearest optimum[1]

[1]If the optimum is not known, the solution with the best empirical fitness is used.

and is given by

$$FDC = \frac{\sum_{i=1}^{n}(f(\mathbf{x}_i) - \overline{f(\mathbf{x}_i)})(d_i^* - \overline{d^*})}{\sqrt{\sum_{i=1}^{n}(f(\mathbf{x}_i) - \overline{f(\mathbf{x}_i)})^2}\sqrt{\sum_{i=1}^{n}(d_i^* - \overline{d^*})^2}}, \tag{9.1}$$

where d_i^* is the distance between the position of sample i and the nearest optimum. The dispersion metric (DM) [76] quantifies the dispersion (i.e., spread) of a subset of the best-fit solutions against the dispersion of the remainder of solutions in the sample and is given by

$$DM = \text{disp}(X^*) - \text{disp}(X), \tag{9.2}$$

where X is the set of samples, $X^* \subset X$ is a subset of the best fit samples from X, and disp(X) is a measure of dispersion, which is taken as the average pairwise Euclidean distance between all solutions in the sample. Together, these measures give a quantification of the modality (via the DM) and deceptiveness (via the FDC) of the search space. While there are certainly more landscape measurements that could be used to quantify the search space [75], these two measures were selected as they require only a uniform random sample, and therefore can be calculated easily in real-time.

Each training instance was formulated as a six-tuple of the form

$$(FDC, DM, \omega, c_1, c_2, class),$$

where *class* was one of the following labels based on the rank, R, of a particular parameter configuration on a specific benchmark problem using the empirical data from Chapter 7:

$$\text{class} = \begin{cases} \text{Excellent (E)} & R < 31 \\ \text{Very Good (VG)} & 31 \leq R < 152 \\ \text{Good (G)} & 152 \leq R < 304 \\ \text{Average (A)} & 304 \leq R < 759 \\ \text{Poor (P)} & 759 \leq R < 1518 \\ \text{Very Poor (VP)} & 1518 \leq R < 2277 \\ \text{Terrible (T)} & 2277 \leq R \end{cases} \tag{9.3}$$

Note that these classifications correspond to the 1st, 5th, 10th, 25th, 50th, 75th, and 100th percentiles. Both the FDC and DM values were computed using the average of 100 trials, each consisting of 1000 uniformly selected points. The training data thus consists of 66792 instances, encompassing 3036 parameter configurations across 22 benchmark problems, based upon the results from Chapter 7. Each classifier was trained by way of stratified 10-fold cross validation using Weka 3.8.2 [40].

9.2.2 Classifiers

This section describes the five classifiers that were constructed using the training data described in Section 9.2.1.

C4.5 (J48) Decision Tree

A decision tree is a machine learning tool for predictive modelling, most commonly employed for classification tasks. The C4.5 algorithm [92], also known as J48, is a decision tree algorithm built upon the earlier iterative dichotomiser 3 (ID3) algorithm [91]. To build a classifier from a collection of training instances, each consisting of a set of attributes and a (known) classification, the C4.5 algorithm recursively selects an attribute a_i that most effectively splits the set of training examples into subsets by maximizing the information gain. Information gain, denoted $IG(a_i, S)$, is a measure of the difference in entropy between a set S before and after it is split on attribute a_i, given by

$$IG(a_i, S) = H(S) - \sum_{t \in T} p(t)H(t) \tag{9.4a}$$

and

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c), \tag{9.4b}$$

where T is the set of subsets created by splitting S on a_i, $t \in T$ is a subset of S such that $S = \bigcup_{t \in T} t$, $p(t)$ is the proportion of elements in class t, and C is the set of classifications present in S. Equation (9.4b) is referred as the entropy of the set S. The attribute a_i that maximizes Equation (9.4a) is then used to make a decision node, which splits the instances into subsets. The algorithm then recursively builds subtrees from each of the subsets.

Random Forest

Random forests are an ensemble technique that combines a collection of tree predictors, such that each tree is built using an independent random sample from the training set [11]. Furthermore, the individual trees are built using random subsets of the training attributes, thereby increasing the diversity in the resulting trees. The justification for the random attribute selection is that it reduces the generalization error without the risk of over-fitting [11]. To form the random sample of instances, instances are uniformly randomly selected, with replacement, from the training set S. To output a final classification, each classifier casts a unit vote, with the most popular class being the final classification.

Class Balancing

Given that the number of examples for the classes identified in Section 9.2.1 were unequal, a second model was built using a class balancing technique for both the J48 and random forest classifiers. Class balancing is an instance weighting scheme that reconfigures the weight associated with each training instance such that the sum of weights assigned to each classification label is equalized while maintaining the same total weight.

Meta-model

In a similar fashion to the random forest algorithm, the meta model is an ensemble technique that uses multiple underlying classifiers. The final classification is taken as the most popular class, as voted by the underlying classifiers. For the purposes of this study, the meta-model is constructed using each of the following four classifiers:

- J48
- J48 (balanced)
- Random forest

- Random forest (balanced)

9.2.3 Classifier Accuracy

This section discusses the predictive accuracy of each classifier. Table 9.3 provides the overall accuracy, defined as the percentage of correctly classified instances, while the confusion matrices for each classifier are provided in Tables 9.4 to 9.8. Bold entries along the diagonal indicate correct predictions while non-diagonal entries indicate incorrect predictions. Additionally, the *precision*, defined as the ratio of true positives to correctly classified instances, and *recall*, defined as the ratio of true positives to the sum of true positives and false negatives, are provided for each of the classifications.

It is clear from Table 9.3 that the random forest models lead to better accuracy than the corresponding J48 models. Examination of the confusion matrices provided in Tables 9.4 to 9.7 shows that the random forest models provided better accuracy than the J48 models for all classes. An additional noteworthy observation was that class balancing lead to decreased overall accuracy. However, from the confusion matrices, it is evident that the accuracy for the 'Excellent' class, which has the lowest number of instances, was increased at the expense of decreased accuracy on other instances. Therefore, it can be concluded that the class balancing was working as intended; class balancing increased the accuracy of the under-represented 'Excellent' class at the expense of decreased accuracy on the other, over-represented classes. The meta model provided a significantly higher accuracy than each of the constituent models, and the precision and recall measures for the meta-model was above 0.95 for all but the 'Very Poor' class. This result is unsurprising as it can leverage the relative strengths of each of the underlying models. Notably, the meta-model can make use of the bias towards the 'Excellent' category from the balanced models, while maintaining the overall accuracy from the non-balanced models.

Table 9.3: Overall accuracy by classifier

Model	Accuracy
J48	84.36%
J48 (balanced)	73.64%
Random Forest	86.64%
Random Forest (balanced)	77.03%
Meta	95.97%

9.3 A Parameter-Free Particle Swarm Optimization using Performance Classifiers

This section proposes a parameter-free PSO algorithm, entitled classifier model particle swarm optimization (CMPSO), using the classifiers built in Section 9.2. As a first consideration, the classifiers were built using two measures of the fitness landscape that need to be supplied. Note that, though the FDC and DM values could be easily computed and supplied to the CMPSO algorithm, it would be preferable to avoid such a scenario as this would constitute additional function evaluations. To address this issue, the proposed algorithm first foregoes a number of PSO iterations to perform a uniform random sampling that is used to compute the FDC and DM values. This ensures that the total number of function evaluations remained unaltered. After the initial sampling period, the trained classifier was used to generate, independently for each particle, random values for the ω, c_1, and c_2 control parameters until one such configuration was classified as 'Excellent' by the employed classifier. Furthermore, the generated parameter configurations were guaranteed to satisfy the convergence criterion given by Equation (2.5). After the parameter selection process, the CMPSO algorithm continued execution in the same fashion as the standard PSO algorithm described in Chapter 2.

9.3.1 Experimental Setup

All examined PSO variants consisted of 30 particles arranged in a star neighbourhood and used a synchronous update strategy. The uniform sampling procedure for the FDC

Table 9.4: Confusion matrix, J48

Predicted\Actual	E	VG	G	A	P	VP	T	Recall
E	**219**	386	88	11	1	2	0	0.310
VG	186	**1780**	555	79	24	1	0	0.678
G	35	678	**1920**	643	51	14	0	0.575
A	6	73	555	**8057**	1279	47	0	0.804
P	5	19	51	1338	**14245**	1031	1	0.854
VP	3	1	7	57	1160	**14353**	1130	0.859
T	0	0	0	0	0	932	**15769**	0.944
Precision	0.482	0.606	0.605	0.791	0.850	0.876	0.933	

and DM values consisted of 33 iterations, leading to a total of 990 (i.e., $\approx$ 1000) samples being used to compute the values for the landscape metrics. Particle velocities were initially set to the zero vector [30]. To prevent infeasible attractors, a particle's personal best position was only updated if a new position had a better objective function value and was within the feasible bounds of the search space. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. The objective fitness value averaged over 50 independent runs, each consisting of 5000 iterations, was taken as the measure of performance for each algorithm.

Statistical analysis of results was done by way of Friedman's test for multiple comparisons among all methods. Furthermore, Shaffer's post-hoc procedure was performed as a means to identify the pairwise comparisons which produced significant differences. Finally, critical difference plots were generated, whereby algorithms which are to the left of the plot demonstrated superior performance, and algorithms that are grouped by a line were found to have insignificantly different performance.

9.3.2 Results

First, a comparison of the various CMPSO variants is discussed. Figure 9.2 depicts the critical difference plot across all 60 benchmark problems, while Figures 9.3 and 9.4 present the critical difference plots on the unimodal and multimodal problems, respectively. Figure 9.2 shows that the best overall rank was attained by CMPSO employing

Table 9.5: Confusion matrix, J48 (balanced)

Predicted\Actual	E	VG	G	A	P	VP	T	Recall
E	**4764.11**	3859.87	823.26	40.49	13.5	26.99	13.5	0.499
VG	1290.4	**5906.78**	2231.85	87.24	18.17	3.63	3.63	0.619
G	317.01	1864.93	**6285.94**	965.31	68.54	39.98	0.00	0.659
A	28.58	138.12	935.41	**7454.67**	939.22	45.72	0.00	0.781
P	13.15	28.01	102.33	1131.4	**7675.68**	589.43	1.72	0.804
VP	10.28	10.85	33.69	67.38	693.17	**8059.44**	666.91	0.845
T	0.57	1.14	0.57	1.71	0.00	497.05	**9040.66**	0.929
Precision	0.742	0.500	0.604	0.765	0.816	0.870	0.929	

the classifier built using J48. However, it is noted that the only strategy that performed significantly worse than the rest, was CMPSO-rf-balanced. It is interesting to note that despite the higher accuracy of the random forest and meta models, the J48 classifier lead to the best performance in the CMPSO algorithm. Therefore, it can be concluded that there is little correlation between the accuracy of the underlying model and the performance of the CMPSO algorithm employing the model.

Comparison with Other PSO Variants

This section compares the performance of CMPSO-j48 against five other PSO variants, namely PSO-1, PSO-2, PSO-7, PSO-TVAC, and PSO-iRC-p5, which represent the three best performing static parameter configurations, a SAPSO algorithm adapting all three control parameters, and the best parametrization of PSO-iRC from Chapter 8, respectively. Figure 9.5 depicts the critical difference plot comparing the performance across all 60 benchmark problems, while Figures 9.6 and 9.7 present the critical difference plots on the unimodal and multimodal problems, respectively.

From Figure 9.5, it can be seen that, while CMPSO-j48 attained the lowest average rank across all problems, there was an insignificant difference in performance among all strategies. Furthermore, the difference in average rank was less than one among all examined strategies. Considering performance on unimodal problems, as shown in Figure 9.6, the CMPSO-j48 algorithm, along with PSO-TVAC, demonstrated the worst overall rank. However, when considering that the training data consisted of 41 multimodal

Table 9.6: Confusion matrix, Random Forest

Predicted\Actual	E	VG	G	A	P	VP	T	Recall
E	**287**	362	51	6	1	0	0	0.406
VG	190	**1860**	523	36	16	0	0	0.709
G	36	520	**2205**	525	48	7	0	0.660
A	3	44	450	**8459**	1034	27	0	0.844
P	3	13	58	1017	**14711**	888	0	0.881
VP	1	1	11	36	975	**14645**	1042	0.876
T	0	0	0	0	2	996	**15703**	0.940
Precision	0.552	0.664	0.669	0.839	0.876	0.884	0.938	

problems and only 19 unimodal problems, it is hypothesized that the classifier developed a bias towards parameter configurations that performed well on multimodal problems. This hypothesis is then reinforced by Figure 9.7, which shows that the CMPSO-j48 fared better on multimodal problems than unimodal problems. In fact, the performance of CMPSO-j48 was insignificantly different than that of PSO-2, PSO-TVAC, and PSO-7, which all attained the best performance on the multimodal problems.

