The Perils of Particle Swarm Optimization in High Dimensional Problem Spaces

by

Elre Talea Oldewage

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

May 2017

Publication data:

Elre Talea Oldewage. The Perils of Particle Swarm Optimization in High Dimensional Problem Spaces. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, April 2018.

The Perils of Particle Swarm Optimization in High Dimensional Problem Spaces

by

Elre Talea Oldewage
E-mail: eoldewage@csir.co.za

Abstract

Particle swarm optimization (PSO) is a stochastic, population-based optimization algorithm. PSO has been applied successfully to a variety of domains. This thesis examines the behaviour of PSO when applied to high dimensional optimization problems. Empirical experiments are used to illustrate the problems exhibited by the swarm, namely that the particles are prone to leaving the search space and never returning. This thesis does not intend to develop a new version of PSO specifically for high dimensional problems. Instead, the thesis investigates why PSO fails in high dimensional search spaces. Four different types of approaches are examined. The first is the application of velocity clamping to prevent the initial velocity explosion and to keep particles inside the search space. The second approach selects values for the acceleration coefficients and inertia weights so that particle movement is restrained or so that the swarm follows particular patterns of movement. The third introduces coupling between problem variables, thereby reducing the swarm's movement freedom and forcing the swarm to focus more on certain subspaces within the search space. The final approach examines the importance of initialization strategies in controlling the swarm's exploration to exploitation ratio. The thesis shows that the problems exhibited by PSOin high dimensions, particularly unwanted particle roaming, can not be fully mitigated by any of the techniques examined. The thesis provides deeper insight into the reasons for PSO's poor performance by means of extensive empirical tests and theoretical reasoning.

Keywords: Particle swarm optimization, high dimensions, large scale optimization

Supervisors : Prof. A. P. Engelbrecht
Dr. C. Cleghorn
Department : Department of Computer Science
Degree : Master of Science

"A single idea, if it is right, saves us the labor of an infinity of experiences."

Jacques Maritain

"To the scientist there is the joy in pursuing truth which nearly counteracts the depressing revelations of truth."

H.P. Lovecraft

Acknowledgements

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

- My supervisors, Prof. Andries Engelbrecht and Dr. Christopher Cleghorn for their continued insight and inspiration.
- Chris Serfontein and Klaus Müller at the CSIR for all the opportunities that they have presented to me.
- Hein, my husband, for listening to me describe experimental results first thing in the morning, admiring at graphs over dinner, and discussing stubborn proofs late at night, when he'd rather be sleeping.

This work has been supported by a studentship from the Council for Scientific and Industrial Research (CSIR).

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

A grey plume drifts through the low sky, like smoke but not smoke,
slow to disperse
reforming and palping like a long streak of foam on the sea; a grubby bag
turning, plastic and drifting
dividing in the sky: a shifting exclamation mark pulls out of shape
turns pale to vanishing, is gone.
A sound like pages riffling, like a thousand paper fans rustling, a darkening
in the air
turning in the low light all together
wheeling, breaking, re-combining, stretching again. Sky geometry.
- *Starlings* by J. D. Barker

1.1 Motivation

Computing hardware is cheaper and more powerful than ever before. With the advent of affordable, efficient computation, comes the ability to solve problems that were previously intractable. One such class of problems is that of large scale optimization (LSO). Large scale optimization refers to optimization problems that have many decision variables, typically more than one hundred [74]. Many optimization methods suffer from the "curse of dimensionality" [3], meaning that the methods' performance deteriorate as

problem dimensionality increases. The complexity of high dimensional problems has two facets: the solution space's hyper-volume usually increases exponentially as the number of dimensions in the problem space grows; and the characteristics of the problem may change with dimensionality.

Particle swarm optimization (PSO) is a nature-inspired optimization algorithm that has been successfully applied to a number of real-world problems [18, 53, 84]. The recent interest in LSO has inspired the invention of PSO variants, developed specifically to solve large scale problems [11, 78]. Although these PSO variants were successful in solving LSO problems, were developed without fully understanding and considering the reasons that standard PSO does not scale well.

The purpose of this thesis is not to develop another variant of PSO to solve high dimensional problems. Instead, this thesis provides an empirical analysis that is deeply rooted in theory, of the limiting behaviours exhibited by PSO in high dimensional spaces. Literature contains a number of studies regarding particular facets of PSO as problem dimensionality increases. This thesis aims to tie a number of these aspects together to form a broad picture of the strengths and weaknesses of standard PSO when applied to high dimensional problems and to unearth underlying reasons for the observed behaviour. This will give practitioners an intuitive idea of what kind of swarm behaviour is beneficial or detrimental when approaching LSO.

1.2 Objectives

This thesis is an in-depth study of PSO applied to high dimensional problems. Specifically, the aim is to determine what traits are beneficial in high dimensional search spaces and to determine the root causes underlying standard PSO's poor behaviour on high dimensional problems. The objectives of this work are summarized as follows:

- To illustrate the problems exhibited by PSO in high dimensional search spaces and to demonstrate that there is no "quick fix".
- To critically evaluate the use of velocity clamping in preventing velocity explosion.

- To consider the influence of the algorithm's parameters on the variance of particle positions and to select algorithm parameters that reduce particle roaming.
- To provide an empirical and theoretical analysis of the influence of component-wise stochastic scaling on the swarm's degrees of freedom.
- To empirically investigate the feasibility of controlling the ratio of exploration to exploitation by introducing grouping mechanisms for stochastic scaling.
- To examine the importance of initialization strategies in the swarm's behaviour.
- To determine what factors influence the swarm's behaviour positively and negatively in high dimensional spaces.

Based on these objectives, the thesis provides the following contributions:

- Due to velocity explosion in the first few iterations, particles leave the search space and never return for the duration of the search (even when the local and global attractors are guaranteed to be in the search space).
- Numerous empirical results show that in high dimensional spaces, swarms that perform local exploitation and search granularly perform better than swarms that attempt to explore the search space or take large steps.
- Velocity clamping can force particles to search in a granular manner by reducing the step sizes of particles.
- In high dimensional spaces, clamping per dimension performs better than clamping based on the magnitude of particle velocities according to empirical results.
- It is demonstrated that, as problem dimensionality increases, the optimal maximum clamping velocity decreases.
- Experiments show that particle roaming behaviour can be reduced by restricting the variance of particle positions.

- Particles with high inertia weights and low acceleration coefficients perform well in high dimensional spaces, as shown by experimentation. Such a particle is less likely to rapidly change direction, which prevents velocity explosion.
- By theoretical proof, if the stochastic scaling components are scalars and the swarm size is less than the problem dimensionality, then it is not possible for the swarm to reach every point in the search space.
- Empirical results show that dimensional coupling may be used to restrict the swarm's movement and encourage granular searching. Multiple dimensional coupling strategies are introduced, in which the same stochastic scalar is applied to all dimensions in a group.
- Opposite to the typical behavioural profile (which first performs exploration and then behaves exploitatively), dimension coupling strategies, that first restrict swarm movement and then gradually allow the swarm more freedom, perform better in high dimensional spaces.
- A new particle initialization strategy is proposed, which initializes particles within a small subspace of the search space.
- As problem dimensionality increases, initialization strategies that only generate positions within a small region of the search space perform better than strategies that attempt to maximize coverage of the search space.

1.3 Dissertation Outline

The remainder of the thesis proceeds as follows:

- **Chapter 2** provides an overview of the PSO algorithm and discusses the most important algorithm parameters such as neighbourhood topology, inertia weight and acceleration coefficients, amongst others.
- **Chapter 3** discusses high dimensional problems and considers the application of PSO to large scale problems. The chapter provides compelling empirical evidence

that shows that PSO exhibits several undesirable traits in high dimensional problem spaces such as velocity explosion and continuous particle roaming.

- **Chapter 4** critically examines the use of velocity clamping to prevent unwanted roaming. Two different mechanisms of clamping are examined, clamping per dimension and clamping based on the norm of the velocity vector.
- **Chapter 5** examines how the variance of particle positions can be constrained in order to reduce particle roaming (by selecting certain values for the inertia weight and acceleration coefficients). The chapter also considers the different movement patterns that can be exhibited by PSO and identifies movement patterns that are advantageous in high dimensional spaces.
- **Chapter 6** considers the effect of component-wise stochastic scaling on the swarm's freedom of movement. The chapter examines the importance of component-wise scaling both theoretically and empirically. The chapter also introduces coupling between variables, i.e. grouping stochastic scalars, as a mechanism to control the ratio of exploration to exploitation.
- **Chapter 7** discusses the importance of the swarm's initialization strategy and its effect on the swarm's behaviour in high dimensional problem spaces.
- **Chapter 8** concludes the thesis with a summary of the main findings and suggests future research.

The thesis is supplemented by a number of appendices, list below:

- **Appendix A** provides a detailed discussion of the benchmark suite that is used in the empirical experiments performed in the thesis.
- **Appendix B** list and defines the most important acronyms that are used or introduced throughout the thesis.
- **Appendix C** provides a list of the mathematical symbols used in this thesis, as well as the corresponding definitions.
- **Appendix D** lists the publications derived from this work.

The illustrations and figures presented throughout the thesis are intended to be viewed in color.

Chapter 2

Background

This chapter is intended to introduce the particle swarm optimization (PSO) algorithm and to discuss key aspects of the algorithm's behaviour. The chapter begins by explaining what is meant by an unconstrained optimization problem in section 2.1. Next, section 2.2 provides an introduction to particle swarm optimization and discusses several important implementation details.

2.1 Optimization

Optimization is the task of finding values for a set of variables so that some measure of optimality is satisfied, where the values are subject to a set of constraints. A tuple of candidate variable values is referred to as a *solution*. The set of all possible solutions that are within the specified constraints is referred to as the *feasible search space*. The "measure of optimality" is defined in terms of one or more *objective function(s)*. If the quality of a solution depends on only one objective function, the optimization problem is referred to as a *single-objective* problem. Otherwise, the problem is a *multi-objective* problem. The search space may be *discrete*, *continuous* or a mixture of both. Optimization problems may be *constrained* or *unconstrained*. Unconstrained probles impose no constraints on the set of allowed solutions. Constrained problems require that the solutions be subject to additional constraints.

Constraints are typically imposed due to the intrinsic properties of the variables being

optimized. Constraints may also be chosen using prior knowledge of the search space, so that the search is restricted to an area in which the solution is known or expected to exist. Boundary constrained problems only require that solutions adhere to boundary constraints which define maximum and minimum values for each variable.

An optimization problem may require that the objective function be either minimized or maximized. This thesis mainly considers minimization problems. Since the task of maximizing a function f is equivalent to minimizing the function $-f$, there is no loss of generality by considering only minimization problems.

This thesis considers boundary constrained, single-objective, minimization problems in continuous search spaces. Formally, such an optimization problem with objective function $f : \mathbb{R}^n \to \mathbb{R}$ is the task of finding $\mathbf{x}$ in $\mathbb{R}^n$ such that

$$\min f(\mathbf{x}). \tag{2.1}$$

The boundary constraints in the j_{th} dimension are expressed as

$$L_j \leq x_j \leq U_j. \tag{2.2}$$

If the boundary constraints are the same in each dimension so that

$$L_1 = L_2 = ... = L_n = L \quad \text{and} \quad U_1 = U_2 = ... = U_n = U \tag{2.3}$$

then the search space is restricted to $(L, U)^n$, where n denotes the dimensionality of the optimization problem.

2.2 Particle Swarm Optimization

This section provides a brief introduction to the particle swarm optimization (PSO) algorithm. Subsection 2.2.1 introduces the position and velocity update equations as originally conceptualized by Kennedy and Eberhart. The remaining subsections discuss the parameters and implementation choices that should be considered when implementing a PSO. More specifically, subsection 2.2.3 discusses PSO with an inertia weight coefficient, which is the algorithm under discussion for the majority of this thesis. Subsection 2.2.4 examines the acceleration coefficients and their relationship with convergence of the

swarm in terms of the expectation and variance of particle positions. Subsection 2.2.5 describes a few common choices for neighbourhood topologies. Subsection 2.2.6 mentions some interesting applications of the PSO algorithm and subsection 2.3 concludes the chapter.

2.2.1 Position and Velocity Update Equations

PSO was introduced by Kennedy and Eberhart in 1995 [20, 40]. It is a population-based, stochastic search algorithm that was inspired by the flocking behaviour of birds.

A swarm of particles are initialized within the search space. Each particle's position represents a possible solution to the optimization problem. The particles are "flown" through the search space for a number of iterations, using local and global information to guide their movement. The quality of a particle's position is evaluated by the objective function and is referred to as the particle's *score* or *fitness* throughout the thesis.

In every iteration, a particle's position is updated according to the position update equation as given below:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{2.4}$$

where $\mathbf{x}_i^{t+1}$ denotes the position and $\mathbf{v}_i^{t+1}$ denotes the velocity of the i_{th} particle at iteration $t+1$. Note that vectors are denoted in bold, as above.

Usually, the particles' initial position are sampled from a uniform distribution in each dimension so that $x_{i,j}^0 \sim U(L, U)$. Other means of initialization have been proposed in order to maximize the initial spread of the swarm. Schoeman and Engelbrecht suggested using a quasi-random number generator such as a Sobol sequence [65]. Richards and Ventura proposed making use of Voronoi tessellations to partition the search space into small cells of approximately the same size and using the centre of each cell as an initial particle position [60]. Parsopoulos and Vrahitis [55] suggested a method that uses the non-linear simplex method [50] to find initial positions with a good score. A full discussion of PSO initialization strategies is postponed to Chapter 4.

The optimization process is driven by the velocity update equation, which determines the direction in which the particles will fly. The velocity update equation as originally

suggested by Eberhart and Kennedy [20] is given by

$$v_{i,j}^{t+1} = v_{i,j}^{t} + c_1 r_{1,j}(y_{i,j}^{t} - x_{i,j}^{t}) + c_2 r_{2,j}(\hat{y}_{i,j}^{t} - x_{i,j}^{t}). \tag{2.5}$$

for particle i in the j_{th} dimension, at iteration $t+1$. The second and third terms are called the cognitive and social components, respectively. These two terms bring about the basic behaviour of a particle, which is to emulate the success of its neighbours as well as its own previous successes.