9.4 Summary

This chapter investigated the short-term and long-term performance of various PSO parameter configurations. In support of the findings in Chapter 7, this investigation provided further evidence to support that the best parameter configurations are, in fact, time-dependent. It was found that both premature and delayed convergence lead to relatively poor performance. Furthermore, it was concluded that an adaptive algorithm that naively selects parameter configurations based upon their short-term performance may lead to sub-optimal performance. To mitigate this issue, this chapter developed predictive models, using machine learning techniques, to predict the long-term performance of various PSO parameter configurations. The classification accuracy of the models was between 73.64% and 95.97%. Using these models, a parameter-free PSO algorithm was proposed. The results indicated that the proposed CMPSO algorithm attained perfor-

Table 9.7: Confusion matrix, Random Forest (balanced)

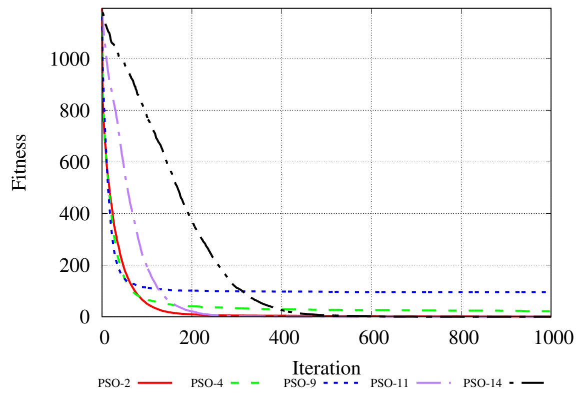
Predicted\Actual	E	VG	G	A	P	VP	T	Recall
E	**4602.16**	4251.26	634.31	40.49	0.00	13.50	0.00	0.482
VG	861.48	**6648.30**	1926.52	90.87	10.90	3.63	0.00	0.697
G	125.66	1650.74	**6571.53**	1133.81	42.84	17.14	0.00	0.689
A	11.43	76.20	652.50	**8079.55**	704.89	17.15	0.00	0.847
P	8.00	21.72	69.75	772.94	**8205.65**	463.08	0.57	0.860
VP	6.85	11.42	23.98	26.27	540.15	**8348.93**	584.12	0.875
T	0.00	0.00	0.00	0.00	0.00	547.33	**8994.38**	0.943
Precision	0.820	0.525	0.665	0.796	0.863	0.887	0.939	

mance that was insignificantly different from other top-performing PSO variants, but eliminated the need to specify values for ω, c_1, and c_2. However, poor performance on unimodal problems was noted.

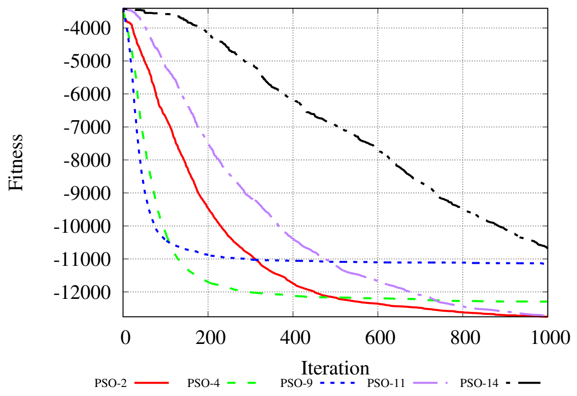
In the next chapter, a PSO algorithm that forgoes the conventional control parameters in favour of a probabilistic position update procedure is proposed.

Table 9.8: Confusion matrix, meta-model

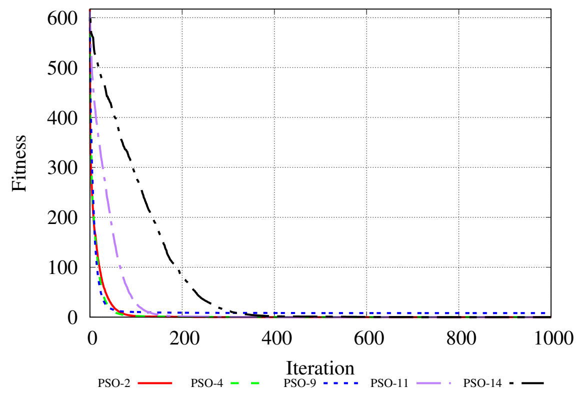
Predicted\Actual	E	VG	G	A	P	VP	T	Recall
E	**707**	0	0	0	0	0	0	1.000
VG	8	**2610**	7	0	0	0	0	0.994
G	1	26	**3298**	16	0	0	0	0.987
A	2	13	144	**9655**	199	4	0	0.964
P	0	5	19	449	**15893**	324	0	0.952
VP	0	0	1	9	417	**15706**	578	0.940
T	0	0	0	0	0	471	**16230**	0.972
Precision	0.985	0.983	0.951	0.953	0.963	0.952	0.966	



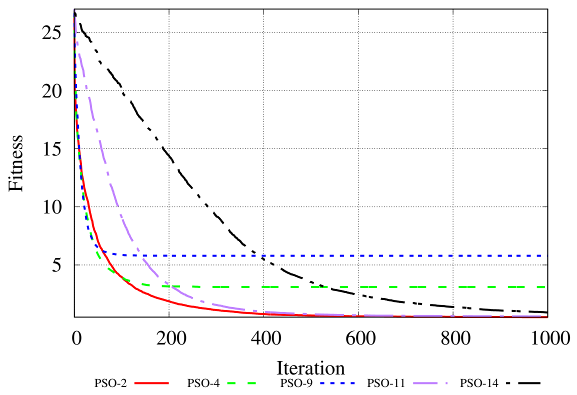
(a) Average fitness over 1000 iterations on f_1



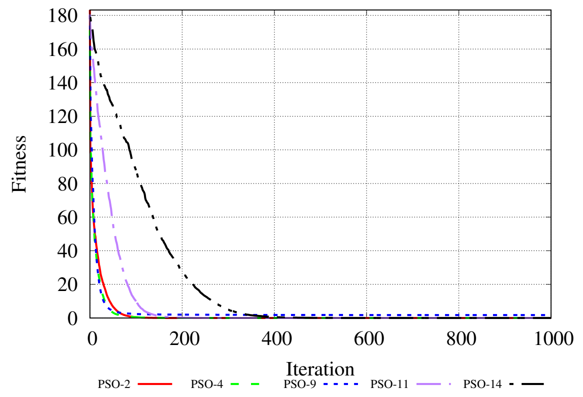
(b) Average fitness over 1000 iterations on f_4



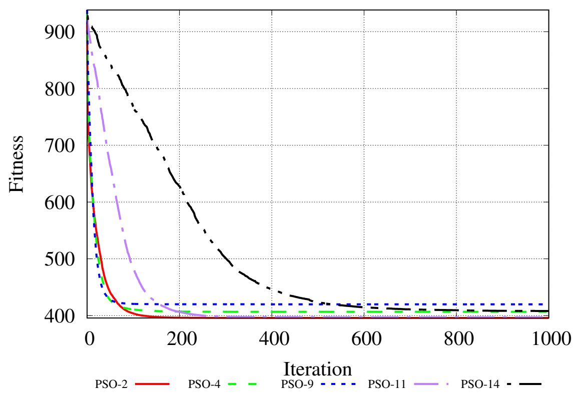
(c) Average fitness over 1000 iterations on f_6



(d) Average fitness over 1000 iterations on f_{14}



(e) Average fitness over 1000 iterations on f_{22}



(f) Average fitness over 1000 iterations on f_{37}

Figure 9.1: Average fitness of selected PSO configurations for the first 1000 iterations on various benchmark problems.

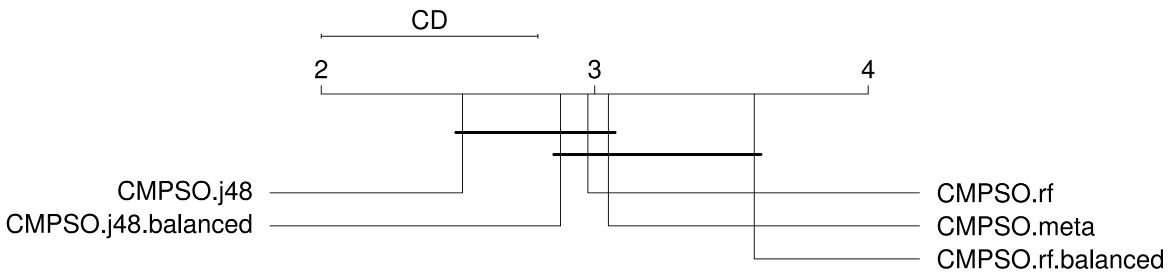


Figure 9.2: Comparison of CMPSO variants across all 60 problems.

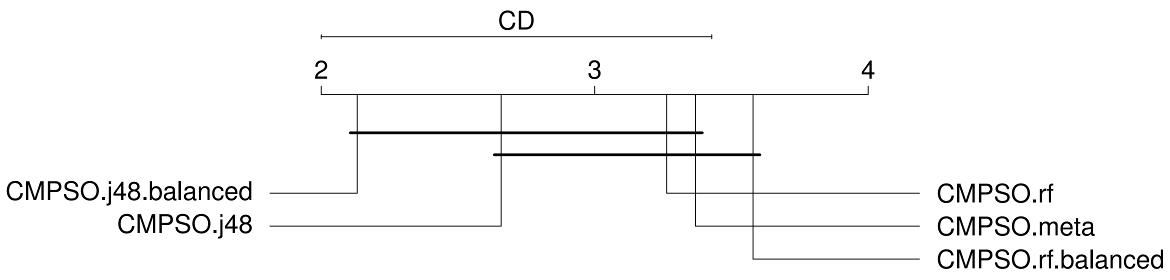


Figure 9.3: Comparison of CMPSO variants on the 19 unimodal problems.

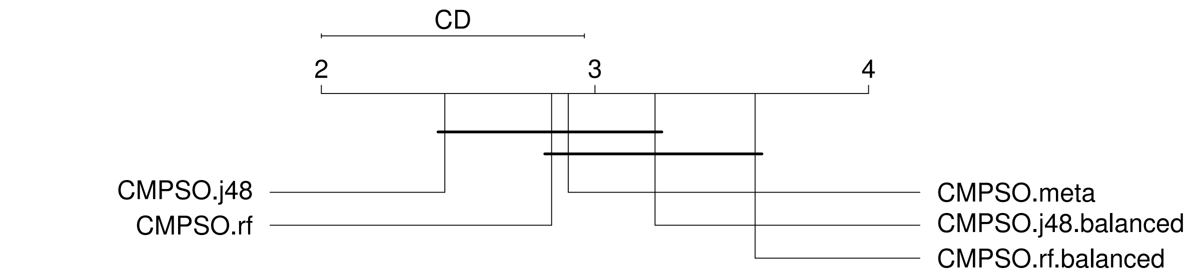


Figure 9.4: Comparison of CMPSO variants on the 41 multimodal problems.

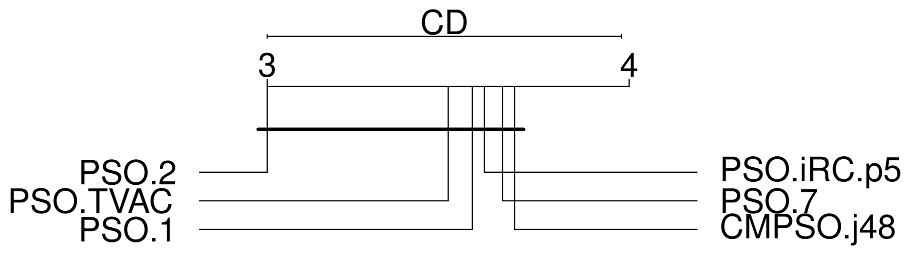


Figure 9.5: Comparison of CMPSO-j48 with other PSO variants across all 60 problems.