The cognitive term $(c_1 r_{1,j}(y_{i,j}^{t} - x_{i,j}^{t}))$ guides the particle towards the best position that it has encountered thus far. The best position encountered by particle i at iteration t is referred to as its *personal best position* and is denoted by $\mathbf{y}_i^t$.

The social term $(c_2 r_{2,j}(\hat{y}_{i,j}^{t} - x_{i,j}^{t}))$ guides the particle towards the best position that has been encountered by any of the particles in its neighbourhood. The best position in the neighbourhood of particle i at iteration t is denoted by $\hat{\mathbf{y}}_i^t$ and is referred to as the *global best position* or the *local best position*, depending on how the particle's neighbourhood is defined. See section 2.2.5 for further discussions about particle neighbourhoods.

The cognitive and social components are weighted by the acceleration coefficients, denoted by c_1 and c_2 which govern the influence of the personal best and global best positions respectively. Every dimension of the cognitive and social components are also weighted by uniform random variables, $r_{1,j}$ and $r_{2,j}$, in the range $[0, 1]$. This introduces stochasticity to the search, thereby allowing the swarm to explore the search space in the areas between its current position and its attractors.

The first term in equation (2.5) is called the momentum component and allows the particle to maintain some of its search direction from one iteration to the next. This may prevent a particle's velocity from oscillating rapidly [23], depending on the values of c_1 and c_2. It also prevents the swarm from rapidly contracting to the global best position, thereby facilitating exploration of the search space [69].

It was originally suggested that particle velocities be initialized uniform randomly throughout the search space, but it has been found that initializing particle velocities to zero may improve convergence time and prevent particles from leaving the search space [26]. Personal best positions may be set equal to the initial positions. Alternatively, personal best positions may also be initialized randomly within the search space. The value of the initial position and initial personal best position should be swapped if

necessary, to ensure that the fitness of the personal best position is better than the particle's initial position.

2.2.2 Velocity Clamping

Eberhart and Kennedy proposed *clamping* the velocity of the particles [20], so that the particle velocities can not be higher than some maximum value. Velocity clamping is applied to each velocity vector after being calculated, but before being used in the position update equation. There are a number of possible ways in which velocity clamping can be implemented. The simplest is to choose a maximum value for the velocity in a given dimension. Let $v_{max,j}$ denote the maximum velocity in dimension j. Then the velocity vector is updated in each dimension j according to

$$v_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1} & \text{if } -v_{max,j} \leq v_{i,j}^{t+1} \leq v_{max,j} \\ v_{max,j} & \text{if } v_{max,j} < v_{i,j}^{t+1} \\ -v_{max,j} & \text{if } v_{i,j}^{t+1} < -v_{max,j} \end{cases} \tag{2.6}$$

2.2.3 PSO with Inertia Weight Coefficient

A number of modifications have been made to the original velocity equation proposed by Kennedy and Eberhart in [20]. This study uses PSO with an inertia weight coefficient as proposed by Eberhart and Shi [69] in 1998.

The social and cognitive components cause a particle to emulate good solutions that have already been encountered, thereby encouraging exploitation. The momentum component causes each particle to maintain its current trajectory. This forces the particle to move through areas of the search space that may not be directly on course to previously observed successes, thereby encouraging exploration of the search space [69]. There must be a balance between exploration and exploitation: a swarm that only explores and never refines good solutions will waste resources exploring fruitless areas of the search space. A swarm that only exploits is likely to converge prematurely to a local optimum. Introducing the inertia weight coefficient to the momentum component provides more control over this exploration-exploitation balance.

The velocity update equation for particle i at iteration t in dimension j is thus given

by

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 r_{1,j}(y_{i,j}^{t} - x_{i,j}^{t}) + c_2 r_{2,j}(\hat{y}_{i,j}^{t} - x_{i,j}^{t}) \tag{2.7}$$

where $w \in (0, 1)$ is called the inertia weight.

Since $w < 1$ decreases the magnitude of the velocity vector, it may seem that using an inertia weight coefficient may remove the need for velocity clamping. Unfortunately, it was found that the use of an inertia weight coefficient alone is not enough to avoid velocity clamping [75, 77, 79]. Instead, attention must be paid to the choice of values for the acceleration coefficients which is the topic of the next subsection.

The standard global best or *global best (gBest) PSO* algorithm for a swarm with n_s particles is summarized in Algorithm 2.1. The algorithm is for an n-dimensional search space with boundaries (L, U) in all dimensions. The algorithm runs for t_{max} iterations. In a *gBest PSO*, the entire swarm forms a single neighbourhood. Thus all the particles have the same neighbourhood best which may be denoted by $\hat{\mathbf{y}}^t$ at iteration t.

2.2.4 Social and Cognitive Acceleration Coefficients

Typically, the values for c_1 and c_2 are chosen in the range $[0, 2]$ and that of w is in $(0, 1)$. Although some regard PSO to be fairly robust with regard to selection of these parameters [20], ill-chosen values may lead to premature convergence, excessive wandering or cause the swarm to exhibit divergent or cyclic behaviour [56, 77, 79]. In general, these parameters should be tuned for each optimization problem.

Eberhart and Shi proposed "rule of thumb" parameter values for which the algorithm performs well [22]. The paper also provided guidelines for choosing parameters that will ensure convergence based on the work of Clerc and Kennedy on constriction models [15]. The chosen values were $c_1 = c_2 = 1.49445$ and $w = 0.7929$.

Further theoretical research has shown that there is a "convergent region" within which swarm convergence is guaranteed [75, 77, 79]. For parameters within the convergent region, it is guaranteed that the expectation and variance of particle positions will converge to constant values. The most accurate convergent region is that derived by Poli [56], given below:

$$0 < c_1 + c_2 < 24\frac{(1 - w^2)}{7 - 5w} \tag{2.8}$$

$$|w| < 1 \tag{2.9}$$

The region above was derived and verified empirically assuming *search stagnation* during which all particles behave independently and no improved solutions are found. The convergent region given above was empirically verified without assumptions by Cleghorn and Engelbrecht [13].

There are also a number of proposed strategies to vary the coefficients w, c_1 and c_2 according to the iteration or other information about the search progress [31, 80]. These schemes include a time-varying, linearly decreasing inertia weight component proposed by Shi and Eberhart [71] and a scheme proposed by Ratnaweera *et. al.* that also varies the acceleration coefficients linearly with time [59]. In general, these strategies are not efficient [31, 80].

2.2.5 Social Structures

A particle's neighbourhood topology or social structure describes the flow of information among the particles. The topology can be expressed in terms of a graph where particles are represented by vertices and the ability to share information is represented by edges. In a star topology, illustrated in figure 2.1, every particle is informed about the successes of every other particle in the swarm. In this case, the particle's neighbourhood is the entire swarm. Particles imitate the overall best solution, which has been shown to cause faster convergence than other neighbourhood topologies [42] due to faster knowledge transfer (though in general, the speed of convergence to an optimum remains problem dependent [27]). When using the star topology, the social term's attractor is referred to as the *global best position*. The algorithm is referred to as a *gBest PSO*.

A ring topology, as illustrated in 2.2 defines a smaller neighbourhood for each particle. This restricts the flow of information among the particles. A particle can only communicate with its k immediate neighbours. In the case where $k = 2$, the particle can only communicate with the particles that are immediately adjacent to it in terms of particle indices.

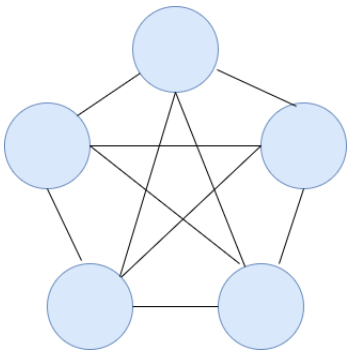


Figure 2.1: Star neighbourhood topology, social structure of a *gBest* PSO

In general, the neighbourhood of a particle is not determined by its position in the search space. Instead, the neighbourhood is defined based on particle indices. This promotes the flow of information. Otherwise, particles in poor regions of the search space would only be guided by other particles in that region, perhaps preventing the particles from leaving to find a better region. Spatial ordering, such as the method proposed by Suganthan [73], is also computationally expensive. For non-spatial topologies, neighbourhoods can overlap. This allows information to be communicated across different neighbourhoods. Nevertheless, the flow of information is restricted, which causes the swarm to lose diversity more slowly than a *gBest PSO*. This encourages exploration of the search space and may provide higher quality solutions in a multimodal search space than a *gBest PSO*. When using the ring topology, the social term's attractor is referred to as the *local best position*. The algorithm is referred to as *local best (lBest) PSO*.

Other neighbourhood topologies have been suggested such the wheel, pyramid, four clusters and Von Neumann topologies [41]. However, this study uses the star and ring topologies for reasons that will be described in Chapter 4.

2.2.6 Applications

PSO has been applied to numerous problems in industry including medicine, reactive power and voltage control, scheduling, antennae design, photovoltaic systems and assembly line balancing problems. In 1999, Eberhart and Hu used PSO to train a neural network that can distinguish between healthy subjects and subjects that are affected by

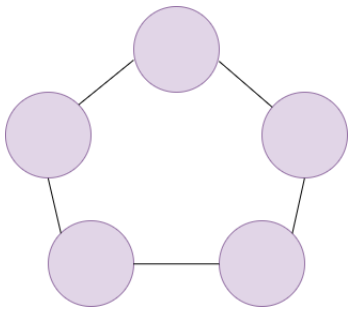


Figure 2.2: Ring neighbourhood topology, social structure of a *lBest* PSO with $k = 2$

Parkinsons or an essential tremor [21]. In 2015, Li *et. al.* [43] proposed a PSO variant that optimized parameters for Otsu image segmantation which is used to process medical images. Costa *et. al.* successfully applied PSO to the problem of reactive power compensation and voltage level control [16]. Pandey *et. al.* developed a heuristic to schedule applications in the cloud, considering both the execution and data transmission costs using PSO [53]. Zhang *et. al.* successfully applied PSO to solve resource-constrained project scheduling problems [84]. A modified PSO was developed by Faria *et. al.* to optimize the management of energy resources in a distribution network [29]. Bataineh made use of PSO to design Chebyshev arrays [2]. Robinson and Rahmat-Samii used PSO to develop a profiled corrugated horn antenna [63]. Ishqaue and Salam applied a deterministic version of PSO to track the maximum power point of photovoltaic arrays under partial shading [35]. PSO has also been used to determine the optimal size of a solar photovoltaic system [76]. In 2017, Delice *et. al.* developed a modified PSO with negative knowledge and applied it to solve the mixed-model, two-sided assembly line balancing problem [18].

In principle, PSO can be applied to any problem that can be expressed in terms of an objective function to be optimized. Since the algorithm does not use gradient information explicitly, PSO can be applied to situations where no gradient information is available or where the objective function is not differentiable. Particularly, PSO can be applied to solve black-box optimization problems.

2.3 Summary

Section 2.1 defined what is meant by an "optimization problem." Section 2.2 introduced the Standard PSO algorithm and explained relevant concepts such as exploration and exploitation. Section 2.2 also discussed the components of the velocity update equation in detail and provided a brief overview of different social structures that may be chosen for information exchange within the swarm.

Chapter 3 begins to examine high dimensional search spaces and the complications that arise in such search spaces. The chapter discusses the curse of dimensionality and provides an overview of existing literature related to PSO in high-dimensional search spaces.

//Initialize a swarm of n_s particles
for all *particles* $i = 1, ..., n_s$ **do**
for all *dimensions* $j = 1, ..., n$ **do**
$x_{i,j}^0 \sim U(L, U)$
$y_{i,j}^0 \sim U(L, U)$
$v_{i,j}^0 = 0$
end for
//Swap personal best and current position if necessary
if $f(\mathbf{x}_i^0) < f(\mathbf{y}_i^0)$
tmp $= \mathbf{y}_i^0$ $\mathbf{y}_i^0 = \mathbf{x}_i^0$ $\mathbf{x}_i^0 =$ tmp
endif
end for
//Search iteratively
for all $t = 1, ..., t_{max}$
for all *particles* $i = 1, ..., n_s$ **do**
if $f(\mathbf{x}_i^t) < f(\mathbf{y}_i^t)$
$\mathbf{y}_i^t = \mathbf{x}_i^t$
endif
if $f(\mathbf{y}_i^t) < f(\hat{\mathbf{y}}^t)$
$\hat{\mathbf{y}}^t = \mathbf{y}_i^t$
endif
end for
for all *particles* $i = 1, ..., n_s$ **do**
Update $\mathbf{v}_i^{t+1}$ using Equation (2.7)
Update $\mathbf{x}_i^{t+1}$ using Equation (2.4)
end for
end for

Algorithm 2.1: Standard algorithm for a gBest PSO

Chapter 3

Symptoms of High Dimensional Problems

This chapter launches the discussion of PSO applied specifically to high dimensional problems, meaning problems with 100 or more dimensions. The aim of this chapter is to illustrate the problems exhibited by PSO on high dimensional problems. Armed with empirical evidence, the chapter identifies "symptoms" that occur in high dimensions such as velocity explosion and continuous particle roaming so that the remainder of the thesis may attempt to unearth their causes and mitigate their effects.

Section 3.1 introduces the "curse of dimensionality" and discusses some of the phenomena that arise in high dimensional spaces. Section 3.2 contains an initial empirical study showcasing the troublesome behaviour of PSO in high dimensions. Section 3.3 considers particle roaming behaviour and how it is amplified in high dimensional search spaces. Lastly, section 3.4 concludes the chapter with a summary and reasoning for the chapters that follow.

3.1 The Curse of Dimensionality

The "curse of dimensionality" is a term coined by Bellman in his book on dynamic programming in 1957 [3]. The phrase refers to the unintuitive phenomena that arise in high dimensional spaces, but are typically not observed in low dimensional spaces.

High dimensional phenomena generally arise from the exponential increase in hyper-volume of the corresponding problem space. In combinatorics, for example, an increase in problem dimensionality leads to an exponentially larger number of possible states to be considered. The curse of dimensionality is problematic in fields such as modelling and optimization [68], machine learning [19], and data mining [72, 83, 85] where information is gleaned from samples in the problem space. The exponential increase in the search space's hyper-volume causes data points to be sparse, making it difficult to perform accurate analysis. The number of samples required to learn statistically significant patterns grows exponentially with the search space. In the case of machine learning, the number of training samples required to accurately characterize the input space increases exponentially [19]. Due to the sheer size of the search space, algorithms and techniques that require systematic sampling or exploration of the search space often become infeasible.

In data mining, the enormity of the search space may lead to *data snooping* bias [85]. Data snooping or *p-hacking* refers to the practice of searching for patterns in a data set without a particular hypothesis. The process typically involves testing many different hypotheses about a single data set until one happens to be statistically significant. When there are a large number of possible associations between variables, but only a few real associations exist, the majority of findings may be false positives [72]. In high dimensional spaces, the number of possible associations to be considered increases exponentially, which may lead to p-hacking.

In the field of modelling the computational demand of generating models and running simulations increases exponentially with problem dimensionality [68]. Evaluation of a high-dimensional objective function is computationally expensive, making numerical optimization difficult [9]. Optimization methods that rely on sampling points from the solution space fail to achieve good coverage of the search space due to the exponential increase in search space hyper-volume and the cost of evaluating a sample [9].

Another problem in high dimensions is that the notion of proximity may become ill-defined. As shown by [4], for certain sampling distributions and distance metrics, all points in the sample approach the same distance apart as the problem dimensionality goes to infinity. Let $d(,)$ denote a distance metric and let S denote the sample set. For

a given query point, $\mathbf{x}$, the distance ratio below:

$$\frac{\max_{\mathbf{y}\in S}\{d(\mathbf{x},\mathbf{y})\}}{\min_{\mathbf{z}\in S}\{d(\mathbf{x},\mathbf{z})\}} \rightarrow_p 1 \tag{3.1}$$

converges in probability to 1 as dimensionality increases (where $\rightarrow_p$ denotes convergence in probability). Particularly, this holds for the L_p norm when the sample is drawn from independent distributions with finite variance that are identical across dimensions. Thus, equation (3.1) will converge when the Euclidean norm is applied to points drawn from a uniform random distribution. The rate of convergence under these conditions were clearly illustrated by Morgan and Gallagher [49]. For dimensionality as low as 20, the maximum distance between the query point and any point in the sample, is less than a single order of magnitude larger than the smallest distance. Algorithms such as k-nearest neighbour [30] that rely on distance metrics such as the Euclidean norm to organize data are thus prone to exhibiting poor performance in high dimensional spaces [58].

3.2 Empirical Illustration

This section contains experiments showcasing the difficulties encountered in high dimensions with regard to optimization problems. The section begins by showing the behaviour exhibited by PSO on typical high dimensional problems in comparison with its behaviour in lower dimensions in section 3.2.1. Section 3.2.2 discusses the use of swarm size adjustments to compensate for dimensionality increase and recognizes the problem of particle roaming.

3.2.1 A Naive Approach

This section performs a somewhat naive comparison of PSO behaviour between high dimensional problems and low dimensional problems for illustrative purposes. Specifically, PSO is used to solve a suite of 10-dimensional problems and a suite of 1000-dimensional problems. The difference in behaviour of the swarms provides a basis for the other empirical illustrations that are performed in section 3.2.2 and the following chapters.

The parameter values chosen are the "rule of thumb" values or the values determined from theoretical results in the field. The experiment thus illustrates what might be encountered when applying PSO to high dimensional problems without any special modifications or forethought, i.e. what a naive practitioner will observe.

The benchmark functions from the CEC'2010 special session and competition on large-scale optimization were used [74]. The benchmark suite allows the degree of separability within certain functions to be specified using the parameter m. In order to scale the problems down to 10 dimensions, the experiment used a value of $m = 10$. Appendix A provides a detailed discussion of the benchmark functions and the parameter m. The search spaces for all the benchmark problems had the same upper and lower bounds in every dimension, denoted by U and L respectively. These values corresponded to the specifications in [74]. The benchmark suite includes separable functions, partially separable functions, and non-separable functions. The swarm was tested on each of the benchmark functions by running 30 independent simulations to achieve statistical significance. The performance of an algorithm on a given benchmark function was characterized in terms of the best fitness value attained in a given simulation. For all of the benchmark functions, the best possible fitness value is 0.

The experiment was performed using a swarm size of 30. No velocity clamping was applied (the effects of velocity clamping are discussed at length in Chapter 4). Swarm updates were performed synchronously. The swarm was allowed to run for 5000 iterations, which provides a total of 30×5000 function evaluations. The values for the cognitive and social acceleration components as well as the inertia weight were chosen as suggested by [15] and are given in table 3.1 along with the other pertinent parameter values.

Initial particle velocities were set to zero [26]. Particle positions were initialized by sampling from a uniform random distribution in every dimension, so that $x_{i,j}^0 \sim U(L, U)$ as described in section 2.2.1. Particles' personal best positions were initialized in the same manner, then both the current position and personal best position were evaluated. If the current position had a better score than the personal best position, the two were swapped.

To ensure that the solution found by the swarm is within the search space, the global

Table 3.1: Parameter Settings

Parameter	Value
n_s	30
Velocity Clamping	No
Function Evaluations	30×5000
c_1	1.49618
c_2	1.49618
w	0.7298

best and personal best positions were only updated if they were within the search space. Thus, both particle attractors were always within the search space.

The experimental setup for all the other experiments in this chapter are similar to what is described here. For the sake of brevity, the sections that follow only describe deviations from this setup.

The mean and standard deviation of the best fitness achieved by the swarms are given in Table 3.2 and Table 3.3. The functions are labeled with numbers that match the source [74].

For the suite of 10-dimensional problems, PSO successfully found a solution within one order of magnitude of the optimal value for three-quarters of the benchmark functions as seen in Table 3.2. The poor performance on functions $F4$ to $F8$ can be understood by looking at their general form:

$$F(\mathbf{x}) = F_\gamma\big(\mathbf{z}(P_1 : P_m)\big) \times 10^6 + F_\alpha\big(\mathbf{z}(P_{m+1} : P_n)\big) \tag{3.2}$$

where F_γ and F_α are two "basic" functions, such as the elliptic function or the spherical function and $\mathbf{z}$ is the shifted candidate solution. P is a random permutation of $\{1, 2, 3, ..., n\}$ where n denotes problem dimensionality and P_j is the j-th component of P. The notation $\mathbf{z}(P_1 : P_m)$ denotes a vector formed by the components of $\mathbf{z}$ that correspond to the indices P_1, P_2, ..., P_m. These benchmark functions are thus the sum of two basic functions, where the parts of the candidate solution that are given as input to each basic function is randomly determined and shuffled (once for all the runs). More information regarding the benchmark functions is provided in appendix A.

Table 3.2: Mean and standard deviation of best fitness for PSO (n = 10)

Function	Mean	Standard Deviation
F1	$0.0000e+00$	$0.0000e+00$
F2	$7.5617e+00$	$1.3097e+00$
F3	$2.3685e-16$	$2.4516e-16$
F4	$1.4429e+11$	$1.8038e+10$
F5	$1.9302e+07$	$1.5302e+06$
F6	$4.0625e+05$	$1.8883e+05$
F7	$0.0000e+00$	$0.0000e+00$
F8	$5.3594e+05$	$3.7449e+05$
F9	$0.0000e+00$	$0.0000e+00$
F10	$7.9597e-01$	$2.5026e-01$
F11	$0.0000e+00$	$0.0000e+00$
F12	$0.0000e+00$	$0.0000e+00$
F13	$0.0000e+00$	$0.0000e+00$
F14	$2.1945e+04$	$4.0079e+03$
F15	$2.6134e+01$	$2.9051e+00$
F16	$1.0038e+00$	$2.5738e-01$
F17	$0.0000e+00$	$0.0000e+00$
F18	$1.3458e+00$	$5.1694e-01$
F19	$0.0000e+00$	$0.0000e+00$
F20	$2.6762e-01$	$2.7497e-01$

Since $m = 10$, equation (3.2) reduces to only the first term for the 10-dimensional problems (since $n = m$, the entire candidate solution is given as input to F_γ). The scaling by 10^6 of the first term causes quality of the solution to be large, even when close to the optimum. Function $F9$ and $F14$ contains a sum of rotated elliptic functions, which is also very large until the solution is very close to zero.

Table 3.3, which records the best fitness attained by PSO on the 1000-dimensional problems, paints a very different picture. PSO failed to attain a fitness value anywhere

Table 3.3: Mean and standard deviation of best fitness for PSO (n = 1000)

Function	Mean	Standard Deviation
F1	$4.3880e+11$	$3.8811e+09$
F2	$2.6099e+04$	$7.4363e+01$
F3	$2.1541e+01$	$1.6471e-03$
F4	$1.6898e+15$	$1.5412e+14$
F5	$1.8676e+08$	$5.1447e+06$
F6	$2.1156e+07$	$4.4054e+04$
F7	$9.0093e+11$	$5.8336e+10$
F8	$6.1919e+16$	$5.8365e+15$
F9	$6.6098e+11$	$5.8167e+09$
F10	$2.6108e+04$	$4.6733e+01$
F11	$1.0914e+03$	$3.5023e-01$
F12	$9.5776e+07$	$5.5507e+05$
F13	$4.3821e+12$	$3.5102e+10$
F14	$7.5268e+11$	$7.8549e+09$
F15	$2.6138e+04$	$5.5472e+01$
F16	$2.1396e+03$	$4.1803e-01$
F17	$1.8030e+08$	$1.1784e+06$
F18	$9.3999e+12$	$3.5413e+10$
F19	$1.9977e+12$	$1.2580e+10$
F20	$1.0441e+13$	$7.6299e+10$

near the optimal value for all the benchmark functions. Additionally, the standard deviation of the best fitness is very high for some of the functions such as $F14$ which has a standard deviation with an order of magnitude of 10^9. Such a high standard deviation indicates poor search quality and erratic, random swarm behaviour.

The experiment shows that problem complexity grows far beyond the ability of a simple, unmodified PSO as the dimensionality increases.

3.2.2 Swarm Size and the Search Space

The next step by the optimistic, naive practitioner may be to increase the size of the swarm. A larger number of particles may be able to traverse a greater portion of the search space because every additional particle provides the swarm with additional "sample points" or information regarding the location of good or bad solutions. From another perspective, every randomly initialized particle has a certain problem-dependent probability of being initialized in a fruitful region of the search space. Thus, every additional particle increases the swarm's chances of encountering good solutions and making it less likely for the swarm to become trapped in local minima.

Unfortunately, the size of the search space grows exponentially as the problem dimensionality is increased. In low dimensions, it is inexpensive to have at least as many particles as there are problem dimensions, which allows relatively good search space coverage. However, a constant value, such as $n_s = 50$ (as recommended by [8]) that achieves good search coverage in low dimensions, will degenerate to sparse coverage in higher dimensions. Even linearly increasing the swarm size with the problem dimensionality seems as though it will be insufficient, since the hyper-volume to be explored grows exponentially.

The ramifications of having fewer particles than problem dimensions is discussed in detail in [10] and in Chapter 6. This section explores the effect of increasing the swarm size, confirming the results of [10], and is useful for further discussion of the pathological behaviour exhibited by PSO in high dimensional search spaces.

The next experiment illustrates the effect of different swarm sizes on the behaviour of the swarm. The experiment described in section 3.2.1 was repeated for both 10 and 1000 dimensions, but for different swarm sizes. The chosen swarm sizes were: $5, 10, 30, 50, 100, 250$. Apart from the swarm size, all other parameters were the same as listed in table 3.1. Each swarm configuration was allowed to run for the same number of function evaluations, namely 30×5000, as before. A swarm of size n_s was thus allowed to run for $\frac{30 \times 5000}{n_s}$-many iterations. Although there may be some question regarding the fairness of using function evaluations as a stopping condition [25], practicality also had to be considered. Running very large swarms for many iterations requires a great deal of computation time, thus a stopping condition had to be chosen that penalized the very

Table 3.4: Rank test comparing swarm size 250 against other swarm sizes ($n = 1000$

	$>$ PSO ($n_s = 250$)	$=$	$<$ PSO ($n_s = 250$)
PSO ($n_s = 100$)	0	2	18
PSO ($n_s = 50$)	0	0	20
PSO ($n_s = 30$)	0	0	20
PSO ($n_s = 10$)	0	0	20
PSO ($n_s = 5$)	0	0	20

large swarms in some way. The allowed number of function evaluations was still very large and should have provided sufficiently many iterations for a swarm to obtain a good solution. Empirically, none of the swarms were still improving in best fitness or updating the global best position by the time that the search was halted, so the stopping condition was deemed fair.

Friedman tests with a p-value of 0.05 were used to detect statistically significant differences among the swarms' performance across varying swarm sizes. If the Friedman test indicated significant differences, further pairwise comparisons were done by means of Mann-Whitney U tests with a p-value of 0.05.

Table 3.4 provides the outcome of the rank tests where the score of the largest swarm ($n_s = 250$) was compared to the scores of all the other swarms. The table may be interpreted as follows: a cell in the first column provides the number of functions for which the swarm configuration listed in that row performed statistically significantly better than the swarm in the column headings. A cell in the last column contains the number of functions for which the situation was reversed, i.e. the swarm configuration in the column headings performed significantly better than the swarm listed in that row. The cells in the middle column contains the number of functions for which the swarms' scores were not significantly different.