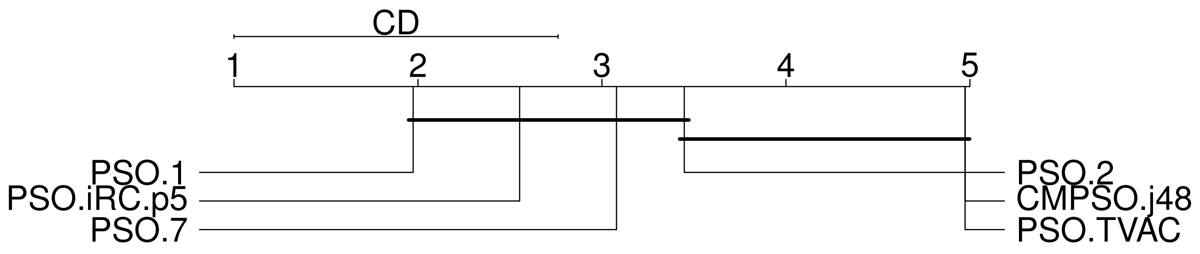


Figure 9.6: Comparison of CMPSO-j48 with other PSO variants on the 19 unimodal problems.

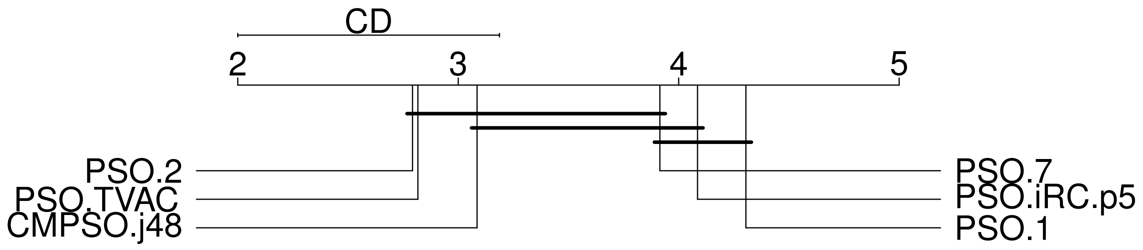


Figure 9.7: Comparison of CMPSO-j48 with other PSO variants on the 41 multimodal problems.

Chapter 10

Gaussian-Valued Particle Swarm Optimization

An alternative approach to self-adaptation of control parameter values is to use PSO variants that do not rely on such control parameter values. An example of a PSO technique that does not rely on the conventional control parameters is the bare bones particle swarm optimization (BBPSO) [58], which updates particle positions probabilistically using a Gaussian distribution. However, the manner in which particle positions are created via BBPSO is strikingly dissimilar to how the standard PSO determines updated particle positions.

In this chapter, a new PSO variant is proposed by formulating a new probabilistic approach to generating particle positions. The new approach is inspired by the BBPSO algorithm, but differs significantly in the manner by which particle positions are generated. Notably, the proposed algorithm generates particle positions using a model that more closely resembles the standard PSO, which as this chapter will demonstrate, provides a clear performance advantage over BBPSO and other PSO configurations.

The remainder of this chapter is structured as follows. Section 10.1 provides background information about the BBPSO algorithm. The proposed algorithm is then described in Section 10.2, while Section 10.3 details the empirical analysis and results. Finally, a summary of the findings in this chapter are provided in Section 10.4.

10.1 Barebones Particle Swarm Optimization

When examining the extent to which particles examine their surroundings, Kennedy [58] noted that particle positions formed a bell curve centred around the midpoint between the global and personal best positions. Based on this result, the BBPSO algorithm [58] eliminates the velocity component of PSO and rather updates particle positions probabilistically according to

$$x_{ij}(t+1) = \mathcal{N}\left(\frac{c_1 y_{ij}(t) + c_2\hat{y}_{ij}(t)}{c_1 + c_2}, |y_{ij}(t) - \hat{y}_{ij}(t)|\right), \tag{10.1}$$

where $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. In the original formulation, the control parameters were set as $c_1 = c_2 = 1$ [58]. Later theoretical results supported the observation of Kennedy by showing that, using the stagnation and deterministic assumptions, each particle will converge toward the weighted average of the personal and neighbourhood bests [105, 107], as given by

$$\frac{c_1\mathbf{y}_i + c_2\hat{\mathbf{y}}_i}{c_1 + c_2}.$$

To further improve the exploration capabilities, the BBPSO update equation was amended to include a per-dimension chance of selecting the personal best position, as given by

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < e \\ \mathcal{N}\left(\frac{c_1 y_{ij}(t) + c_2\hat{y}_{ij}(t)}{c_1 + c_2}, |y_{ij}(t) - \hat{y}_{ij}(t)|\right) & \text{otherwise,} \end{cases} \tag{10.2}$$

where e is a user-supplied parameter controlling the degree to which the personal best location is exploited.

10.2 Gaussian Valued Particle Swarm Optimization

To provide the motivation for the proposed algorithm, consider the PSO velocity equation given in Equation (2.1) when $\omega = 0$. With no inertia, the velocity calculation simplifies to

$$v_{ij}(t+1) = c_1 r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2ij}(t)(\hat{y}_{ij}(t) - x_{ij}(t)). \tag{10.3}$$

Note that, because $r_{1ij}(t), r_{2ij}(t) \sim U(0,1)$, Equation (10.3) can be stated alternatively as

$$v_{ij}(t+1) = v_{1ij}(t) + v_{2ij}(t), \tag{10.4}$$

where $v_{1ij}(t) \sim U(0, c_1(y_{ij}(t) - x_{ij}(t)))$ and $v_{2ij}(t) \sim U(0, c_2(\hat{y}_{ij}(t) - x_{ij}(t)))$[1]. It can be easily observed that the position update becomes a sum of two uniform distributions, thereby leading to a trapezoidal distribution [73]. The shape of the resulting trapezoidal distribution is then governed by the distance between the current particle's position and the personal and neighbourhood best positions, respectively. Even with the reintroduction of the inertia component, the same general observation can be made: the particle position update depends heavily upon not only the personal and neighbourhood best positions, but also the distance between the current particle and these two attractors.

The Gaussian-valued particle swarm optimization (GVPSO) algorithm is proposed by employing the observations noted above to probabilistically determine particle positions. The position update mechanism for GVPSO is formulated by employing a Gaussian distribution centred at a random point taken from the aforementioned trapezoidal distribution. The Gaussian distribution is used to modulate the particle step sizes based upon the distance between the current position and the personal and neighbourhood best positions. Specifically, an ancillary position, $\Delta_{ij}(t)$, is calculated for each particle in every dimension using Equations (2.1) and (2.2) with $\omega = 0$ and $c_1 = c_2 = 1$. This effectively retains the core movement pattern of PSO, without the reliance on control parameter values. The particle's new position is then determined using a Gaussian distribution centred between the current position and $\Delta_{ij}(t)$ with a standard deviation based on the magnitude of the distance between the current position and $\Delta_{ij}(t)$ according to

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < e \\ \mathcal{N}\left(\frac{x_{ij}(t)+\Delta_{ij}(t)}{2}, |\Delta_{ij}(t) - x_{ij}(t)|\right) & \text{otherwise,} \end{cases} \tag{10.5a}$$

where

$$\Delta_{ij}(t) = x_{ij}(t) + r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + r_{2ij}(t)(\hat{y}_{ij}(t) - x_{ij}(t)), \tag{10.5b}$$

[1]Without loss of generality, this assumes that $c_1(y_{ij}(t) - x_{ij}(t)) > 0$ and $c_2(\hat{y}_{ij}(t) - x_{ij}(t)) > 0$, otherwise the bounds must be flipped, i.e., 0 becomes the upper bound.

and e is an exploitation parameter, as seen in Equation (10.2). Note that GVPSO, in the same manner as BBPSO, eliminates the need for the conventional PSO parameters ω, c_1, and c_2, but that there remains one control parameter, e. However, the GVPSO algorithm differs from BBPSO by creating particle positions that more closely mimic the position update of the standard PSO through the use of distance information, and thus the two attractors remain to have a strong influence. Furthermore, the step sizes in the GVPSO algorithm are implicitly controlled by the distances between the current particle and the two attractors, thereby leading to diminishing step sizes as the positions and attractors inevitably converge. Thus, the GVPSO is expected to exhibit both initial exploration and exploitation in the later phase of the search process.

10.3 Experimental Results and Discussion

This section presents the experimental design regarding the empirical examination of GVPSO. Section 10.3.1 describes the parametrization and statistical analysis. Section 10.3.2 performs a sensitivity analysis on the value of the exploitation parameter, while Section 10.3.3 presents a comparison of GVPSO to other PSO variants.

10.3.1 Experimental Setup

To first examine the effect of the exploitation probability parameter e, 10 values of e were examined for GVPSO and BBPSO, namely values between 0.0 and 0.9 in increments of 0.1. Linearly decreasing variants (GVPSO-LD and BBPSO-LD), whereby the value of e was linearly decreased from 0.9 to 0.0 to provide a waning focus on the personal best positions, were also examined. The performance of GVPSO was then compared against the following PSO strategies:

- BBPSO
- PSO-1, PSO-2, and PSO-7, which represent the three best static parameter configurations found in Chapter 8
- PSO-TVAC, which represents a SAPSO algorithm that adapts the value of all three conventional PSO control parameters

- PSO-iRC-p5, which is the best performing variant of PSO-iRC from Chapter 8

All examined variants consisted of 30 particles arranged in a star neighbourhood and used a synchronous update strategy. To prevent infeasible attractors, a particle's personal best position was only updated if a new position had a better objective function value and was within the feasible bounds of the search space. No boundary constraints were enforced, except the restriction of personal best positions to the feasible search space. For the BBPSO algorithm, the original parametrization of $c_1 = c_2 = 1$ was used. Where applicable, particle velocities were initialized to the zero vector [30]. For PSO-TVAC, the social acceleration coefficient was linearly increased from 0.5 to 2.5, while the values of the cognitive and inertia control parameters were linearly decreased from 2.5 to 0.5 and 0.9 to 0.4, respectively. For PSO-iRC, parameter configurations were re-sampled every 5 iterations (i.e., according to PSO-iRC-p5). The value of the objective function (i.e., the fitness), averaged over 50 independent runs each consisting of 5000 iterations, was taken as the measure of performance for each algorithm. Additionally, the diversity, taken as the average distance from the swarm centre [83], and the average particle step size, given by Equation (4.2), were also measured.

Statistical analysis of results was done by way of Friedman's test for multiple comparisons among all methods. Furthermore, Shaffer's post-hoc procedure was performed as a means to identify the pairwise comparisons that produced significant differences. Finally, critical difference plots were generated, whereby algorithms which are to the left of the plot (i.e., those with lower average ranks) demonstrated superior performance, and algorithms which are grouped by a line were found to have insignificantly different performance.

10.3.2 Analysis of the Exploitation Probability

Figures 10.1 to 10.3 show the critical difference plots for the GVPSO algorithm over the entire set of problems, the unimodal problems, and the multimodal problems, respectively. Figures 10.4 to 10.6 show the same results for the BBPSO algorithm. Despite the exact values that lead to the best performance being different among the two algorithms, the general observations were the same. When considering unimodal problems,

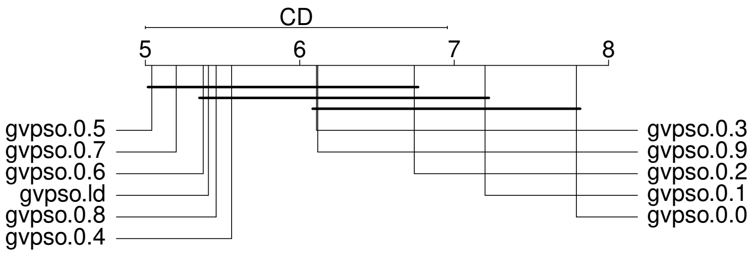


Figure 10.1: Comparison of GVPSO exploit probabilities over all 60 benchmark problems.

smaller values of e (i.e., 0.2–0.5) tended to perform better, while larger values (i.e., 0.6–0.8) tended to perform better for multimodal problems. While intuition may lead to the expectation that enhanced exploitation, i.e., a larger value of e, may be preferred for unimodal problems, this observation can be explained by considering the position update given by Equation (10.2). When both attractors are giving consistent direction information (e.g., in a unimodal landscape), the step size should, in theory, be larger via the probabilistic update as opposed to the exploitative update mechanism. Thus, larger step sizes will lead to more rapid movement toward the attractors, resulting in overall better performance for low values of e. When considering overall performance, mid-range values of e (i.e., 0.4–0.7) tended to perform the best, showing that both exploration and exploitation were beneficial to the GVPSO algorithm. Based upon these results, GVPSO and BBPSO with values of e set to 0.5, 0.6, and 0.7 were subsequently compared against the other PSO techniques.