At first glance, table 3.4 indicates that a larger swarm size is advantageous. The swarm with 250 particles performed significantly better than the smaller swarms in almost all cases. Upon inspection of figure 3.1, the results seem a bit suspect. Figure 3.1 follows a single run of the algorithm with $n_s = 30$ on F3 (shifted Ackley's function) in 1000 dimensions and shows the best objective function value of the particles' current

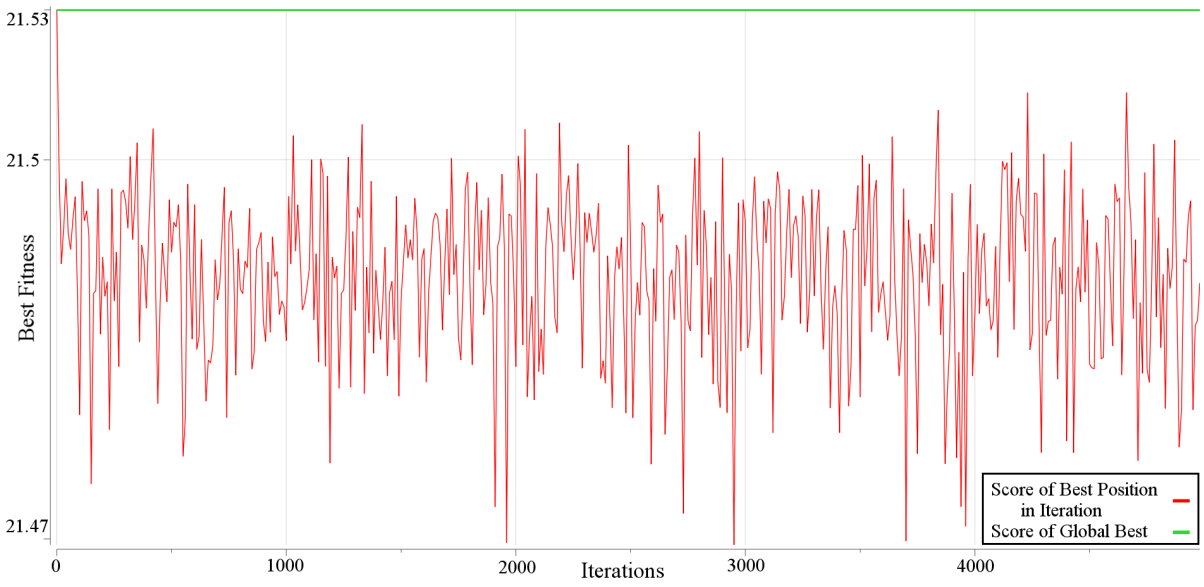


Figure 3.1: Score of best position and score of global best throughout search on F3 ($n_s = 30$, $n = 1000$)

positions for every iteration as well as the objective function value of the global best position. Disconcertingly, the score of the global best position never changes throughout the search. This observation held regardless of swarm size or benchmark problem.

The only situation in which particles may have better scores than the global best position is when those particles were not in bounds. Since the global best position was never updated, figure 3.1 implies that the particles immediately left the bounds of the search space within the first iteration and failed to return. Figure 3.2 confirms the hypothesis by showing the percentage of the swarm that was out of bounds throughout the search. The values plotted in figure 3.2 were averaged over all 30 runs of the algorithm with $n_s = 30$. Similar plots were obtained for all the other swarm sizes on all of the benchmark functions.

Since the parameter values are convergent (i.e. the expectation and variance of particle positions will converge to constant values) according to [56] and [12], the problem is not that the experiments were performed with divergent parameters which caused the exacerbated roaming behaviour. Further discussion of swarm convergence and particle roaming behaviour is postponed to Chapter 5.

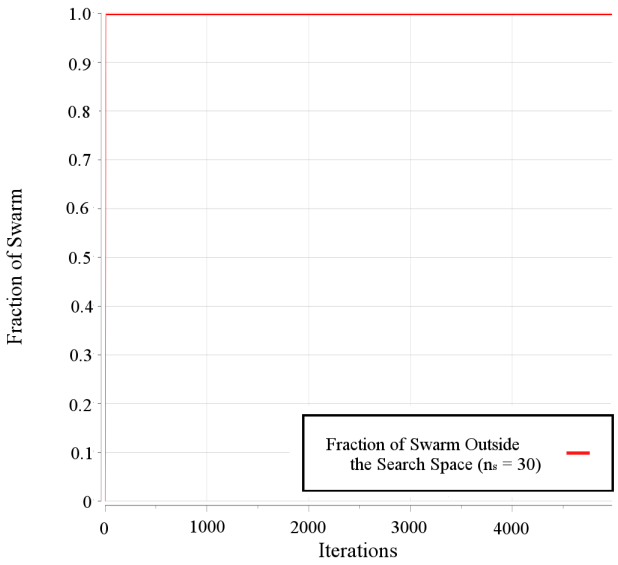


Figure 3.2: Fraction of swarm outside the search space on F3 ($n_s = 30$, $n = 1000$)

Both the local and global attractors were confined to the search space, so it seems intuitive for the swarm to be pulled back into the search space. Indeed, [24] postulated that given sufficiently many iterations, particles will eventually return to the search space. Unfortunately, as shown in this section, such intuitions do not hold in high dimensional spaces. The section that follows will relate these observations regarding particle roaming to existing literature.

3.3 Particle Roaming

Particle roaming behaviour is documented in existing literature [24, 34]. Empirically, even in low dimensional spaces, particle roaming can be observed. Particle roaming refers to a phenomenon that usually occurs in the first few iterations of the search where particles are likely to leave the search space. Typical particle roaming behaviour in low dimensional spaces is captured in figures 3.3, 3.5 and 3.7. These will be placed in contrast with figures 3.4, 3.6 and 3.8, which illustrate particle roaming in high dimensional spaces. These figures are plotted for a typical run of the algorithms on F7. Similar plots were

obtained for the other benchmark functions.

The most conspicuous aspect to examine is how many particles are outside the search space, i.e. roaming. The fraction of the swarm that was outside of the search space was measured and is shown in figures 3.3 and 3.4. For the low dimensional case, the fraction of particles outside the search space increased sharply and then rapidly decreased as the particles were pulled back towards the attractors (figure 3.3). For the high dimensional problem, the particles also left the search space at once and started roaming. However, the roaming particles failed to return to the search space (figure 3.4).

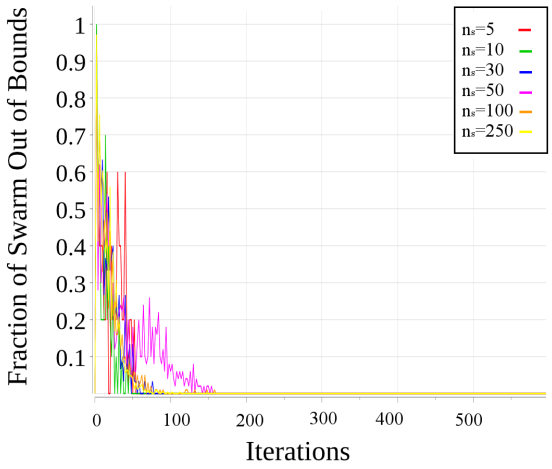


Figure 3.3: Fraction of swarm outside search space on F7 ($n = 10$)

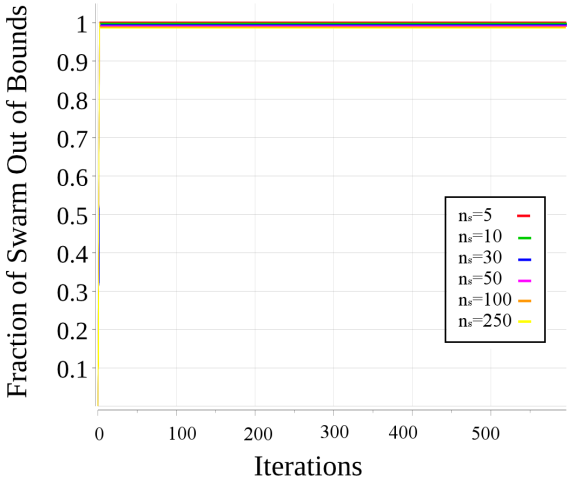


Figure 3.4: Fraction of swarm outside search space on F7 ($n = 1000$)

Other artefacts related to the roaming behaviour can be observed in the swarm diversity and the average particle velocity. Swarm diversity characterizes the spread of the swarm and may be used to illustrate exploration and exploitation behaviour. Different measures for swarm diversity have been suggested such as the swarm diameter and radius [61], the average distance around the swarm centre [42], and the normalized average distance around the swarm centre [61]. Engelbrecht and Olorunda [51] found that the average distance from the swarm centre [42] provides a good compromise between accuracy, robustness and computational efficiency, so this thesis will use the average

distance from the swarm centre to measure diversity. The swarm diversity is given by

$$\mathcal{D} = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n} (x_{i,j} - \hat{x}_j)^2} \tag{3.3}$$

where $\hat{\mathbf{x}}$ denotes the swarm centre, given as

$$\hat{\mathbf{x}} = \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{x}_i \tag{3.4}$$

The average particle velocity is given by

$$\mathcal{V} = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n} v_{i,j}^2} \tag{3.5}$$

Particle roaming in low dimensional spaces usually takes place in the first few iterations, when swarm diversity is the highest. As illustrated in figure 3.5, the diversity started high, showed a small spike for some of the swarms, and then gradually decreased as the search progressed and the particles converged. Observe that all the swarms converged to a fixed point since the diversity decreased to 0, regardless of swarm size.

The high dimensional case is now considered. Note that, for the high dimensional problems, a larger range in diversity values is expected, since the metric does not correct for the number of dimensions, i.e. the maximum distance from the swarm centre will grow with dimensionality since every extra dimension adds a non-negative value to the sum inside the square root in equation (3.3). However, it would also be expected of a successful search that the diversity would decrease to a relatively small amount as the swarm eventually focused on exploiting a fruitful region.

As shown in figure 3.6, for $n = 1000$, the swarm diversity was initially much higher than for the low dimensional case - as expected, but then plummeted and settled to constant value within the first 50 iterations. In comparison, the diversity drop was much slower for $n = 10$, with the diversity still decreasing by iteration 100. Additionally, the value around which the diversity settled for the high dimensional case was between 1400 and 1500 which is unfortunately very far from 0. It can thus be concluded that none of the swarms converged to a fixed point for the high dimensional case. It is also interesting to note that the average distance from the swarm center was lower for the smaller swarm

sizes, but seemed to increase asymptotically towards the same value as the swarm size increases. It is clear that simply changing the swarm size will not rectify the swarm's behaviour.

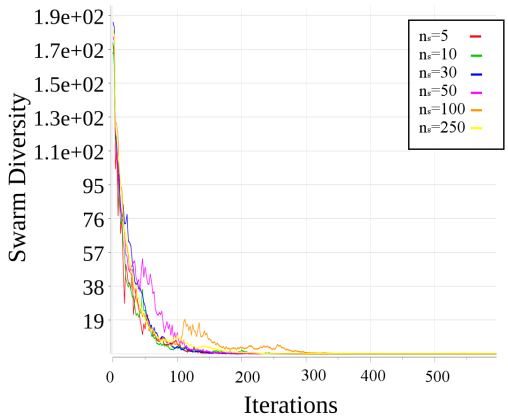


Figure 3.5: Swarm diversity on F7 for varying swarm sizes ($n = 10$)

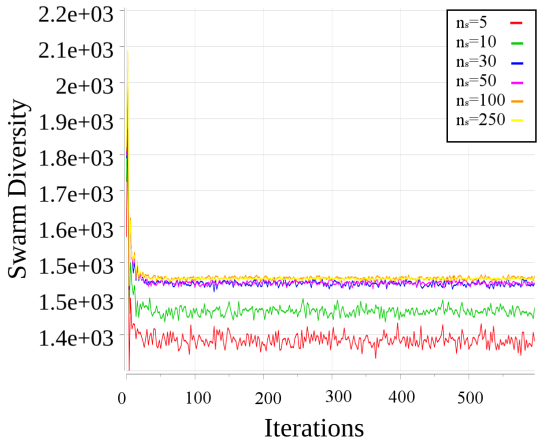


Figure 3.6: Swarm diversity on F7 for varying swarm sizes ($n = 1000$)

Figures 3.7 and 3.8 plot the average particle velocity throughout the search for $n = 10$ and $n = 1000$, respectively. For the low dimensional problem (figure 3.7), the average particle velocity spiked sharply from the initialized value of 0 as the particles began to explore and roam, then gradually decreased as the search progressed and the particles converged to the solution. The swarms on the high dimensional problems exhibited a similar initial spike and drop. Although the initial spike for $n = 1000$ was an order of magnitude larger than that of $n = 10$, this was not unexpected since the average particle velocity does not correct for the number of problem dimensions.

For $n = 1000$ (figure 3.8), the average particle velocity reduced rapidly and then settled to a large value within the first 50 iterations where it remained for the rest of the search. In comparison, for the $n = 10$ problem, the swarms were still decreasing in average particle velocity by iteration 100.

Generally, the average particle velocity behaves similarly to the average swarm diversity in that both metrics exhibit large initial spikes for both problem dimensionalities. Both metrics reduce to zero for the low dimensional problem, but stay at a large constant

value for the high dimensional problem.

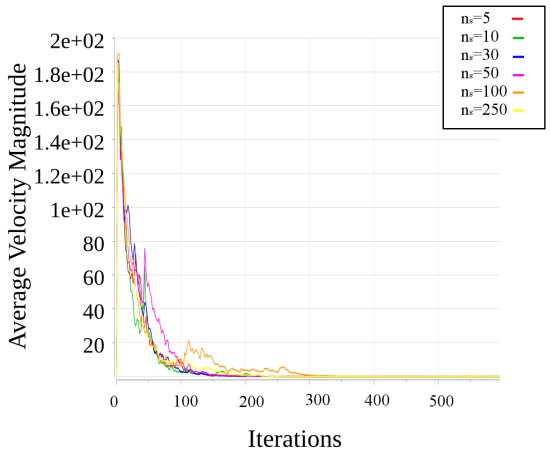


Figure 3.7: Average particle velocity on F7 for varying swarm sizes ($n = 10$)

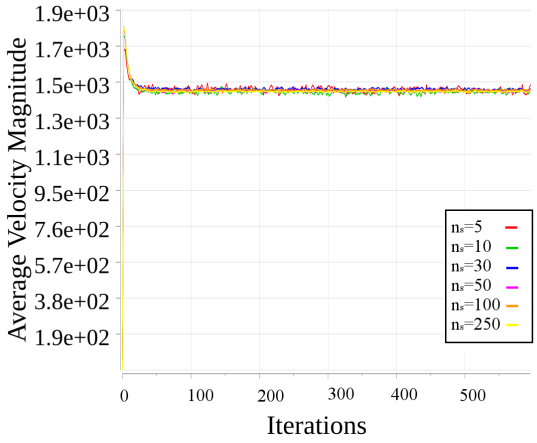


Figure 3.8: Average particle velocity on F7 for varying swarm sizes ($n = 1000$)

In lower dimensional problems, the roaming particles return to the search space and the swarm diversity and average particle velocity decrease. Results from [24] suggest that the early roaming behaviour may even be beneficial to the search if the attractors are appropriately constrained. But as illustrated by the results in this chapter, in high dimensional search spaces, the roaming behaviour continues indefinitely.