Figures 10.7 to 10.10 present the average diversity and movement values for select configurations of GVPSO and BBPSO on four benchmark problems, namely f_1, f_4, f_{13}, and f_{27}. Note that f_1 is a unimodal problem, f_4 and f_{13} are multi-modal problems, and f_{27} is a composition problem. Configurations without the exploitation parameter, i.e., GVPSO-0.0 and BBPSO-0.0, and configurations with the best overall performing values of e, namely GVPSO-0.5 and BBPSO-0.6, are presented. First, it is noteworthy that both the diversity and step sizes of the GVPSO algorithm were decreasing over time on all the problems shown in Figures 10.7 to 10.10, indicating stability was exhibited. However, it is noted that the diversity and step sizes were more erratic in the GVPSO algorithm than in the BBPSO algorithm. This is likely a result of the rapidly changing attractors, thereby causing the step sizes to be somewhat erratic in the GVPSO algorithm.

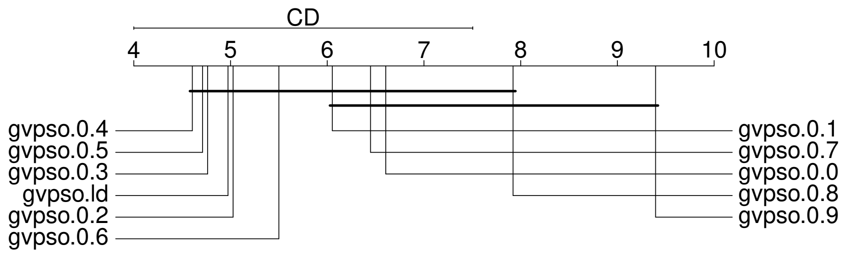


Figure 10.2: Comparison of GVPSO exploit probabilities over the 19 unimodal benchmark problems.

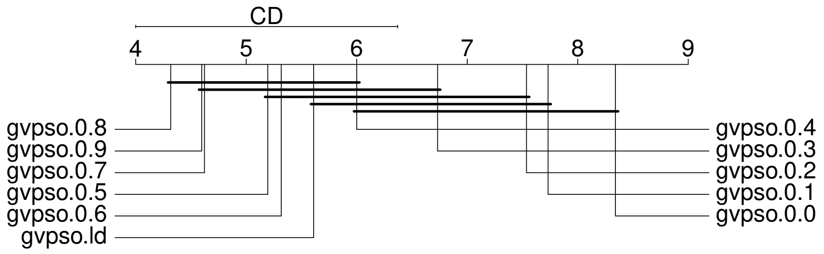


Figure 10.3: Comparison of GVPSO exploit probabilities over the 41 multimodal benchmark problems.

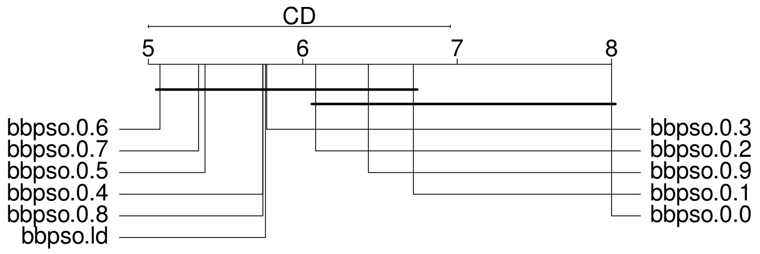


Figure 10.4: Comparison of BBPSO exploit probabilities over all 60 benchmark problems.

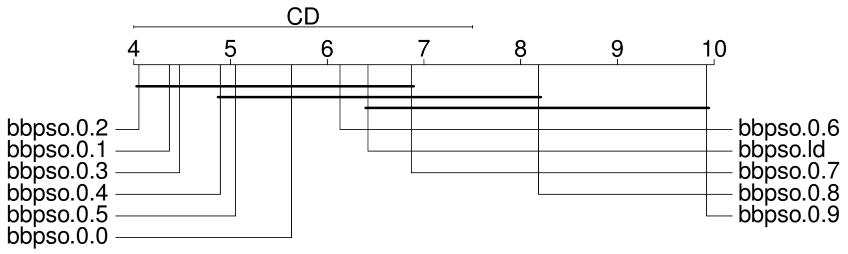


Figure 10.5: Comparison of BBPSO exploit probabilities over the 19 unimodal benchmark problems.

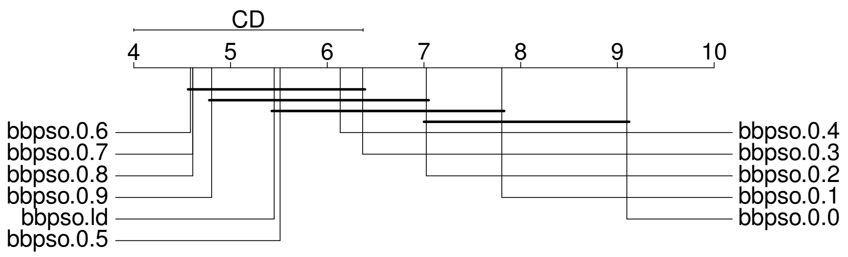
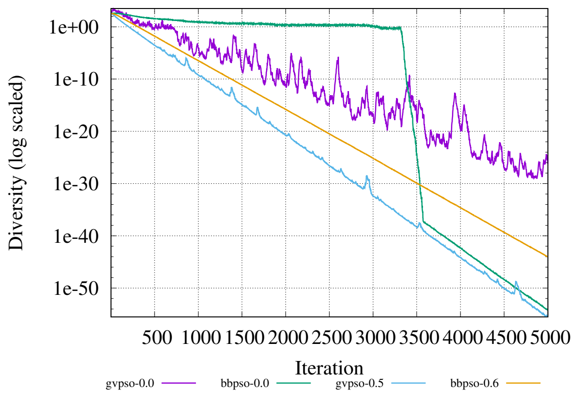
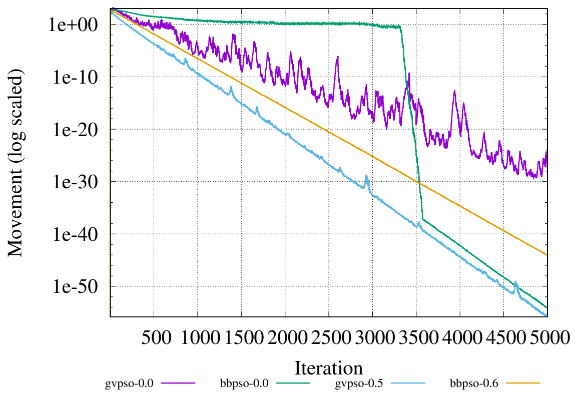


Figure 10.6: Comparison of BBPSO exploit probabilities over the 41 multimodal benchmark problems.

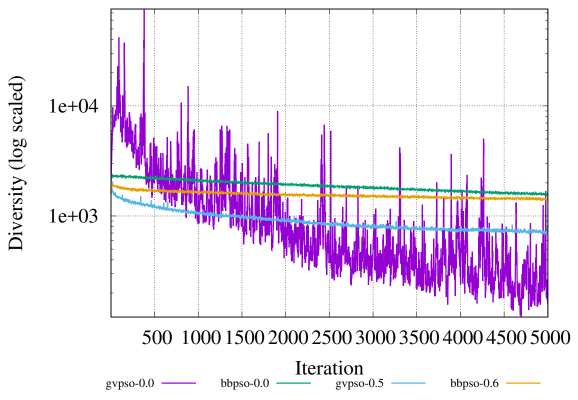


(a) Diversity, f_1

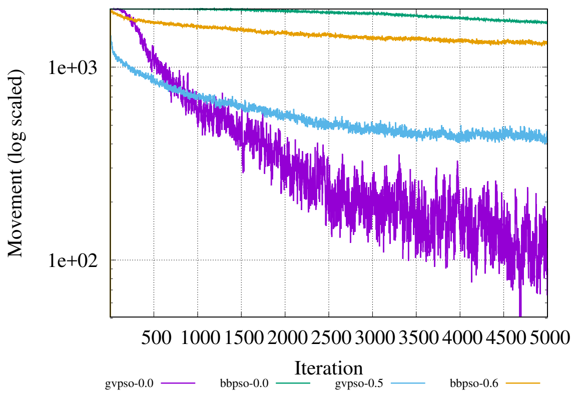


(b) Movement, f_1

Figure 10.7: Diversity and movement profiles for f_1.

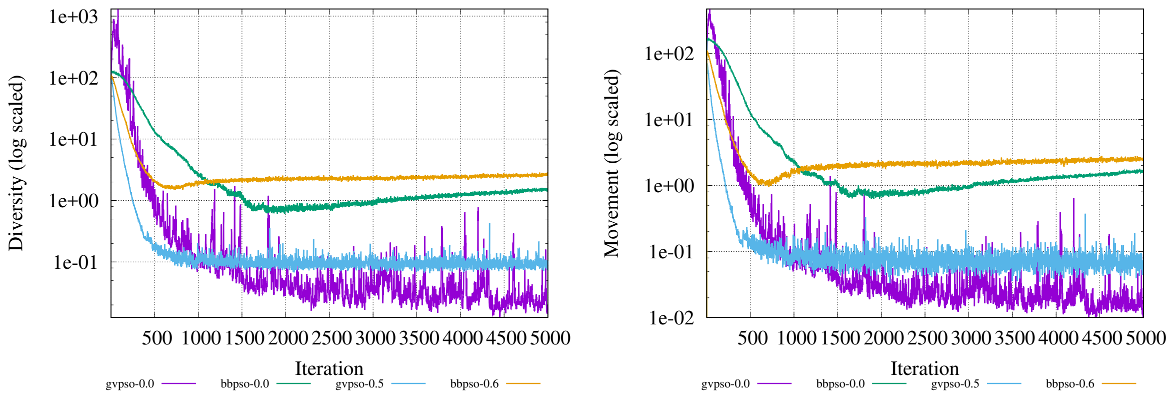


(a) Diversity, f_4



(b) Movement, f_4

Figure 10.8: Diversity and movement profiles for f_4.



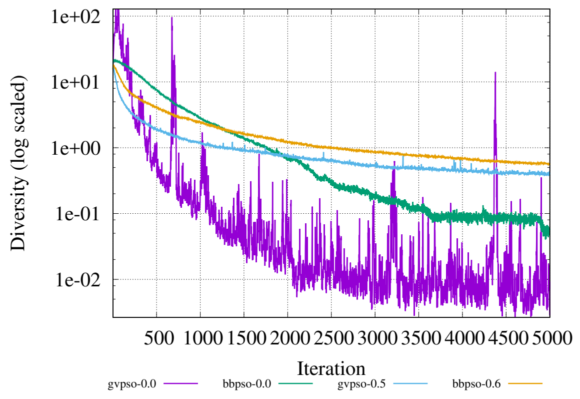
(a) Diversity, f_{13} (b) Movement, f_{13}

Figure 10.9: Diversity and movement profiles for f_{13}.

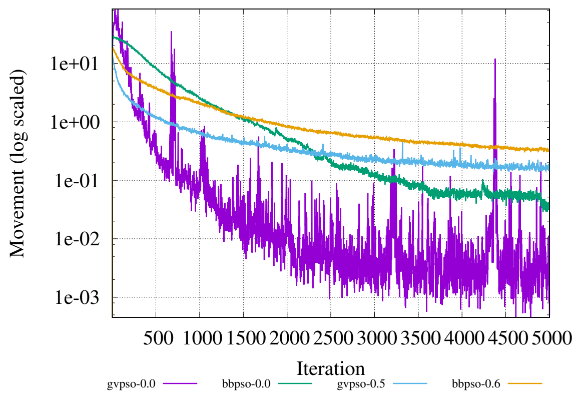
10.3.3 Comparison with Other Particle Swarm Optimization Techniques

This section presents the results from comparing GVPSO (with $e = \{0.5, 0.6, 0.7\}$) against the other PSO variants. Figure 10.11 shows the results across all benchmark problems. It is first observed that the best average ranks across all benchmark problems were attained by the three configurations of GVPSO, clearly indicating the merit of this approach. Despite the better average rank attained by GVPSO, the critical difference plot indicates that there was no significant difference in performance between GVPSO, BBPSO, and PSO-2. However, it is also observed from Figure 10.11 that PSO-2 attained a notably worse average rank than each of the GVPSO and BBPSO configurations. The remaining PSO variants, namely PSO-1, PSO-7, PSO-TVAC, and PSO-iRC-p5 all performed significantly worse than GVPSO.

When examining the critical difference plot for the unimodal problems, as shown in Figure 10.12, PSO-1 demonstrated the lowest average rank, despite there being no significant difference noted among all examined PSO variants. A further noteworthy observation is that the average ranks for the GVPSO configurations were all better than those of the BBPSO configurations, further demonstrating the superiority of the proposed movement mechanism of GVPSO over that of the BBPSO algorithm.



(a) Diversity, f_{27}



(b) Movement, f_{27}

Figure 10.10: Diversity and movement profiles for f_{27}.

Finally, Figure 10.13 presents the critical difference plot depicting the results over the multimodal problem instances. It is again noted that the best performance was attained by one of the configurations of GVPSO. Note, however, that no significant difference in performance was observed between any of the configurations of GVPSO and BBPSO. Figure 10.13 shows there was a striking difference in average rank between the probabilistic PSO variants (i.e., GVPSO and BBPSO) and the conventional PSO variants, thus providing a clear indication that the probabilistic approaches were superior when faced with multimodal environments.

10.4 Summary

This chapter proposed a new PSO variant, entitled GVPSO, which generates particle positions probabilistically according to a Gaussian distribution. The GVPSO algorithm is loosely inspired by the BBPSO algorithm, but differs significantly from BBPSO by generating particles according to a distribution that more closely resembles the conventional PSO position update. An analysis of the single parameter of GVPSO was first performed, followed by a comparison of GVPSO to BBPSO and five additional PSO configurations. Results indicate that GVPSO generally outperforms the other strategies, with a slight degradation of performance noted in unimodal environments.

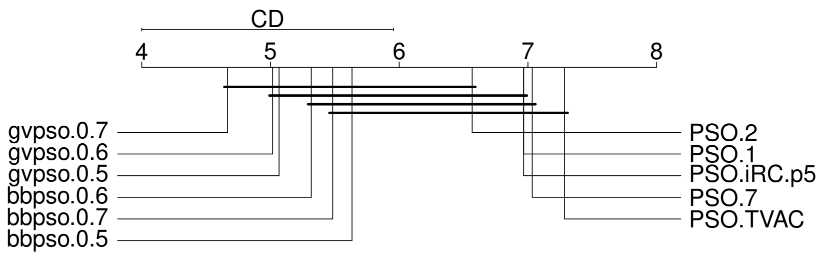


Figure 10.11: Comparison of GVPSO with other PSO variants over all 60 benchmark problems.

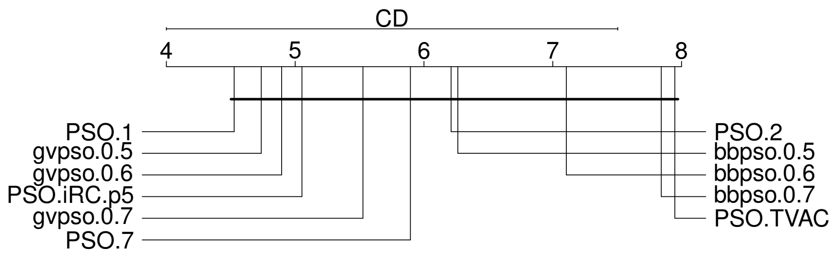


Figure 10.12: Comparison of GVPSO with other PSO variants over the 19 unimodal benchmark problems.

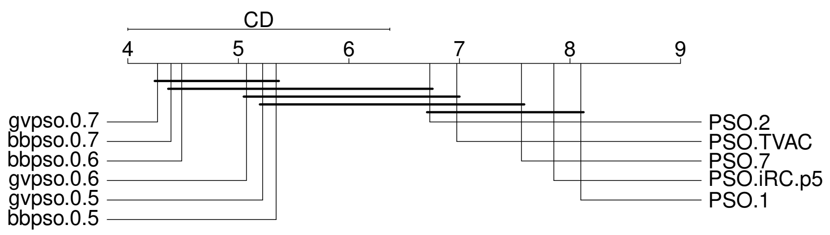


Figure 10.13: Comparison of GVPSO with other PSO variants over the 41 multimodal benchmark problems.

The following chapter provides a summary of the findings in this thesis. Additionally, avenues for future work are presented.

Chapter 11

Conclusions

This chapter provides a summary of the contributions and findings of thesis in Section 11.1, while potential avenues for future work are provided in Section 11.2.

11.1 Summary of Conclusions

The primary objectives of this thesis were to analyse existing parameter control mechanisms for the PSO algorithm, and to propose improved variants of PSO that do not rely on *a priori* values for the control parameters. This section summarizes the findings made while achieving the corresponding sub-objectives.

The first objective of this thesis was to provide a thorough review of existing SAPSO techniques. Chapter 3 provided an extensive review of 29 SAPSO algorithms. The examined algorithms were broadly classified into two categories, namely those that made use of introspective observation to refine the values of the control parameters, and those that did not. From this review, it was found that a majority of the past efforts have focused on adapting only the inertia weight parameter, while neglecting the cognitive and social acceleration coefficients. Furthermore, it was found that only nine of the 29 examined algorithms actually reduced the number of control parameters compared to the standard PSO. Thus, there is a clear need for new PSO variants that control the values for each of the three parameters, while introducing minimal new parameters.

The second objective of this thesis was to provide an analysis of the search behaviour for existing SAPSO techniques. Chapters 4 and 5 achieved this goal by deriving exact conditions for order-2 stability to be observed for various SAPSO strategies. In addition, empirical evidence was provided in support of the theoretical findings. Chapter 4 found that the majority of the examined algorithms exhibited either premature convergence or rapid divergence. Chapter 5 provided a more in-depth empirical investigation of inertia weight control strategies, ultimately concluding that none of the examined strategies, with the exception of a random inertia weight, outperformed a static inertia weight. These results clearly indicate the sad state of SAPSO research.

The third objective of this thesis was to investigate the parametrization of PSO to identify areas within the parameter space that lead to good performance. Chapter 6 provided a preliminary study of 1012 parameter configurations, leading to the discovery of a region in parameter space where the parameters tended to perform well. This region was notably correlated with proximity to the boundaries of the stable region given by Equation (2.5). It was also found, by examining the star and ring topologies, that topology does play a role in determining the regions in parameter space that lead to good performance, and therefore, naively comparing results using the same parametrizations with both the star and ring topologies may be misleading. Furthermore, it was found that neither modality nor separability had a significant impact on the optimal region for parameters. Chapter 7 expanded this study for the star topology by also investigating cognitively- and socially-biased parameter configurations. A total of 3036 parameter configurations were examined and it was found that the balance between the social and cognitive acceleration coefficients does impact the optimal parameter region. Finally, this chapter proposed a new recommendation for the selection of PSO parameters.

The fourth objective of this thesis was to provide evidence in support of the real-time adaptation of PSO control parameters. Chapter 7, in addition to the contributions listed above, also investigated the time-dependence of PSO control parameter values. It was found that the ideal parameter values to employ shift over time, thereby indicating that the optimal parameter values to employ change as the search progressed. Specifically, larger values for the cognitive and social acceleration coefficients were preferred as the search progressed. This topic was revisited in Chapter 9, where it was found that short-term performance in the PSO algorithm was not necessarily indicative of long-term performance. More specifically, parameter configurations that performed well initially tended to perform worse long-term. These results indicate that, not only are adaptive control strategies warranted, but that adaptive mechanisms that naively rely on early performance indicators may lead to sub-optimal performance.

The fifth and final objective of this thesis was to propose novel PSO variants that do not rely on *a priori* specification of values for the conventional control parameters. Based on the optimal parameter regions found in Chapters 6 and 7, Chapter 8 proposed a restricted region from which to select control parameter values. An adaptive PSO algorithm, entitled PSO-iRC, was proposed by sampling parameter configurations from the discovered region according to two different update schedules. It was found that re-sampling the values for the control parameters at fixed intervals performed best. Note that the PSO-iRC algorithm eliminates the need to specify values for ω, c_1, and c_2, but introduces one new parameter. When compared to 14 PSO parameter configurations recommended by the literature, the PSO-iRC algorithm was found to be among the top performing strategies for nearly all examined environment types.

A second PSO variant was proposed in Chapter 9 based on novel performance predictors built using the data from Chapter 7. These models were built to predict the long-term performance of PSO parameter configurations based on two simple landscape characteristics. The accuracy for these models ranged from 73.64% to 95.97%. A parameter-free algorithm, namely CMPSO, was then proposed by using the trained models to select a parameter configuration that was predicted to perform well on the current problem. The results indicated that the CMPSO algorithm attained performance that was insignificantly different from other top-performing PSO variants, while eliminating the need to specify values for ω, c_1, and c_2.

Chapter 10 proposed a probabilistic PSO variant that did not rely on the conventional PSO control parameters. Inspired by the BBPSO algorithm, the proposed GVPSO algorithm generated particle positions probabilistically according to a Gaussian distribution. However, the distribution used to generate particle positions in the GVPSO algorithm more closely resembled the position update mechanism found in the standard PSO. Note that the GVPSO algorithm eliminates the need to specify values for ω, c_1, and c_2, at the expense of introducing one new parameter. Results indicated that the GVPSO algorithm generally outperformed the other examined PSO variants.

11.2 Future Work

There are a number of avenues of future work that have emerged as a result of this thesis. These avenues of future work are briefly discussed in this section.

A first avenue of future work lies in the analysis of the search behaviour under different algorithmic configurations. Notably, previous studies have provided evidence suggesting that the dimensionality of the problem has an effect on the performance of various parameter configurations. Therefore, analysing the optimal parameter regions, and other results from this thesis, with respect to other algorithmic and benchmark configurations, is an important avenue of future work.

Another possible extension of this work lies in further empirical examination of the proposed algorithms. It was discovered that both the CMPSO and GVPSO algorithms suffered from degraded performance on unimodal environments. Therefore, further investigation to determine both the cause, and remedies, for this behaviour are needed. Additionally, the scalability of the proposed algorithms should be investigated.

An important extension of this work is to discover a mechanism for the real-time prediction of performance for a given set of PSO parameter values. Such a mechanism would have tremendous implications for SAPSO algorithms as it would allow for real-time prediction of long-term success. As found in Chapter 9, naively relying on only the objective fitness can be extremely misleading. It is hypothesized that, at minimum, information regarding the fitness, particle movement, and diversity will be required for such a real-time prediction to be made. An effective parameter search is, therefore, very likely to be multi-objective in nature. It should thus be investigated whether a multi-objective formulation of the parameter tuning problem will lead to new insights.

An additional extension of this research is to further investigate the effects of fitness landscape characteristics on the performance of the PSO algorithm. Previous research has found that *a priori* fitness landscape characteristics can be used to predict the performance of the PSO with reasonable accuracy [77]. Future work should investigate whether such techniques can be applied in real-time to prevent the need for *a priori* characterization. Furthermore, it is reasonable to assume that not only the characteristics of the global landscape are of importance, but also the visible landscape induced by the current state of the swarm. Additionally, a more thorough investigation of the performance implications of various neighbourhood topologies, with respect to the parametrization of PSO, is needed given that this may also be correlated with the landscape characteristics of the current problem.

Bibliography

[1] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing.* IEEE, 2011, pp. 633–640.

[2] J. Barrera and J. J. Flores, "Equivalence of the constriction factor and inertia weight models in particle swarm optimization: a geometric series analysis," in *Proceedings of the 2008 Seventh Mexican International Conference on Artificial Intelligence.* IEEE, 2008, pp. 188–191.

[3] A. Beham, E. Pitzer, and M. Affenzeller, "Fitness landscape based parameter estimation for robust taboo search," in *Proceedings of the 2013 Conference on Computer Aided Systems Theory.* Springer, 2013, pp. 292–299.

[4] T. Beielstein, "Tuning PSO parameters through sensitivity analysis," Universitat Dortmund, Tech. Rep., 2002.

[5] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference.* ACM, 2002, pp. 11–18.

[6] M. R. Bonyadi and Z. Michalewicz, "Stability analysis of the particle swarm optimization without stagnation assumption," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 814–819, 2016.

[7] M. R. Bonyadi and Z. Michalewicz, "Analysis of stability, local convergence, and transformation sensitivity of a variant of the particle swarm optimization algo-

rithm," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 370–385, 2016.

[8] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2017.

[9] M. Bonyadi and Z. Michalewicz, "Impacts of coefficients on movement patterns in the particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 1–1, 2016.

[10] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*. IEEE, 2007, pp. 120–127.

[11] L. Breiman, "Random forests," *Maching Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proceedings of the Workshop on Particle Swarm Optimization*, vol. 1. Purdue School of Engineering and Technology, 2001, pp. 1–6.