The relationship between particle roaming and problem dimensionality has been proved theoretically in literature. Helwig and Wanka [34] related problem dimensionality to the probability of particles leaving the search space. Before stating the theorems, a definition of the term *with overwhelming probability* is provided below:

Definition 3.1. An event $\mathrm{A}(n)$ occurs *with overwhelming probability* with respect to n if there exists a constant $\gamma > 0$ so that

$$\mathrm{P}(\mathrm{A}(n)) = 1 - e^{-\Omega(n^{\gamma})} \tag{3.6}$$

where Ω refers to the *big-O* notation for expressing asymptotic behaviour.

There are two theorems of interest, the first of which states that particles will leave the search space *with overwhelming probability* within the first iteration. The theorem

is stated formally below [34]

Theorem 3.1. *Assume the following initial conditions for each particle i and dimension j:*

1. *The initial velocity $v_{i,j}^0$ is sampled from a uniform random distribution in $[-r, r]$*
2. *The initial position $x_{i,j}^0$ is also sampled from $U[-r, r]$*
3. *The personal best position is equal to the initial position, so that $y_{i,j}^0 = x_{i,j}^0$*

Then every particle will leave the search space $[-r, r]^n$ with overwhelming probability.

In other words, the probability that a particle will leave the search space rapidly approaches 1 as n is increased. The result was proved for any neighbourhood topology and for additional velocity initialization strategies (namely initialization to zero and initialization to half the difference between the upper and lower search space bounds), though the theorem above is stated in terms of uniform random initialization.

The other theorem states that there is a fixed probability for a particle to leave the search space in a given dimension.

Theorem 3.2. *Assume that conditions 1 to 3 from Theorem 1 hold. Then each particle that is a local attractor ($\mathbf{x}_i^0 = \hat{\mathbf{y}}^0$) leaves the search space in, on average, $\frac{w}{4}n$ dimensions within the first iteration.*

Although Helwig and Wanka proved the result for a swarm in which the personal best position is set to the particle's initial position, the empirical results in figures 3.9 and 3.10 show similar results. Figure 3.10 shows that the average number of dimensions that were out of bounds per iteration was approximately 5.7%. This held across all benchmark functions and for all swarm sizes. In contrast, on the low dimensional problem ($n = 10$), the particles were never out of bounds on more than one dimension.

Particle roaming is thus unavoidable with a simple PSO and becomes exacerbated in high dimensional spaces.

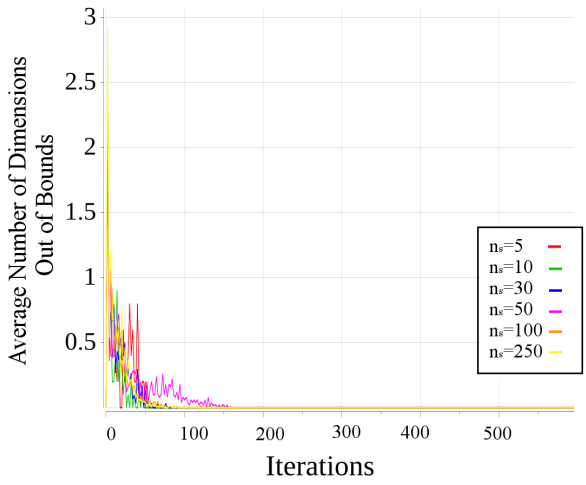


Figure 3.9: Average number of dimensions out of bounds on F7 for varying swarm sizes ($n = 10$)

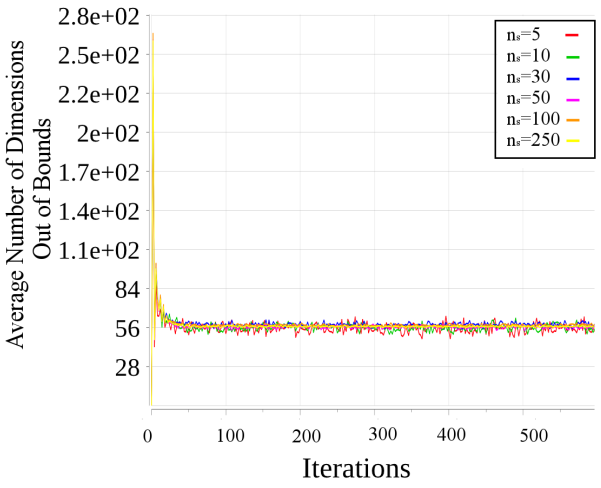


Figure 3.10: Average number of dimensions out of bounds on F7 for varying swarm sizes ($n = 1000$)

3.4 Summary

This chapter introduced the most troublesome behaviour of PSO in high dimensional problem spaces, namely that the particles leave the search space almost immediately and fail to return. The roaming behaviour is exacerbated as the dimensionality of the problem increases. Empirical experiments were leveraged to illustrate the effects of high dimensional search spaces on the swarm's behaviour, particularly on the swarm's velocity explosion and roaming behaviour. A number of metrics were introduced such as swarm diversity and average particle velocity which may be used to illustrate particle roaming behaviour and the extent to which it occurs.

Unfortunately, it is no easy task to prevent particles from leaving the search space or to coax them back into the search space once they are out of bounds. Further avenues of study that are pursued in this thesis are listed below:

1. Apply velocity clamping to limit the particles' maximum step size to prevent the particles from leaving the search space by curbing the initial velocity explosion.

2. Examine the inertia weight and acceleration coefficients of the swarm in pursuit of

parameter values that may force the swarm's variance to be small enough that the particles stay within the search space.

3. Use different initialization strategies that may curb particle roaming.

4. Limit the effect of the stochastic scaling components, which may discourage particle roaming.

In the following chapters, each of these items are discussed in detail. The next chapter critically examines the use of velocity clamping to prevent particle from leaving the search space.

Chapter 4

Velocity Clamping

The previous chapter observed that for high dimensional problems, particle velocities may become very large and particles are prone to leaving the search space, never to return. This chapter examines the extent to which velocity clamping may help in preventing the initial velocity explosion.

It has been claimed [20] that velocity clamping can be employed to prevent particles from leaving the search space by preventing the initial explosion of particle velocities. However, velocity clamping does not modify particle positions and thus does not directly prevent particles from roaming. Limiting the magnitude of a particle's velocity by an upper bound will limit the particle's step size. The particle will thus have to perform more steps to traverse the same distance in the search space. The step size determines the granularity of the search and thus how much the swarm will focus on exploring the entire search space or exploiting locally.

Preventing a particle from taking very large steps may prevent the initial velocity explosion, but the strategy introduces a problem dependent parameter (namely, the maximum allowed velocity). Clamping to values that are too large will have no visible effect. On the other hand, values that are too small may lead to rapid loss of diversity and premature convergence [23]. Additionally, the type of clamping applied may directly influence the behaviour of the particles (as described in sections 4.1.2 and 4.1.2).

This chapter proceeds as follows: section 4.1 introduces two clamping strategies for consideration. Section 4.2 performs a number of experiments that examine the rela-

tionship between the optimal clamping value and problem dimensionality for each of the clamping strategies, compares the two clamping strategies in terms of performance and ascertains whether velocity clamping is sufficient to prevent particles from roaming outside the search space. Lastly, section 4.3 contains the chapter's closing remarks.

4.1 Clamping Methods

Velocity clamping was introduced to prevent rapid velocity growth which may cause particles to leave the boundaries of the search space [20]. It may also be used to control the ratio of exploration to exploitation performed by the swarm [23], by controlling the granularity of particle steps.

Velocity clamping limits a particle's step size to some maximum, typically chosen as a fraction of the search space. The granularity of the search is thus determined by the chosen velocity limit. The optimal value for the maximum step size is problem dependent [46, 70]. Note that velocity clamping does not modify or confine the positions of the particles directly; particles can still leave the search space.

Dynamic methods of velocity clamping - where the maximum velocity changes as a function of the search iteration - have also been proposed. Among these are strategies such as decreasing the maximum velocity when there has not been any improvement of the global best position for a number of iterations [66], allowing the maximum velocity to decay exponentially [28], and constraining velocities using the hyperbolic tangent function [23].

This chapter considers two common methods of static velocity clamping, which are discussed in the following two subsections.

4.1.1 Method 1 - Clamping Per Dimension

The first method, as proposed by Eberhart and Kennedy [20], chooses some maximum value for the velocity in each dimension, denoted by $v_{max,j}$. Each dimension j of the velocity vector is then examined and if its absolute value exceeds the chosen maximum, $v_{max,j}$, the j-th velocity component's value is adjusted. The velocity vector is updated

in each dimension j according to

$$v_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1} & \text{if } -v_{max,j} \leq v_{i,j}^{t+1} \leq v_{max,j} \\ v_{max,j} & \text{if } v_{max,j} < v_{i,j}^{t+1} \\ -v_{max,j} & \text{if } v_{i,j}^{t+1} < -v_{max,j} \end{cases} \tag{4.1}$$

The value for $v_{max,j}$ is typically chosen as a fraction of the search space. Let the bounds of the search space for dimension j be given by $[L_j, U_j]$, then

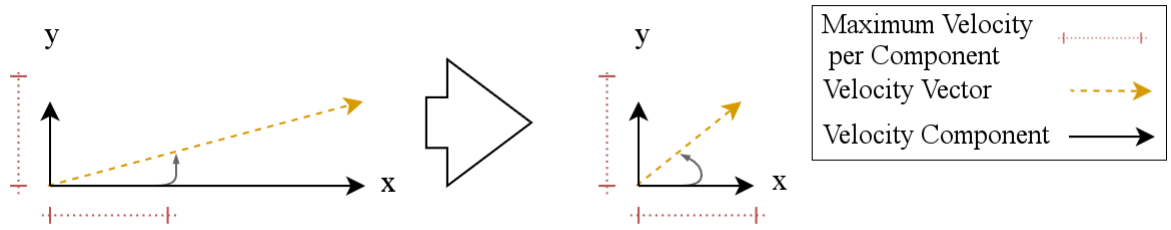
$$v_{max,j} = \delta(U_j - L_j) \tag{4.2}$$

where $\delta \in (0, 1)$.

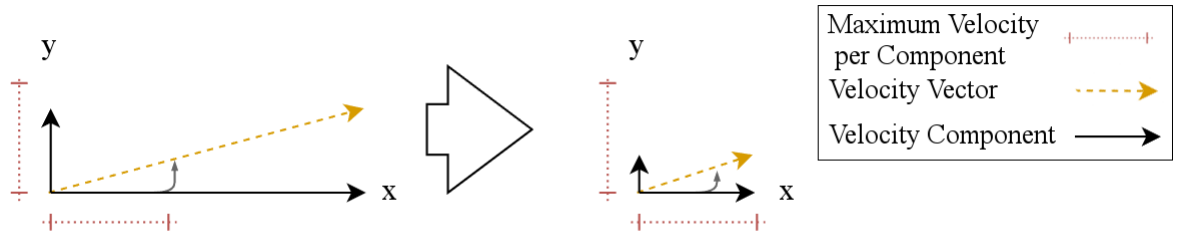
For the problems under consideration, the search space was the same in all dimensions so that $L_1 = L_2 = ... = L_n$ and $U_1 = U_2 = ... = U_n$. Therefore, the upper and lower limits of the search space in dimension j will therefore be denoted as U and L respectively.

Observe that the direction of a particle in a single dimension is thus not changed by velocity clamping (since a positive $v_{i,j}^{t+1}$ remains positive and vice versa). However, if $n > 1$, then the direction of the velocity vector is changed by applying velocity clamping as illustrated in figure 4.1a for the 2-dimensional case. The black arrows indicate the velocity components in the x-y plane, the dotted red lines next to the x and y components indicate the maximum velocity for each component, and the orange arrow indicates the magnitude and direction of the velocity vector. Velocity clamping per dimension may thus force particles into unfavourable regions of the search space, by distorting information from the local and global best positions and from the particle's previous trajectory [62, 67].

Clamping per dimension has another potential problem. If the particle velocities are clamped in all dimensions, then all the velocity components are equal to the maximum velocity. The particles are thus restricted to the boundaries of a hypercube defined by $[\mathbf{x}_i^t - \mathbf{v}_{max}, \mathbf{x}_i^t + \mathbf{v}_{max}]$. Although it is possible that the optimum may be found in this region, it is unlikely [23, 67]. The hypercube problem can be mitigated by using an inertia weight (as done in this chapter) or by decreasing the maximum velocity over time.



(a) Velocity Clamping with method 1 - per dimension



(b) Velocity Clamping with method 2 - magnitude

Figure 4.1: Illustrations of the effects of velocity clamping

4.1.2 Magnitude - Method 2

Velocity clamping based on the magnitude of the velocity vector aims to preserve the search direction of the particle. Instead of examining every dimension of the velocity vector individually, this strategy considers the total magnitude of the velocity vector. If the magnitude of the particle's velocity exceeds some chosen maximum, then the entire velocity vector is adjusted so that its magnitude is within bounds, but its direction is preserved. This is mathematically expressed by

$$\mathbf{v}_i^{t+1} = \begin{cases} \mathbf{v}_i^{t+1} & \text{if } \|\mathbf{v}_i^{t+1}\| \leq v_{max} \\ \frac{v_{max}}{\|\mathbf{v}_i^{t+1}\|}\mathbf{v}_i^{t+1} & \text{if } \|\mathbf{v}_i^{t+1}\| > v_{max} \end{cases} \tag{4.3}$$

where $\|.\|$ denotes the Euclidean norm. As was the case for clamping per dimension, the value for v_{max} is determined based on some fraction of the maximum step size. Whereas clamping per dimension only considers the maximum step size possible in each

dimension, this strategy must consider the maximum distance that a particle can travel (i.e. the largest magnitude that the velocity vector can attain).