[13] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Computers & Operations Research*, vol. 33, no. 3, pp. 859–871, 2006.

[14] P. Chauhan, K. Deep, and M. Pant, "Novel inertia weight strategies for particle swarm optimization," *Memetic Computing*, vol. 5, no. 3, pp. 229–251, 2013.

[15] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural exponential inertia weight strategy in particle swarm optimization," in *Proceedings of the 2006 6th World Congress on Intelligent Control and Automation*, vol. 1. IEEE, 2006, pp. 3672–3675.

[16] H.-h. Chen, G.-q. Li, and H.-l. Liao, "A self-adaptive improved particle swarm optimization algorithm and its application in available transfer capability calculation," in *Proceedings of the 2009 Fifth International Conference on Natural Computation*, vol. 3. IEEE, 2009, pp. 200–205.

[17] C. W. Cleghorn and A. Engelbrecht, "Particle swarm optimizer: the impact of unstable particles on performance," in *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence.* IEEE, 2016, pp. 1–7.

[18] C. W. Cleghorn and A. P. Engelbrecht, "A generalized theoretical deterministic particle swarm model," *Swarm Intelligence*, vol. 8, no. 1, pp. 35–59, 2014.

[19] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm convergence: an empirical investigation," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation.* IEEE, 2014, pp. 2524–2530.

[20] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm variants: standardized convergence analysis," *Swarm Intelligence*, vol. 9, no. 2-3, pp. 177–203, 2015.

[21] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption," *Swarm Intelligence*, pp. 1–22, 2017.

[22] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, vol. 3. IEEE, 1999, pp. 1951–1957.

[23] M. Clerc, "Stagnation analysis in particle swarm optimisation or what happens when nothing happens," HAL, Tech. Rep. 1, 2006.

[24] Y. Cooren, M. Clerc, and P. Siarry, "Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm," *Swarm Intelligence*, vol. 3, no. 2, pp. 149–178, 2009.

[25] K. Deep, P. Chauhan, and M. Pant, "A new fine grained inertia weight particle swarm optimization," in *Proceedings of the 2011 World Congress on Information and Communication Technologies.* IEEE, 2011, pp. 424–429.

[26] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary

and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[27] C. Dong, G. Wang, Z. Chen, and Z. Yu, "A method of self-adaptive inertia weight for PSO," in *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, vol. 1. IEEE, 2008, pp. 1195–1198.

[28] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, vol. 1, no. 7. IEEE, 2000, pp. 84–88.

[29] R. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation.* IEEE, 2001, pp. 94–100.

[30] A. Engelbrecht, "Particle swarm optimization: velocity initialization," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation.* IEEE, 2012, pp. 1–8.

[31] A. Engelbrecht, "Particle swarm optimization: global best or local best?" in *Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence.* IEEE, 2013, pp. 124–135.

[32] A. Engelbrecht, "Roaming behavior of unconstrained particles," in *Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence.* IEEE, 2013, pp. 104–111.

[33] S.-K. S. Fan and Y.-Y. Chiu, "A decreasing inertia weight particle swarm optimizer," *Engineering Optimization*, vol. 39, no. 2, pp. 203–228, 2007.

[34] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the Second International Conference on Innovative Computing, Information and Control.* IEEE, 2007, pp. 475–475.

[35] M. Friedman, "A Comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[36] M. Friedman, "The use of ranks to avoid the assumtion of normality implisit in the analysis of variance." *Journal of American Statistical Association*, vol. 32, no. 32, pp. 675–701, 1937.

[37] Y.-l. Gao, X.-h. An, and J.-m. Liu, "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation," in *Proceedings of the 2008 International Conference on Computational Intelligence and Security.* IEEE, 2008, pp. 61–65.

[38] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2014, pp. 1641–1649.

[39] V. Gazi, "Stochastic stability analysis of the particle dynamics in the PSO algorithm," in *Proceedings of the 2012 IEEE International Symposium on Intelligent Control.* IEEE, 2012, pp. 708–713.

[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[41] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "Inertia weight control strategies for particle swarm optimization," *Swarm Intelligence*, vol. 10, no. 4, pp. 267–305, 2016.

[42] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "The sad state of self-adaptive particle swarm optimizers," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation.* IEEE, 2016, pp. 431–439.

[43] K. R. Harrison, B. M. Ombuki-Berman, and A. P. Engelbrecht, "Optimal parameter regions for particle swarm optimization algorithms," in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation.* IEEE, 2017, pp. 349–356.

[44] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "Self-adaptive particle swarm optimization: a review and analysis of convergence," *Swarm Intelligence*, vol. 12, no. 3, pp. 187–226, 2018.

[45] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 20–35, 2018.

[46] A. Hashemi and M. Meybodi, "A note on the learning automata based algorithms for adaptive parameter selection in PSO," *Applied Soft Computing*, vol. 11, no. 1, pp. 689–705, 2011.

[47] A. B. Hashemi and M. R. Meybodi, "Adaptive parameter selection scheme for PSO: A learning automata approach," in *Proceedings of the 2009 14th International CSI Computer Conference.* IEEE, 2009, pp. 403–411.

[48] J.-z. Hu, J. Xu, J.-q. Wang, and T. Xu, "Research on particle swarm optimization with dynamic inertia weight," in *Proceedings of the 2009 International Conference on Management and Service Science.* IEEE, 2009, pp. 1–4.

[49] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009.

[50] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the 2011 Learning and Intelligent Optimization Conference.* Springer, 2011, pp. 507–523.

[51] M. Jiang, Y. Luo, and S. Yang, "Particle swarm optimization - stochastic trajectory analysis and parameter selection," in *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization.* I-Tech Education and Publishing, 2007, pp. 179–198.

[52] M. Jiang, Y. Luo, and S. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Information Processing Letters*, vol. 102, no. 1, pp. 8–16, 2007.

[53] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm," *Chaos, Solitons & Fractals*, vol. 37, no. 3, pp. 698–705, 2008.

[54] D. Ju-Long, "Control problems of grey systems," *Systems & Control Letters*, vol. 1, no. 5, pp. 288–294, 1982.

[55] S. Jun and L. Jian, "An improved self-adaptive particle swarm optimization algorithm with simulated annealing," in *Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application.* IEEE, 2009, pp. 396–399.

[56] V. Kadirkamanathan, K. Selvarajah, and P. Fleming, "Stability analysis of the particle dynamics in particle swarm optimizer," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 245–255, 2006.

[57] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.

[58] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* IEEE, 2003, pp. 80–87.

[59] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[60] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the 2002 Congress on Evolutionary Computation.*, vol. 2. IEEE, 2002, pp. 1671–1676.

[61] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation.* ACM Press, 2009, pp. 1749–1750.

[62] K. Lei, Y. Qiu, and Y. He, "A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization," in *Proceedings of the 2006 1st International Symposium on Systems and Control in Aerospace and Astronautics.* IEEE, 2006, pp. 977–980.

[63] B. J. Leonard and A. P. Engelbrecht, "On the optimality of particle swarm parameters in dynamic environments," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1564–1569.

[64] M.-S. Leu and M.-F. Yeh, "Grey particle swarm optimization," *Applied Soft Computing*, vol. 12, no. 9, pp. 2985–2996, 2012.

[65] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 627–646, 2012.

[66] X. Li, H. Fu, and C. Zhang, "A self-adaptive particle swarm optimization algorithm," in *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, vol. 5. IEEE, 2008, pp. 186–189.

[67] Z. Li and G. Tan, "A self-adaptive mutation-particle swarm optimization algorithm," in *Proceedings of the 2008 Fourth International Conference on Natural Computation*, vol. 1. IEEE, 2008, pp. 30–34.

[68] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing*, vol. 19, no. 2, pp. 431–448, 2015.

[69] W. H. Lim and N. A. Mat Isa, "Particle swarm optimization with adaptive time-varying topology connectivity," *Applied Soft Computing*, vol. 24, pp. 623–642, 2014.

[70] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[71] Q. Liu, "Order-2 stability analysis of particle swarm optimization," *Evolutionary Computation*, vol. 23, no. 2, pp. 187–216, 2015.

[72] Q. Liu, W. Wei, H. Yuan, Z.-H. Zhan, and Y. Li, "Topology selection for particle swarm optimization," *Information Sciences*, vol. 363, no. 2015, pp. 154–173, 2016.

[73] E. J. Lusk and H. Wright, “Deriving the probability density for sums of uniform random variables,” *The American Statistician*, vol. 36, no. 2, pp. 128–130, 1982.

[74] N. Lynn, M. Z. Ali, and P. N. Suganthan, “Population topologies for particle swarm optimization and differential evolution,” *Swarm and Evolutionary Computation*, 2017.

[75] K. M. Malan and A. P. Engelbrecht, “A survey of techniques for characterising fitness landscapes and some possible ways forward,” *Information Sciences*, vol. 241, pp. 148–163, 2013.

[76] K. M. Malan and A. P. Engelbrecht, “Ruggedness, funnels and gradients in fitness landscapes and the effect on PSO performance,” in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation.* IEEE, 2013, pp. 963–970.

[77] K. M. Malan, “Characterising continuous optimisation problems for particle swarm optimisation performance prediction,” Ph.D. dissertation, University of Pretoria, 2014.

[78] F. Mascia, P. Pellegrini, M. Birattari, and T. Stützle, “An analysis of parameter adaptation in reactive tabu search,” *International Transactions in Operational Research*, vol. 21, no. 1, pp. 127–152, 2014.

[79] M. A. Montes de Oca, J. Peña, T. Stützle, C. Pinciroli, and M. Dorigo, “Heterogeneous particle swarm optimizers,” in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation.* IEEE, 2009, pp. 698–705.

[80] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*, 1st ed. Prentice-Hall, 1989.

[81] F. V. Nepomuceno and A. P. Engelbrecht, “A self-adaptive heterogeneous pso for real-parameter optimization,” in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation.* IEEE, 2013, pp. 361–368.

[82] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.

[83] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation.* IEEE, 2008, pp. 1128–1134.

[84] A. Panda, S. Ghoshal, A. Konar, B. Banerjee, and A. K. Nagar, "Static learning particle swarm optimization with enhanced exploration and exploitation using adaptive swarm size," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation.* IEEE, 2016, pp. 1869–1876.

[85] B. B. Pandey, S. Debbarma, and P. Bhardwaj, "Particle swarm optimization with varying inertia weight for solving nonlinear optimization problem," in *Proceedings of the 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization.* IEEE, 2015, pp. 1–5.

[86] B. Panigrahi, V. Ravikumar Pandi, and S. Das, "Adaptive particle swarm optimization approach for static and dynamic economic load dispatch," *Energy Conversion and Management*, vol. 49, no. 6, pp. 1407–1415, 2008.

[87] R. Pavón, F. Díaz, R. Laza, and V. Luzón, "Automatic parameter tuning with a Bayesian case-based reasoning system. A case of study," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3407–3420, 2009.

[88] P. Pellegrini, T. Stützle, and M. Birattari, "A critical analysis of parameter adaptation in ant colony optimization," *Swarm Intelligence*, vol. 6, no. 1, pp. 23–48, 2012.

[89] R. Poli, "Mean and variance of the sampling distribution of particle swarm optimizers during stagnation," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 712–721, 2009.

[90] R. Poli and D. Broomhead, "Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation.* ACM Press, 2007, pp. 134–141.

[91] J. R. Quinlan, "Discovering rules by induction from large collections of examples," *Expert Systems in the Micro Electronics Age*, 1979.

[92] J. R. Quinlan, *C4.5: programs for machine learning.* Morgan Kaufmann, 1993.

[93] J. Rada-Vilela, M. Zhang, and W. Seah, "A performance study on synchronous and asynchronous updates in particle swarm optimization," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation.* ACM Press, 2011, p. 21.

[94] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[95] M. Reza Bonyadi, X. Li, and Z. Michalewicz, "A hybrid particle swarm with a time-adaptive topology for constrained optimization," *Swarm and Evolutionary Computation*, vol. 18, pp. 22–37, 2014.

[96] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263–278, 1996.

[97] J. P. Shaffer, "Modified sequentially rejective multiple test procedures," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 826–831, 1986.

[98] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation.* IEEE, 1998, pp. 69–73.

[99] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3. IEEE, 1999, pp. 1945–1950.

[100] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, Tech. Rep. May, 2005.

[101] M. Sugeno, *Industrial applications of fuzzy control*, 1st ed. Elsevier, 1985.