As before, let the bounds of the search space for dimension j be given by $[L_j, U_j]$. Then

$$v_{max} = \delta \sqrt{\sum_{j=1}^{n} (U_j - L_j)^2} \tag{4.4}$$

$$= \delta \|\mathbf{U} - \mathbf{L}\| \tag{4.5}$$

where $\mathbf{U} = [U_1, U_2, ..., U_n]^T$ and $\mathbf{L} = [L_1, L_2, ..., L_n]^T$ are n-dimensional vectors.

This approach does not modify the velocity's direction and thus preserves the information from the momentum, social and cognitive components (see figure 4.1b). However, this method is susceptible to outliers in the sense that all the velocity vector's components will be adjusted even if there are only a few large components. A velocity that consists of a few very large components and many small components will be penalized just as heavily as a velocity vector that is rather large in many dimensions. This may cause the particles to move very slowly in most dimensions due to very large velocities in other dimensions.

4.2 Experimental Results

This section presents empirical results and observations regarding the behaviour of the two velocity clamping strategies as problem dimensionality increases. Two sets of experiments were performed. The experimental method followed was very similar to the method described in Chapter 3, except that velocity clamping was applied.

The first experiment examined the relationship between problem dimensionality and the optimal value for δ and is discussed in section 4.2.1. The first experiment was used to empirically determine an optimal δ for each strategy. The second experiment, described in section 4.2.2 compared the two clamping methods in terms of swarm performance (using the optimal δ values found in the first experiment). Finally, section 4.2.3 analyses the better-performing strategy to determine whether the strategy successfully mitigated the symptoms exhibited by PSO on the high dimensional problems.

4.2.1 Optimal δ-Values

In order to determine an optimal δ for each clamping strategy, a number of different values were tried, i.e. $\delta = \{0.005, 0.01, 0.05, 0.1, 0.25, 0.5\}$. Larger values of δ were not tested because preliminary empirical tests showed no difference in swarm behaviour for $\delta = 0.5$ and unclamped swarms. Each swarm configuration was tested on a number of different problem dimensionalities, i.e. $n \in \{10, 100, 250, 500, 750\}$. Each combination of δ and n were run on each benchmark function 30 times, to provide 30 independent samples (as in Chapter 3).

The optimal value for δ was determined for each strategy over all the functions of a given problem dimensionality (using the benchmark suite in Appendix A). It is thus assumed that the optimal δ-value is dimension dependent. The comparison among different δ-values was performed in a similar manner to [6] and is described below.

Let the problem dimension n and velocity clamping strategy be fixed. Let two different algorithm configurations (i.e. with two different values for δ) be denoted by g and h. For every (g, h) pair and function f, define $s_{g,h,f}$ as follows

$$s_{g,h,f} = \begin{cases} 1 \text{ if } R_{g,f} < R_{h,f} \text{ with } p \leq 0.05 \\ 0 \text{ if } p \geq 0.05 \\ -1 \text{ if } R_{h,f} < R_{g,f} \text{ with } p \leq 0.05 \end{cases} \tag{4.6}$$

where $R_{g,f}$ denotes the median of the best score attained by configuration g when optimizing function f and p denotes the confidence bound of the Mann-Whitney U test. For every configuration pair (g, h), the "wins" of g over h is measured in terms of a point system. This measure is denoted by $z_{g,h}$ and is calculated by

$$z_{g,h} = \begin{cases} 3 \text{ if } \sum_{f=1}^{F} s_{g,h,f} > 0 \\ 1 \text{ if } |\sum_{f=1}^{F} s_{g,h,f}| = 0 \\ 0 \text{ if } \sum_{f=1}^{F} s_{g,h,f} < 0 \end{cases} \tag{4.7}$$

where F denotes the number of functions in the benchmark suite. For the benchmark suite in appendix A, $F = 20$.

The total "score" of a configuration g is the sum of its wins over all the other con-

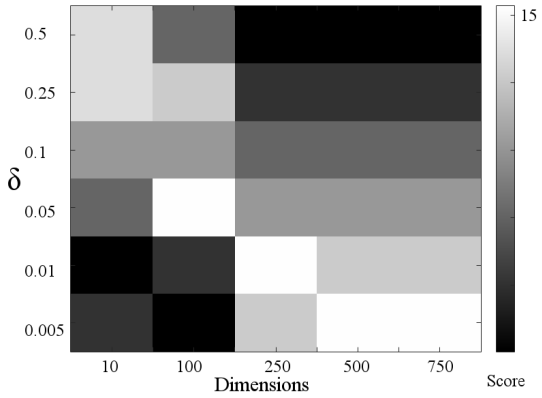


Figure 4.2: Performance of method 1 as δ and n vary (lighter is better)

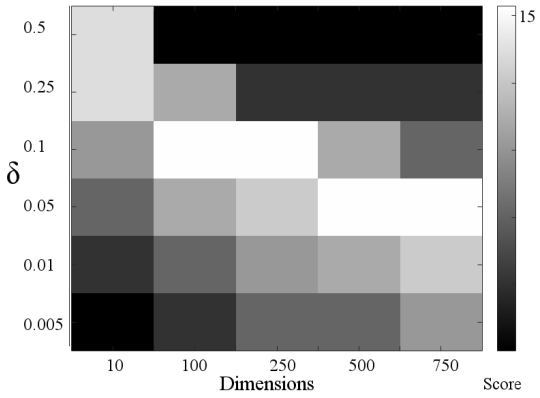


Figure 4.3: Performance of method 2 as δ and n vary (lighter is better)

figurations. The total score, M_g, of configuration g is denoted by

$$M_g = \sum_{h=1;h\neq g}^{J} z_{g,h} \tag{4.8}$$

where J denotes the number of configurations. Since six different values for δ were considered, $J = 6$. If $M_g > M_h$, then configuration g performed better on the benchmark suite than configuration h.

The results of the comparison described above are plotted in figures 4.2 and 4.3. The figures plot the δ-values on the y-axis and the problem dimensionality on the x-axis. The score of configuration g determines the lightness of its block. So the best configuration for a given n can be found by finding the lightest block in n's column. Figure 4.2 shows the scores for the per-dimension clamping strategy and figure 4.3 shows the scores for clamping based on velocity magnitude. The optimal values for δ for both velocity clamping methods are also given in Table 4.1.

The figures show that for both clamping methods, as the dimensionality increased, the optimal value for δ (denoted by δ^*) decreased significantly by orders of magnitude. Thus in higher dimensional search spaces, particle step sizes must be made smaller. Smaller step sizes will facilitate locally exploitative behaviour, which has been shown to improve performance in high dimensional spaces [81, 82]. These results will be examined closely for each clamping method in the two subsections that follow.

Table 4.1: Optimal δ-value for problem dimensionality

	Method 1	Method 2
n = 10	0.5 or 0.25	0.5 or 0.25
n = 100	0.05	0.01
n = 250	0.01	0.01
n = 500	0.005	0.05
n = 750	0.005	0.05

4.2.1.1 Optimal δ Method 1

When considering the higher dimensional problems ($n = 500$ and $n = 750$), method 1's δ^* was the smallest value under consideration ($\delta = 0.005$).

In unclamped swarms, the velocity explosion becomes more pronounced as problem dimensionality is increased. This is not simply because there are more terms in the velocity magnitude expression ($\sqrt{\sum_{i=1}^{n} v_i^2}$), as shown by figures 4.4 and 4.5, which provide component-wise information about swarm velocities.

Figure 4.4 shows the average maximum velocity component for all the particles in an unclamped swarm, as problem dimensionality varies from 10 to 5000. Figure 4.5 shows the average minimum velocity component for all the particles in an unclamped swarm. As the problem dimensionality increases, the range of particles' velocity shifts higher. In other words, particle velocities become larger as the problem dimensionality increases, even when examined component-wise. It is thus intuitive that smaller δ values will perform better as n increases.

The fact that small δ-values perform well can be further explained by considering the nature of high dimensional problems: as n increases, the hyper-volume of the search space increases exponentially, making the likelihood of encountering the region containing the global optimum far smaller. It has been suggested that in such high dimensional spaces it may be more fruitful to employ locally exploitative strategies [82]. Smaller values of δ, which force local exploitation, are thus advantageous for high dimensional problems. (Later chapters elaborate on the idea of focusing on local exploitation rather

than exploration).

Figure 4.2 establishes that δ^* decreases as the problem dimensionality increases. It is not immediately clear whether the trend continues as $n \to \infty$ or whether there is convergence to a particular δ-value.

Further simulations were run on problem sets of even higher dimensionality to test whether the optimal δ-value continued to decrease as dimensionality increases. The simulations were run for $\delta = \{0.005, 0.001, 0.0005, 0.0001\}$ on problem suites with dimensionality in $n = \{1000, 1500, 2000, 5000, 10000\}$.

Figure 4.6 plots the score of each δ-value for a problem suite of given dimensionality, where lighter colours indicate better scores. Scores were calculated as described earlier in this section. As problem dimensionality became very large, the optimal δ-value remained 0.005 and smaller values degraded performance. Thus, δ^* converges to a value near 0.005 as n goes to infinity.

Based on these observations, the relationship between δ and n appears to be an exponential model of the form

$$\delta^* = ae^{bn} + c \tag{4.9}$$

where a, b and c are constants. Fitting an exponential curve of this form (using Matlab's *fit* function with the parameter *exp1*) produced the model

$$\delta^* = 0.3977e^{-0.02159n} + 0.005 \tag{4.10}$$

as depicted in figure 4.7. The exact coefficients for the model are likely dependent on the algorithm parameters such as swarm size, inertia weight and acceleration coefficients. Nevertheless, showing that the relationship between δ^* and n is exponential remains an interesting result.

In summary: if the δ-values are too large, relative to the search space, then velocity clamping fails to prevent the velocity explosion. The higher the problem dimensionality, the greater the velocity explosion (again, consult figures 4.4 and 4.5), and thus smaller values of δ are required. However, δ^* converges to a fixed value as n goes to infinity. Convergence of the optimal δ-values implies that there comes a point where the repercussions of mitigating the initial velocity explosion are too high: although the initial

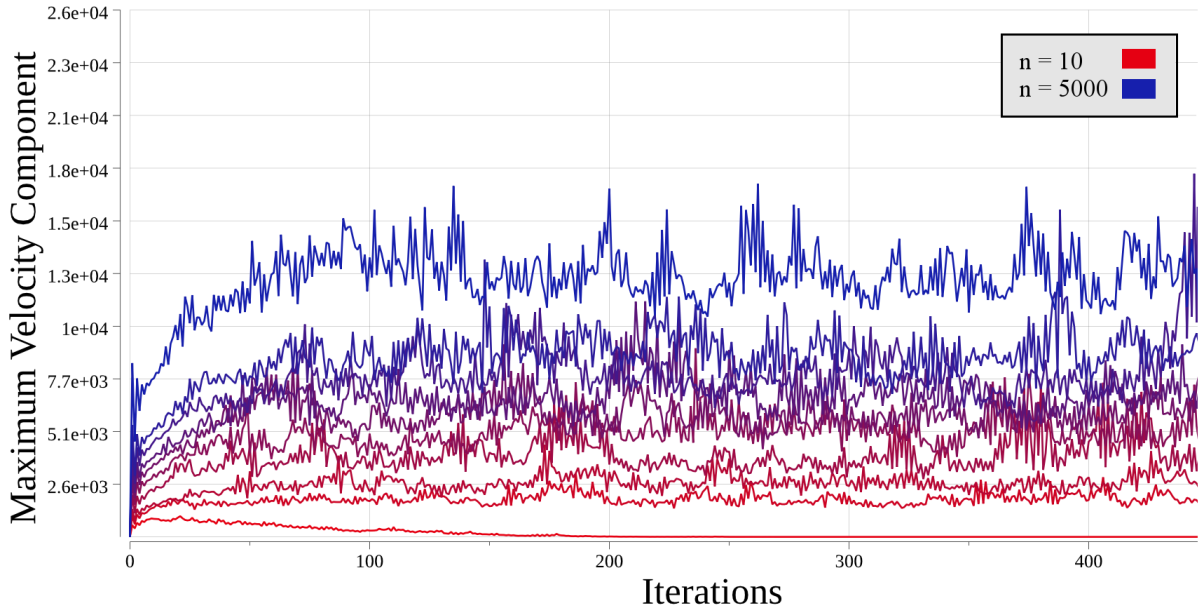


Figure 4.4: Maximum velocity component for F1 across dimensionalities (averaged over 10 runs, n in $\{10, 50, 100, 250, 500, 750, 1000, 1500, 2000, 5000\}$)

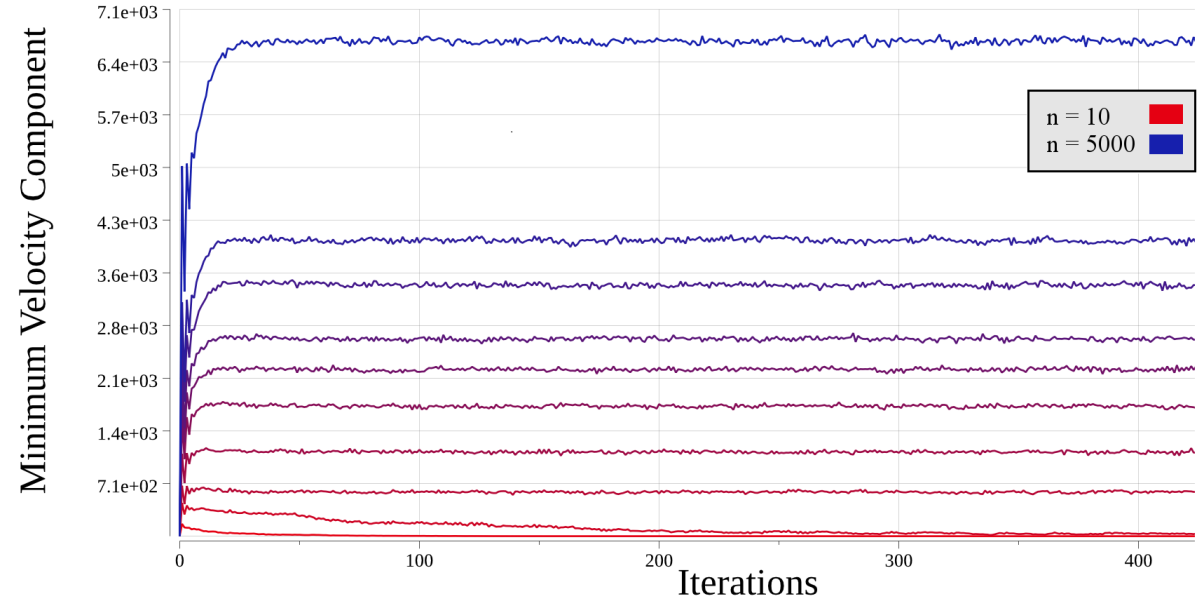


Figure 4.5: Minimum velocity component for F1 across dimensionalities (averaged over 10 runs, n in $\{10, 50, 100, 250, 500, 750, 1000, 1500, 2000, 5000\}$)

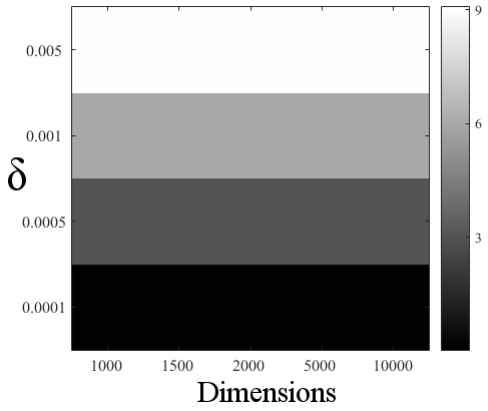


Figure 4.6: Performance of method 1 for smaller δ and larger n (lighter is better)

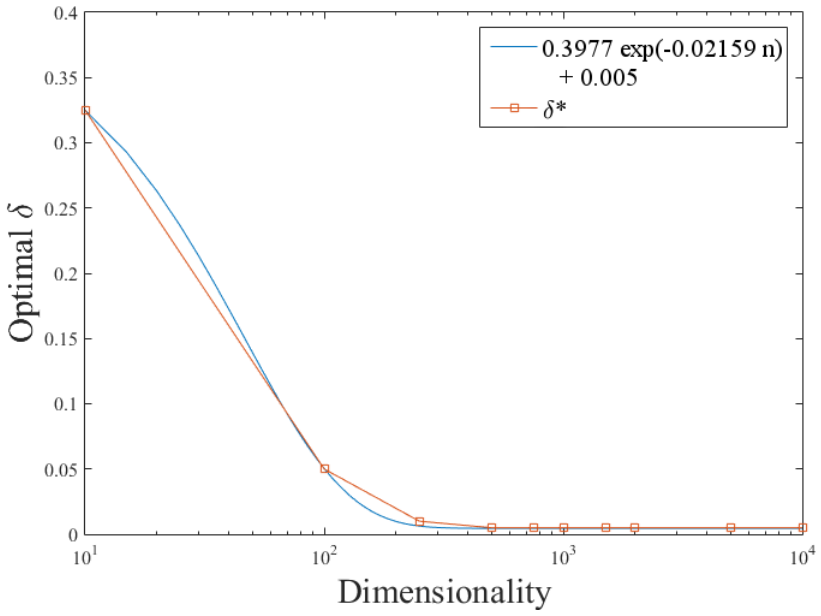


Figure 4.7: Optimal delta for dimensionality, exponential model

velocity explosion can be contained, applying the same clamping value throughout the search impedes the swarm's ability to find a good solution.

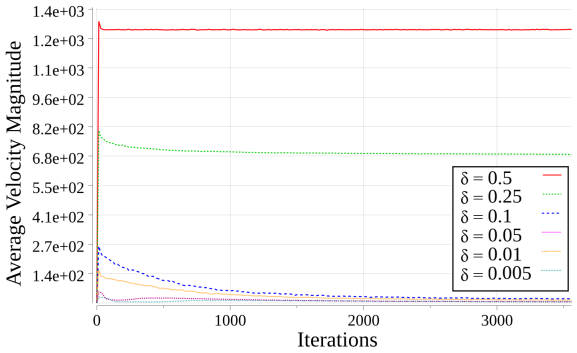


Figure 4.8: Average velocity of swarm for F14 by method 1 (averaged over 30 runs, $n = 750$)

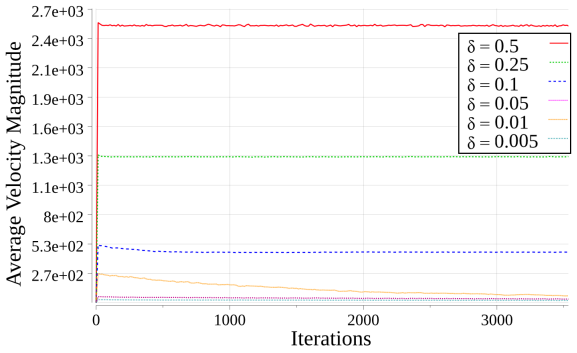


Figure 4.9: Average velocity of swarm for F14 by method 2 (averaged over 30 runs, $n = 750$)

4.2.1.2 Optimal δ Method 2

Considering the discussion regarding the importance of granular searching in section 4.2.1.1, it is counter-intuitive that method 2 performed better when using larger δ values. It is even more unexpected when one observes that for a fixed δ, method 2's average velocity magnitude is usually larger than that of method 1's by an order of magnitude (as depicted by figures 4.8 and 4.9). The reason for method 2's larger average velocities sheds light on the counter-intuitive optimal values for δ, as explained below.

Figure 4.10 shows that, for both clamping methods, more than half of the velocity components were below the average, implying that there were a few large components that skewed the average upward. Figure 4.10 also illustrates that, when method 2 was applied, a particle's velocity component had more small components than when applying method 1. The figure was generated by calculating the average absolute component value of a particle's velocity, then counting the number of components with absolute value below that average.

When using method 2, a velocity vector needs only one very large component for clamping to be applied on the entire vector. If the component is very large, all the other components in the vector will be heavily penalized, since clamping preserves direction. To make matters worse, the large component will remain large after clamping, relative to the other components and will likely require clamping in the next iteration, hence

preserving a large velocity magnitude.

In comparison, per dimensional clamping would not have penalized the reasonably sized velocities at all and instead would have decreased the problematic velocity component to a point where future clamping may not have been necessary. This is illustrated in figures 4.8 and 4.9, which shows the initial velocity spike decreasing more sharply for swarms clamped by method 1, than for swarms clamped by method 2.

Figure 4.10: Number of velocity components below the average absolute component velocity on F17, 750 dimensions

As n increases, the probability of the particle leaving the search space due to a large velocity in at least one dimension increases [34], thereby increasing the occurrences of "unfair" clamping. This explains why method 2 performs increasingly poorly as problem dimensionality increases. The concept of unfair clamping also explains why the optimal value for δ is larger than for method 1: if the value of δ is very small, then the unfairly penalized components of the velocity vector are even smaller, to the point where very little search progress is made in the majority of the problem dimensions (refer again to figure 4.10).

Table 4.2: Pairwise comparison of clamping methods across problem dimensionalities

	Method 1			Method 2		
	+	=	-	+	=	-
n = 10	1	19	0	0	19	1
n = 100	10	3	7	7	3	10
n = 250	13	1	6	6	1	13
n = 500	16	0	4	4	0	16
n = 750	17	2	1	1	2	17

4.2.2 Clamping Method Comparison

From Table 4.1, an optimal configuration for each clamping method can be obtained for a given n. Using these values, the two clamping methods were compared against each other to determine which method is best for a given dimensionality and whether this changes as the dimensionality increases.

To compare the two algorithms, a Friedman test was performed with a confidence bound $p \leq 0.05$ to test whether the difference in their performance was statistically significant. If the Friedman test indicated a significant difference, Mann-Whitney U tests were performed with a p-value of 0.05. The number of wins (functions for which one method performed statistically significantly better than the other), draws (functions on which there was no statistically significant difference in performance) and losses were calculated for each n. These values are reflected in table 4.2.

In low dimensions ($n = 10$), the two methods are not significantly different on all but one function. This may largely be due to the fact that in low dimensional spaces, the particles are not clamped as frequently as in higher dimensions. As the problem dimensionality increases, the strategy of clamping per dimension (method 1), gradually outperforms clamping based on magnitude (method 2). For $n = 100$, method 1 wins on half of the functions. For the highest dimensionality ($n = 750$), method 1 statistically significantly outperforms method 2 on all but 3 of the functions. Clearly, method 1 is the better choice, particularly for high dimensional problems.

The maximum step size allowed by method 2 depends directly on the number of problem dimensions. Thus, as n increases, the particles are allowed larger step sizes. Thus, for large n, the clamping method fails to enforce sufficiently granular searching. As discussed in subsection 4.2.1, simply decreasing δ is not a solution because it will slow down searching in dimensions that already had sufficiently small velocity components.

4.2.3 Swarm Behaviour

The question of whether the velocity clamping was "successful' has not yet been answered. Velocity clamping serves two main purposes: controlling the ratio of exploration to exploitation and preventing particles from leaving the search space. Thus, this subsection examines these two aspects of swarm behaviour. Since clamping particle velocities per dimension has been shown to perform better than clamping based on magnitude, clamping per dimension (method 1) will be discussed in this subsection.

Figure 4.11 shows the fraction of the swarm that was out of bounds at every iteration. For the largest δ, the swarm quickly left the search space and failed to return for the remainder of the search. For smaller values of δ, the fraction of the swarm outside the search space decreases. But even for the smallest δ-values, some of the particles left the search space after the first few hundred iterations. This behaviour was more pronounced for some of the other benchmark functions such as F9. Thus, even clamping to a very small velocity can not prevent particles from roaming outside the search space, although it may mitigate the problem to some extent.

Additionally, although the performance of method 1 was better than method 2, the actual results obtained on the benchmark functions were still far from the optimal values of 0 (see Table 4.3). Although small values of δ discouraged particles from leaving the search space, small step sizes may also cause the swarm to converge prematurely to a local minimum. For almost all the benchmark functions, the swarm diversity became very small within the first 500 iterations, an indication of premature convergence (see figure 4.12 for a typical example).

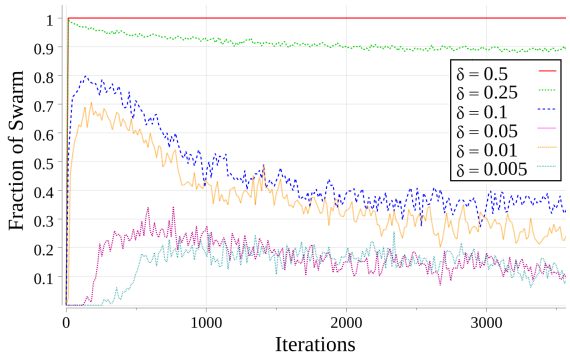


Figure 4.11: Fraction of swarm out of bounds for F9 by method 1 and 750 dimensions

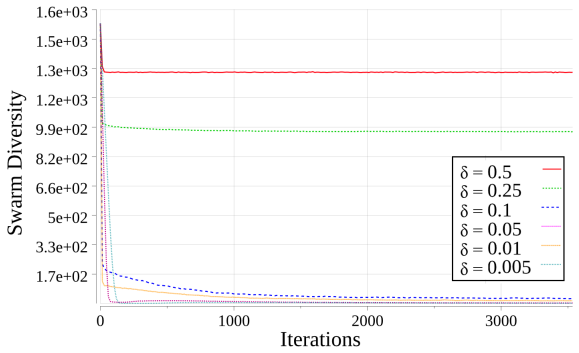


Figure 4.12: Swarm diversity for F9 by method 1 and 750 dimensions

4.3 Summary

This chapter examined whether velocity clamping can successfully be applied to prevent particles from leaving the search space in high dimensional spaces. The chapter aimed to investigate the relationship between the optimal maximum velocity and problem dimensionality. Two different velocity clamping strategies were considered: method 1 clamped every dimension independently and method 2 clamped the entire velocity vector in a way that preserved its direction.

The optimal values to which the particle velocities should be clamped was found to be dependent on problem dimensionality, even when clamping by dimension. It was observed that, as the number of problem dimensions increases, the optimal maximum step size decreases for both of the velocity clamping strategies.

For a given problem dimensionality, the optimal value for δ when clamping based on magnitude was higher than when clamping per dimension. This counter-intuitive observation can be explained by noting that method 2 will "unfairly" adjust reasonably-sized velocity components because of a few very large velocity components in order to preserve the velocity's direction. Small velocity components will be even smaller after clamping is applied, thus slowing down the search process for those dimensions.

Although the two clamping methods performed very similarly for low-dimensional spaces ($n = 10$), clamping per dimension became increasingly advantageous as the prob-

Table 4.3: Mean and standard deviation of best fitness achieved by method 1 with $\delta = 0.005$ and 750 dimensions

Function	**Mean**	**Standard Deviation**
F1	$2.3641E+08$	$1.1711E+07$
F2	$6.6518E+03$	$2.2892E+01$
F3	$1.9696E+01$	$6.6803E-03$
F4	$9.0220E+09$	$1.0233E+09$
F5	$4.5411E+07$	$1.9031E+06$
F6	$1.8913E+07$	$1.3660E+05$
F7	$6.3700E+02$	$2.4809E+01$
F8	$2.7861E+06$	$3.9762E+05$
F9	$3.8684E+08$	$5.9134E+06$
F10	$6.4532E+03$	$2.0604E+01$
F11	$7.3341E+02$	$4.3445E-01$
F12	$7.2441E+04$	$2.3541E+03$
F13	$1.7914E+06$	$2.2766E+05$
F14	$7.1905E+08$	$1.2383E+07$
F15	$6.6617E+03$	$2.2755E+01$
F16	$1.4567E+03$	$6.6091E-01$
F17	$9.8013E+04$	$3.8881E+03$
F18	$4.9748E+06$	$3.9487E+05$
F19	$6.4384E+08$	$2.0608E+07$
F20	$1.6417E+08$	$1.6433E+08$

lem dimensionality grew, to the point where clamping per dimension outperformed the other strategy on 17 out of the 20 benchmark functions for $n = 750$. It is intuitive that the clamping strategy which is independent of n should perform better as problem dimensionality becomes very large.