[102] J. Sun, W. Xu, and B. Feng, "Adaptive parameter control for quantum-behaved particle swarm optimization on individual level," in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4. IEEE, 2005, pp. 3049–3054.

[103] M. Taherkhani and R. Safabakhsh, "A novel stability-based adaptive inertia weight for particle swarm optimization," *Applied Soft Computing*, vol. 38, no. 4, pp. 281–295, 2016.

[104] M. Tanweer, S. Suresh, and N. Sundararajan, "Self regulating particle swarm optimization algorithm," *Information Sciences*, vol. 294, pp. 182–202, 2015.

[105] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[106] F. van den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, 2001.

[107] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[108] E. Van Zyl and A. P. Engelbrecht, "A subspace-based method for PSO initialization," in *Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 226–233.

[109] E. van Zyl and A. Engelbrecht, "Comparison of self-adaptive particle swarm optimizers," in *Proceedings of the 2014 IEEE Symposium on Swarm Intelligence*. IEEE, 2014, pp. 1–9.

[110] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4515–4538, 2011.

[111] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[112] Z. Wu and J. Zhou, "A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment," in *Proceedings of the 2007 International Conference on Computational Intelligence and Security*. IEEE, 2007, pp. 133–136.

[113] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4560–4569, 2013.

[114] Z. Xuanping, Q. G. Du Yuping, and Q. Zheng, "Adaptive particle swarm algorithm with dynamically changing inertia weight," *Journal of Xi'an Jiaotong University*, vol. 10, pp. 1039–1042, 2005.

[115] C. Yang, W. Gao, N. Liu, and C. Song, "Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight," *Applied Soft Computing*, vol. 29, pp. 386–394, 2015.

[116] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1205–1213, 2007.

[117] K. Yasuda, N. Iwasaki, G. Ueno, and E. Aiyoshi, "Particle swarm optimization: a numerical stability analysis and parameter adjustment based on swarm activity," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 3, no. 6, pp. 642–659, 2008.

[118] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, “Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements,” in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation.* IEEE, 2013, pp. 2337–2344.

[119] Z.-h. Zhan, Jing Xiao, J. Zhang, and Wei-neng Chen, “Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis,” in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation.* IEEE, 2007, pp. 3276–3282.

[120] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[121] W. Zhang, D. Ma, J.-j. Wei, and H.-f. Liang, “A parameter selection strategy for particle swarm optimization based on particle positions,” *Expert Systems with Applications*, vol. 41, no. 7, pp. 3576–3584, 2014.

Appendix A

Benchmark Problems

This appendix provides both the equation and feasible domain for the set of benchmark problems used in this thesis. A summary of the benchmark problems is given in Table A.1, where the column 'F' specifies the unique identifier associated with each function. The column 'M' specifies the modality, where the values 'U' and 'M' denote unimodal and multimodal landscapes respectively, and the column 'S' denotes the separability, where 'S' and 'NS' denote separable and non-separable functions respectively. Note that for some functions, shifted, rotated, shifted and rotated, and noisy variants were also used, resulting in a total of 60 functions. The configuration of the modified versions are indicated in the columns 'Sh', 'R', 'ShR', and 'N' respectively. Finally, a ✓ in column 'C' denotes a composition function. This suite of functions has been demonstrated to include a range of different landscape characteristics, including both smooth and rugged fitness landscapes along with a variety of gradients [38].

A function, f, was shifted using

$$f^{Sh}(\mathbf{x}) = f(\mathbf{x} - \gamma) + \beta,$$

where β and γ are constants. Rotation was implemented using either a randomly generated orthonormal rotation matrix, denoted by "ortho", or a linear transformation matrix, denoted by "linear". In either scenario, the rotation was performed using Salomon's method [96], with a new rotation matrix computed for each of the independent runs using the condition number provided. The rotated functions, denoted by f^R, were then computed by multiplying the decision vector $\mathbf{x}$ by the transpose of the rotation matrix. Noisy functions, denoted by f^N, were generated by multiplying each decision variable by a noise value sampled from a Gaussian distribution with the specified mean and deviation. Table A.1 provides the configuration parameters for the shifted, rotated, rotated and shifted, and noisy versions of the functions in the columns 'Sh', 'R', 'ShR', and 'N', respectively.

It should be noted that the composition functions $f_{27} - f_{37}$ are equal to functions $f_{15} - f_{25}$ from the CEC 2005 benchmark set [100]. For each of these problems, the composed functions are each rotated using linear transformation matrices with independent condition numbers. The reader is directed to [100] for specific details regarding the configurations of these functions.

The equations for each benchmark problem are as follows.

f_1, the absolute value function, defined as

$$f_1(\mathbf{x}) = \sum_{j=1}^{n_x} |x_j| \tag{A.1}$$

with each $x_j \in [-100, 100]$.

f_2, the ackley function, defined as

$$f_2(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n_x}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n_x}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e \tag{A.2}$$

with each $x_j \in [-32.768, 32.768]$.

f_3, the alpine function, defined as

$$f_3(\mathbf{x}) = \sum_{j=1}^{n_x} |x_j \sin(x_j) + 0.1x_j| \tag{A.3}$$

with each $x_j \in [-10, 10]$.

f_4, the egg holder function, defined as

$$\begin{aligned} f_4(\mathbf{x}) = \sum_{j=1}^{n_x-1} \Bigg(& -(x_{j+1} + 47)\sin\left(\sqrt{|x_{j+1} + x_j/2 + 47|}\right) \\ & + \sin\left(\sqrt{|x_j - (x_{j+1} + 47)|}\right)(-x_j)\Bigg) \end{aligned} \tag{A.4}$$

with each $x_j \in [-512, 512]$.

f_5, the elliptic function, defined as

$$f_5(\mathbf{x}) = \sum_{j=1}^{n_x} (10^6)^{\frac{j-1}{n_x-1}} x_j^2 \tag{A.5}$$

with each $x_j \in [-100, 100]$.

f_6, the griewank function, defined as

$$f_6(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{j=1}^{n_x} x_j^2 - \prod_{j=1}^{n_x} \cos\left(\frac{x_j}{\sqrt{j}}\right) \tag{A.6}$$

with each $x_j \in [-600, 600]$.

f_7, the hyperellipsoid function, defined as

$$f_7(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^2 \tag{A.7}$$

with each $x_j \in [-5.12, 5.12]$.

f_8, the michalewicz function, defined as

$$f_8(\mathbf{x}) = -\sum_{j=1}^{n_x} \sin(x_j) \left(\sin\left(\frac{j x_j^2}{\pi}\right)\right)^{2m} \tag{A.8}$$

with each $x_j \in [0, \pi]$ and $m = 10$.

f_9, the norwegian function, defined as

$$f_9(\mathbf{x}) = \prod_{j=1}^{n_x} \left(\cos(\pi x_j^3) \left(\frac{99 + x_j}{100}\right)\right) \tag{A.9}$$

with each $x_j \in [-1.1, 1.1]$.

f_{10}, the quadric function, defined as

$$f_{10}(\mathbf{x}) = \sum_{i=1}^{n_x} \left(\sum_{j=1}^{i} x_j\right)^2 \tag{A.10}$$

with each $x_j \in [-100, 100]$.

f_{11}, the quartic function, defined as

$$f_{11}(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^4 \tag{A.11}$$

with each $x_j \in [-1.28, 1.28]$.

f_{12}, the rastrigin function, defined as

$$f_{12}(\mathbf{x}) = 10n_x + \sum_{j=1}^{n_x}(x_j^2 - 10\cos(2\pi x_j)) \tag{A.12}$$

with each $x_j \in [-5.12, 5.12]$.

f_{13}, the rosenbrock function, defined as

$$f_{13}(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2\right) \tag{A.13}$$

with each $x_j \in [-30, 30]$.

f_{14}, the saloman function, defined as

$$f_{14}(\mathbf{x}) = -\cos\left(2\pi\sum_{j=1}^{n_x} x_j^2\right) + 0.1\sqrt{\sum_{j=1}^{n_x} x_j^2} + 1 \tag{A.14}$$

with each $x_j \in [-100, 100]$.

f_{15}, the generalized schaffer 6 function, also known as the pathological function, defined as

$$f_{15}(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(0.5 + \frac{\sin^2(100x_j^2 + x_{j+1}^2) - 0.5}{1 + 0.001(x_j^2 - 2x_j x_{j+1} + x_{j+1}^2)^2}\right) \tag{A.15}$$

with each $x_j \in [-100, 100]$.

f_{16}, the schwefel 1.2 function, defined as

$$f_{16}(\mathbf{x}) = \sum_{i=1}^{n_x}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{A.16}$$

with each $x_j \in [-100, 100]$.

f_{17}, the schwefel 2.6 function, defined as

$$f_{17}(\mathbf{x}) = \max_j\{|\mathbf{A}_j\mathbf{x} - \mathbf{B}_j|\} \tag{A.17}$$

with each $x_j \in [-100, 100]$, each $a_{ij} \in \mathbf{A}$ is uniformly sampled from $U(-500, 500)$ such that $\det(\mathbf{A}) \neq 0$, and each $\mathbf{B}_j = \mathbf{A}_j\mathbf{r}$ where each $r_i \in \mathbf{r}$ is uniformly sampled from $U(-100, 100)$. A shifted version of f_{17} was also used.

f_{18}, the schwefel 2.13 function, defined as

$$f_{18}(\mathbf{x}) = \sum_{j=1}^{n_x}(\mathbf{A}_j - \mathbf{B}_j(\mathbf{x}))^2 \tag{A.18}$$

with each $x_j \in [-\pi, \pi]$, and

$$\mathbf{A}_j = \sum_{i=1}^{n_x}(a_{ij}\sin(\alpha_i) + b_{ij}\cos(\alpha_i))$$

and

$$\mathbf{B}_j(\mathbf{x}) = \sum_{i=1}^{n_x}(a_{ij}\sin(x_i) + b_{ij}\cos(x_i))$$

where $a_{ij} \in \mathbf{A}, b_{ij} \in \mathbf{B}$, $a_{ij}, b_{ij} \sim U(-100, 100)$, and $\alpha_i \sim U(-\pi, \pi)$. A shifted version of f_{18} was also used.

f_{19}, the schwefel 2.21 function, defined as

$$f_{17}(\mathbf{x}) = \max_j\{|x_j|, 1 \leq j \leq n_x\} \tag{A.19}$$

with each $x_j \in [-100, 100]$.

f_{20}, the schwefel 2.22 function, defined as

$$f_{18}(\mathbf{x}) = \sum_{j=1}^{n_x}|x_j| + \prod_{j=1}^{n_x}|x_j| \tag{A.20}$$

with each $x_j \in [-10, 10]$.

f_{21}, the shubert function, defined as

$$f_{19}(\mathbf{x}) = \prod_{j=1}^{n_x}\left(\sum_{i=1}^{5}(i\cos((i+1)x_j + i))\right) \tag{A.21}$$

with each $x_j \in [-10, 10]$.

f_{22}, the spherical function, defined as

$$f_{20}(\mathbf{x}) = \sum_{j=1}^{n_x} x_j^2 \tag{A.22}$$

with each $x_j \in [-5.12, 5.12]$.

f_{23}, the step function, defined as

$$f_{21}(\mathbf{x}) = \sum_{j=1}^{n_x} (\lfloor x_j + 0.5 \rfloor)^2 \tag{A.23}$$

with each $x_j \in [-100, 100]$.

f_{24}, the vincent function, defined as

$$f_{22}(\mathbf{x}) = -\left(1 + \sum_{j=1}^{n_x} \sin(10\sqrt{x_j})\right) \tag{A.24}$$

with each $x_j \in [0.25, 10]$.

f_{25}, the weierstrass function, defined as

$$\begin{aligned} f_{25}(\mathbf{x}) = &\sum_{j=1}^{n_x} \left(\sum_{i=1}^{20} (a^i \cos(2\pi b^i (x_j + 0.5))) \right) \\ &- n_x \sum_{i=1}^{20} (a^i \cos(\pi b^i)) \end{aligned} \tag{A.25}$$

with each $x_j \in [-0.5, 0.5], a = 0.5$, and $b = 3$. A rotated and shifted version of f_{25} was also used.

f_{26}, a shifted expansion of the griewank and rosenbrock functions (Equations (A.6) and (A.13) respectively) with each $x_j \in [-3, 1]$. Note that f_{26} is equivalent to f_{13} from the 2005 CEC benchmark suite.

$f_{27} - f_{37}$, composition functions equivalent to $f_{15} - f_{25}$ from the 2005 CEC benchmark suite. All functions have each $x_j \in [-5, 5]$, with the exception of f_{37} which has each $x_j \in [2, 5]$.

Table A.1: Characteristics of the benchmark functions.

F	M	S	Sh	R	ShR	N	C
f_1	U	S					
f_2	M	NS	$\beta = -140$ $\gamma = 10$	ortho (1)	$\beta = -140$ $\gamma = -32$ linear (100)		
f_3	M	S					
f_4	M	NS					
f_5	U	S	$\beta = -450$ $\gamma = 10$	ortho (1)	$\beta = -450$ $\gamma = 10$ ortho (1)		
f_6	M	NS	$\beta = -180$ $\gamma = 10$	othro (1)	$\beta = -180$ $\gamma = -60$ linear (3)		
f_7	U	S					
f_8	M	S					
f_9	M	NS					
f_{10}	U	NS					
f_{11}	U	S				$N(0, 1)$	
f_{12}	M	S	$\beta = -330$ $\gamma = 2$	ortho (1)	$\beta = -330$ $\gamma = 1$ linear (2)		
f_{13}	M	NS	$\beta = 390$ $\gamma = 10$	ortho (1)			
f_{14}	M	NS					
f_{15}	M	NS			$\beta = -300$ $\gamma = 20$ linear (3)		

Continued on next page

Table A.1 – continued from previous page

F	M	S	Sh	R	ShR	N	C
f_{16}	U	NS	$\beta = -450$ $\gamma = 10$	ortho (1)		$N(0, 0.4)$ $\beta = -450$ $\gamma = 10$	
f_{17}	U	NS	$\beta = -310$				
f_{18}	M	NS	$\beta = -460$				
f_{19}	U	S					
f_{20}	U	S					
f_{21}	M	NS					
f_{22}	U	S	$\beta = -450$ $\gamma = 10$				
f_{23}	M	S					
f_{24}	M	S					
f_{25}	M	S			$\beta = 90$ $\gamma = 0.1$ linear (5)		
f_{26}	M	NS	$\beta = -130$ $\gamma = 1$				
f_{27}	M	NS					✓
f_{28}	M	NS		Yes			✓
f_{29}	M	NS				Yes	✓
f_{30}	M	NS		Yes			✓
f_{31}	M	NS		Yes			✓
f_{32}	M	NS		Yes			✓
f_{33}	M	NS		Yes			✓
f_{34}	M	NS		Yes			✓
f_{35}	M	NS		Yes			✓
f_{36}	M	NS		Yes			✓
f_{37}	M	NS		Yes			✓

Appendix B

Acronyms

This appendix provides a list of acronyms used in this thesis. Acronyms are listed alphabetically and typeset in bold, with the meaning of the acronym alongside.

AIWPSO	Adaptive Inertia Weight Particle Swarm Optimization
APSO-VI	Adaptive Parameter Tuning of Particle Swarm Optimization Based on Velocity Information
APSO-ZZLC	Adaptive Particle Swarm Optimization by Zhan, Zhang, Li, and Chung
BBPSO	Bare Bones Particle Swarm Optimization
CDIW-PSO	Chaotic Descending Inertia Weight Particle Swarm Optimization
CI	Computational Intelligence
CMPSO	Classifier Model Particle Swarm Optimization
DAPSO	Dynamic Adaptation Particle Swarm Optimization
DE-PSO	Double Exponential Self-Adaptive Inertia Weight Particle Swarm Optimization
DW-PSO	Decreasing Inertia Weight Particle Swarm Optimization
FG-PSO	Fine Grained Inertia Weight Particle Swarm Optimization

GPSO	Grey Particle Swarm Optimization
GVPSO	Gaussian-Valued Particle Swarm Optimization
IPSO-CLL	Improved Particle Swarm Optimization by Chen, Li, and Liao
IPSO-LT	Improved Particle Swarm Optimization
LD-PSO	Logarithm Decreasing Particle Swarm Optimization
PSO	Particle Swarm Optimization
PSO-AIWF	Particle Swarm Optimization with Adaptive Inertia Weight Factor
PSO-ICSA	Particle Swarm Optimization with Simulated Annealing
PSO-iRC	Particle Swarm Optimization with Improved Random Constants
PSO-LDIW	Particle Swarm Optimization with Linearly-Decreasing Inertia Weight
PSO-NEIW	Particle Swarm Optimization with Natural Exponent Inertia Weight
PSO-NL	Particle Swarm Optimization with Nonlinear Inertia Weight
PSO-NLI	Particle Swarm Optimization with Nonlinear Improved Inertia Weight
PSO-OIW	Particle Swarm Optimization with Oscillating Inertia Weight
PSO-RAC	Particle Swarm Optimization with Random Acceleration Coefficients
PSO-RBI	Particle Swarm Optimization with Rank-Based Inertia Weight
PSO-RIW	Particle Swarm Optimization with Random Inertia Weight
PSO-SAIC	Self-Adaptive Particle Swarm Optimization with Individual Coefficient Adjustment
PSO-SIW	Particle Swarm Optimization with Sugeno Inertia Weight
PSO-TVAC	Particle Swarm Optimization with Time-Varying Acceleration Coefficients

SAPSO	Self-Adaptive Particle Swarm Optimization
SAPSO-DWCY	Self-Adaptive Particle Swarm Optimization by Dong, Wang, Chen, and Yu
SAPSO-LFZ	Self-Adaptive Particle Swarm Optimization by Li, Fu, and Zhang
SI	Swarm Intelligence
SRPSO	Self-Regulating Particle Swarm Optimization
UAPSO-A	Adventurous Unified Adaptive Particle Swarm Optimization

Appendix C

Symbols

This appendix lists the symbols used throughout this thesis, along with their corresponding definitions. For symbols defined in Chapter 3, the corresponding algorithms are provided in parenthesis.

C.1 Chapter 1

ω	Inertia weight
c_1	Cognitive acceleration coefficient
c_2	Social acceleration coefficient

C.2 Chapter 2

t	Time/iteration
i	Particle index
j	Problem dimension index
$v_{ij}(t)$	Particle velocity
$x_{ij}(t)$	Particle position
$r_{1ij}(t), r_{2ij}(t)$	Random values sampled from a uniform distribution $\sim U(0,1)$
$y_{ij}(t)$	Personal best position

$\hat{y}_{ij}(t)$	Neighbourhood best position
$\otimes$	Component-wise multiplication of two vectors
$v_{\mathbf{max},j}$	Maximum permitted velocity
$x_{\mathbf{min},j}, x_{\mathbf{max},j}$	Feasible bounds of the search space

C.3 Chapter 3

$f(\mathbf{x})(t)$	The objective fitness of solution $\mathbf{x}$
T	Maximum number of iterations
n_s	Swarm size
n_d	Number of problem dimensions
α, β, γ	Arbitrary control parameter values
ω_s, ω_f	Initial and final values for the inertia weight
$r(t)$	A random value sampled from a uniform distribution $\sim U(0, 1)$
$\|\|\mathbf{x}, \mathbf{y}\|\|_2$	Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$
c_{1s}, c_{1f}	Initial and final values for the cognitive acceleration coefficient
c_{2s}, c_{2f}	Initial and final values for the social acceleration coefficient
$\omega_{min}, \omega_{max}$	Minimum and maximum values for the inertia weight
s	Shape parameter for the inertia weight control function (PSO-SIW)
$z(t)$	The logistic map (CDIW-PSO)
ω_c	Base value for the inertia weight (PSO-NLI)
u	A control parameter value $\in$ [1.0001, 1.005] (PSO-NLI)
a	Constant controlling the convergence speed (LD-PSO)
k	Value controlling the period of a sinusoidal curve (PSO-OIW)
α	An exponent controlling the linearity of the inertia weight (PSO-NL)
$h_i(t)$	Evolutionary speed factor of a particle (DAPSO)
$s(t)$	Aggregation degree (DAPSO)
$\xi_i(t)$	Related distance (PSO-SAIC)
ω_a, ω_b	Arbitrary values for ω (PSO-SAIC)

c_{1a}, c_{1b}	Arbitrary values for c_1 (PSO-SAIC, PSO-ICSA)
c_{2a}, c_{2b}	Arbitrary values for c_2 (PSO-SAIC, PSO-ICSA)
$R_i(t)$	Fitness rank of a particle (PSO-RBI, SAPSO-DWCY)
$c_i(t)$	Convergence factor of a particle (IPSO-LT)
$d_i(t)$	Diffusion factor of a particle (IPSO-LT)
$\eta_i(t)$	Adaptive coefficient of a particle (PSO-ICSA)
S_1, S_2, S_3, S_4	Algorithmic state classifications (APSO-ZZLC)
$\delta(t)$	Control parameter step size (APSO-ZZLC)
n_ω	Number of inertia weight values (UAPSO-A)
n_c	Number of acceleration coefficient values (UAPSO-A)
τ	Threshold denoting a successful iteration (UAPSO-A)
$p_j(t)$	Selection probability for a parameter configuration (UAPSO-A)
$\|A\|$	Number of actions in the learning automaton (UAPSO-A)
a	Reward step size (UAPSO-A)
b	Penalty step size (UAPSO-A)
$r_{ij}(t)$	Relational coefficient of a particle (GPSO)
$\Delta_{ij}(t)$	Absolute distance between $\hat{y}_{ij}(t)$ and $x_{ij}(t)$ (GPSO)
$\Delta_{min}(t)$	Minimum value of Δ_{ij} (GPSO)
$\Delta_{max}(t)$	Maximum value of Δ_{ij} (GPSO)
ξ	Resolution control between $\Delta_{min}(t)$ and $\Delta_{max}(t)$ (GPSO)
$g_i(t)$	Grey relational coefficient of a particle (GPSO)
$v_{ideal}(t)$	Ideal velocity (APSO-VI)
v_s	Initial ideal velocity (APSO-VI)
$T_{0.95}$	Point at which 95% of the search is complete (APSO-VI)
Δ_s	Step size for the inertia weight value (APSO-VI, SRPSO)
η	Constant controlling the rate of acceleration (SRPSO)

C.4 Chapter 4

$\Delta(t)$	Average particle step size
$\Delta_{max}(t)$	Threshold for convergent behaviour classification
l, u	Lower and upper bounds for the feasible domain
Δ_p	Average parameter step size
CP	Proportion of parameters adhering to the convergence criterion
IP	Proportion of particles with a bound violation

C.5 Chapter 5

C	Shorthand for $c_1 + c_2$
$g(C)$	Expression used to reformulate the convergence criterion on ω
BRF	Best rank frequency
IP	Proportion of particles with a bound violation

C.6 Chapter 6

p	An arbitrary parameter configuration
$\mathcal{P}$	A set of parameter configurations
$R_g(p)$	Rank using the global best (star) topology
$R_l(p)$	Rank using the local best (ring) topology

C.7 Chapter 8

k	Arbitrary number of iterations

C.8 Chapter 9

d_i*	Distance between a sample i and the nearest optimum

X	A set of samples for calculating the dispersion metric
X^*	A subset of the samples from X with the best fitness values
E	'Excellent' classification
VG	'Very good' classification
G	'Good' classification
A	'Average' classification
P	'Poor' classification
VP	'Very poor' classification
T	'Terrible' classification
S	Training set
a_i	Arbitrary attribute of a training instance
$IG(a_i, S)$	Information gain from splitting S on attribute a_i
C	Set of classifications in S

C.9 Chapter 10

$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ
e	Exploitation ratio
$\Delta_{ij}(t)$	Ancillary particle position

Appendix D

Derived Publications

This appendix provides a listing of the publications derived from this thesis.

- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. The sad state of self-adaptive particle swarm optimizers. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 431–439. IEEE, 2016.
- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. Inertia weight control strategies for particle swarm optimization. In *Swarm Intelligence*, volume 10 (4), pages 267–305. Springer, 2016.
- K.R. Harrison, B.M. Ombuki-Berman, and A.P. Engelbrecht. Optimal parameter regions for particle swarm optimization algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 349–356. IEEE, 2017.
- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. An adaptive particle swarm optimization algorithm based on optimal parameter regions. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1606–1613. IEEE, 2017.
- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. Self-adaptive particle swarm optimization: a review and analysis of convergence. In *Swarm Intelligence*, volume 12 (3), pages 187–226. Springer, 2018.

- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm. In *Swarm and Evolutionary Computation*, volume 41, pages 20–35. Elsevier, 2018.

- K.R. Harrison, A.P. Engelbrecht, and B.M. Ombuki-Berman. Gaussian-valued particle swarm optimization. *Eleventh International Conference on Swarm Intelligence*, in press. 2018.