When examining the actual behaviour of the swarm, it was observed that velocity clamping was relatively successful in reducing the number of particles outside the search

space. However, even for the most restrictive δ-values, there were always some particles outside the search space. Furthermore, there were cases in which clamping prevented initial particle explosion of velocities, but eventually the particles left the search space anyway.

Even the best-performing swarm configurations found solutions very far from the optimum for the high dimensional problems. The swarms with small maximum velocities mostly succeeded in confining the particles to the search space, but were prone to premature convergence.

Clearly, velocity clamping alone is not effective at mitigating the velocity explosion while simultaneously preserving swarm diversity. Simply forcing the particles to take smaller steps is thus not the answer to the symptoms of high dimensionality. The next chapter will take a more nuanced approach to limiting particle step sizes by carefully choosing values for the acceleration coefficient and inertia weight.

Chapter 5

Variance of Particle Positions

This chapter examines the variance of particle positions and how it is influenced by the inertia weight and acceleration coefficients of PSO. A brief overview of existing literature on the variance of particle positions is given in section 5.1. This chapter examines the advantages and disadvantages of certain movement patterns exhibited by PSO in high dimensional spaces. The inertia weight and acceleration coefficients can be used to control the variance of particle positions, thereby controlling the swarm's range of movement. By choosing parameters so that the variance of swarm positions is small, the velocity explosion may be mitigated. This approach towards preventing particle roaming is examined in section 5.2. Particular values for w, c_1 and c_2 can also be used to bring about certain patterns of particle movement such as controlling the level of oscillation or the smoothness of particle trajectories. Section 5.3 examines different movement patterns and determines which are more suitable to high dimensional spaces. Section 5.4 concludes the chapter.

5.1 A Brief History of Variance

This section provides a brief overview of the literature regarding the variance of particle positions and its relationship to the inertia weight and acceleration coefficients.

Usually, theoretical analysis of particle behaviour requires some form of the *stagnation assumption* which assumes that the personal best positions and the global best

position have stopped improving. Under stagnation, each particle behaves independently of the other particles since no new information is introduced by a global or personal best position update. Thus, each particle's behaviour can be studied separately when stagnation is assumed. Furthermore, each particle's dimensions are independent, so the particle need only be analyzed in one dimension. This significantly reduces the complexity of solving the equations. Due to the independence of particles and problem dimensions, the subscript i and j for particle positions and velocities may be dropped for the purposes of the discussion that follows.

Jian *et. al.* [36] performed convergence analysis of the expectation and variance of particle positions. The analysis yields *convergence regions* for the choice of w, c_1 and c_2 parameter values, within which all the particles in the swarm are guaranteed to converge in expectation. Poli [56] extended the results of [36], by deriving an expression for the variance of a particle's position, which allows practitioners to choose parameters that will bring about certain behaviours in the variance (such as self-limiting growth or convergence to a fixed point). Poli also emphasized the importance of variance dynamics in understanding and influencing PSO's search behaviour.

Poli [56] proved that under stagnation, an arbitrary particle will converge to an arbitrary fixed point if all the following conditions hold:

$$c < \frac{12(w^2 - 1)}{5w - 7} \tag{5.1}$$

$$c > 0 \tag{5.2}$$

$$-1 < w < 1 \tag{5.3}$$

where $c_1 = c_2 = c$. The fixed point for the variance of a particle's positions is given by

$$\sigma^2 = \frac{c(w + 1)}{4(c(5w - 7) - 12w^2 + 12)}(\hat{y} - y)^2 \tag{5.4}$$

Liu [44] proved that the convergence regions found by [36] and [56] are valid even under a weak stagnation assumption. Liu [44] extended the findings of [36, 56] by assuming that w, $\phi_1 = c_1 r_1$ and $\phi_2 = c_2 r_2$ are arbitrary random variables with known expected values and variances (whereas previous analysis had assumed w to be constant and ϕ_1 and ϕ_2 to be uniformly distributed random variables). Bonyadi and Michalewicz

[5] also assumed that y and $\hat{y}$ are random variables as opposed to constant values. Using the stangant distribution assumption, Bonyadi and Michalewicz derived convergence boundaries that are necessary (proven theoretically) and sufficient (proven empirically) for convergence of position variance. Cleghorn and Engelbrecht [14] made a further generalization under the non-stagnant distribution assumption (i.e. without assuming stagnation) where the personal and global best positions were considered as convergent sequences of random variables. The analysis presented by Cleghorn and Engelbrecht can be applied to obtain the convergence boundaries for general classes of PSOs. Furthermore, the assumption that the expected value and variance of personal and global best positions are convergent sequences was shown to be a necessary condition for convergence (i.e. is the weakest possible assumption under which the expected value and variance of particle positions converge).

5.2 Restricting Variance

This section uses existing theory regarding the variance of particle positions to restrict particle movement. Section 5.2.1 explains that the variance of particle positions can be reduced to a chosen fraction of its original value by calculating corresponding values for the inertia weight and acceleration coefficients. Swarms with variance restricted to different values were then tested on the high dimensional benchmark suite from Appendix A. Section 5.2.2 describes the experimental method and section 5.2.3 analyses the results of the experiments. The effects of restricting the variance are discussed and the best performing swarm configurations are identified. Section 5.2.4 summarizes this section and provides motivation for the section that follows.

5.2.1 Restricting the Variance of Particle Positions

In the experiments that were performed in Chapter 3, the stagnation assumption held after the first iteration, because all the particles left the search space and personal and global best positions were only updated if the position was valid. Since the particles did not return to the subspace, the remainder of the search was under stagnation.

According to Poli [56], the standard deviation of a particle's position under stagnation

is given by

$$\begin{aligned}\sigma &= \frac{1}{2}\sqrt{\frac{c(w+1)}{c(5w-7)-12w^2+12}}|\hat{y}-y| \\ &= V_c|\hat{y}-y| \end{aligned} \tag{5.5}$$

where

$$V_c = \frac{1}{2}\sqrt{\frac{c(w+1)}{c(5w-7)-12w^2+12}} \tag{5.6}$$

Observe that, if the conditions for convergence in expectation and variance hold (equations (5.1) to (5.3)), then the standard deviation will only be zero if $y = \hat{y}$.

The experiments in Chapter 3 used convergent parameter values, which guarantee that the expected value and variance of the particle positions will converge to a constant. In high dimensions, the particles leave the search space immediately, so the personal best and global best positions are never updated. Therefore, the variance of the particle positions immediately becomes a large constant. The swarm thus converges within the first iteration. Even if the attractors are confined to the search space, the resulting variance is large enough for all the particles to stay outside the search space.

For the usual "good parameters" for the inertia weight and acceleration coefficients ($w = 0.7298$ and $c_1 = c_2 = 1.49618$) that were used in Chapter 3, the coefficient in equation (5.5) evaluates to

$$\frac{1}{2}\sqrt{\frac{c(w+1)}{c(5w-7)-12w^2+12}} \tag{5.7}$$

$$= \frac{1}{2}\sqrt{\frac{1.49618(0.729844+1)}{1.49618(5\dot{0}.729844-7)-12\dot{0}.729844^2+12}} \tag{5.8}$$

$$= 1.0432 \tag{5.9}$$

which is larger than one. The maximum possible value for $|y-\hat{y}|$ is the range of the search space, given by $U-L$. Thus, for the given c and w values, the maximum possible standard deviation is larger than the size of the search space. If $|y-\hat{y}|$ for each of the particles in the swarm and the particles are distributed normally around the center of

the search space, approximately 38% of the particles will be located within the search space (i.e. within half a standard deviation of the center). The probability of a particle's next position being outside the search space is thus much higher than the probability of being inside the search space, therefore most of the swarm will be located outside of the search space (see figure 5.1).

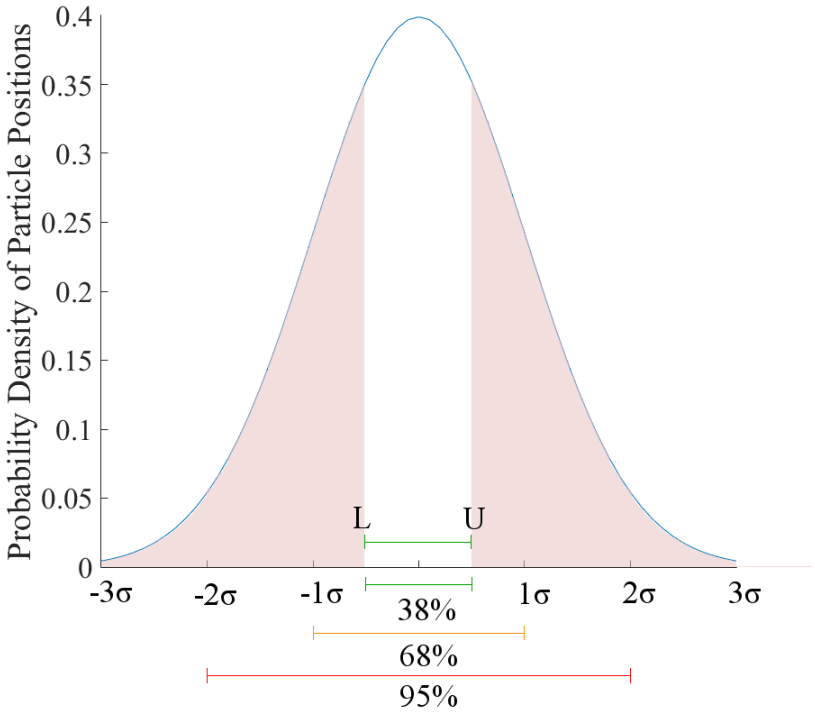


Figure 5.1: Probability distribution of swarm for worst case scenario in 1-D

Of course, this is a worst case scenario. Consider a better scenario, in which the distance between the personal best position and the global best position is approximately half of the search space, i.e. $|y - \hat{y}| = \frac{1}{2}(U - L)$. On average, approximately 68% of the particles will be found within the search space (i.e. within one standard deviation of the center), which leaves a third of the particles outside the search space.

If the value of $|y - \hat{y}|$ is known or can be estimated, then equation (5.5) can be solved for values of c and w that ensure that a particle is within the search space. If y and $\hat{y}$ are both independent, uniformly distributed variables between L and U, then the expected

value of $|y - \hat{y}|$ is zero. However, y and $\hat{y}$ are not independent, since both variables depend on the location of the benchmark function's optimum.

Thus, calculating the coefficient V_c to obtain an exact standard deviation is a difficult, problem-dependent exercise. Instead, the value of the coefficient V_c can be used to restrict the standard deviation to some fraction of what it would be otherwise. For example, suppose that the aim is to restrict the standard deviation to some fraction, γ, of what it would be otherwise. Then setting the value of V_c to γ and solving for c yields:

$$\gamma = V_c \tag{5.10}$$

$$= \frac{1}{2}\sqrt{\frac{c(w+1)}{c(5w-7) - 12w^2 + 12}} \tag{5.11}$$

$$\therefore c = \frac{12(1-w^2)}{7 - 5w + \frac{w+1}{4\gamma^2}} \tag{5.12}$$

By allowing w to range between 0 and 1 and solving for the corresponding c-value, it is possible to produce a set of (c, w) pairs for which the standard deviation will be the desired fraction of what it would have been otherwise.

It should be noted that, although two pairs of (w, c) values may restrict the standard deviation to the same amount, those two pairs may bring about very different search behaviours in the swarm because the w and c coefficients play very different roles. For example, the pair $(0.1, 1.5632)$ and $(0.9, 0.5182)$ both restrict the standard deviation fractionally by a half (i.e. $\gamma = 0.5$ and the standard deviation is half of what it would be otherwise), the different configurations cause very different behaviour in the swarms: the first swarm will have almost no momentum and high acceleration coefficients whereas the second swarm has very high momentum and low acceleration coefficients.

Two questions arise:

- To what fraction should the standard deviation be restricted?
- How are the optimal w and c chosen from the resulting sets?

Both of these questions will be answered by empirical study in section 5.2.3. The remainder of this section is devoted to motivating the method of restricting the standard deviation fractionally (i.e. so that it is a certain fraction of what its value would be otherwise) and its applicability to high dimensional spaces in particular.

The discussion regarding the standard deviation of a particle's position has thus far considered only a single dimension, since all dimensions are independent under stagnation. However, another phenomenon arises as more dimensions are considered. An n-dimensional particle's standard deviation, σ, will be a vector with each component j given by

$$\sigma_j = V_c|\hat{y}_j - y_j| \tag{5.13}$$

As the dimensionality grows, the Euclidean length of σ grows, since every additional dimension will add a non-negative term to the norm that depends on the distance between $\hat{y}_j$ and y_j, as shown below:

$$\|\sigma\| = \sqrt{\sum_{j=1}^{n} \sigma_j^2} \tag{5.14}$$

$$= \sqrt{\sum_{j=1}^{n} (V_c|\hat{y}_j - y_j|)^2} \tag{5.15}$$

$$= \sqrt{V_c^2 \sum_{j=1}^{n} |\hat{y}_j - y_j|^2} \tag{5.16}$$

$$= V_c \sqrt{\sum_{j=1}^{n} |\hat{y}_j - y_j|^2} \tag{5.17}$$

Unless the particle is the global best, it is unlikely for $|\hat{y}_j - y_j|$ to be zero in many dimensions. Thus, the hyper-volume within which particles are likely to be found grows with dimensionality, and may be larger than the search space in hyper-volume. As an example, consider figure 5.2, which depicts a 2-dimensional search space with bounds $[-2, 2]^2$. Suppose that $|\hat{y}_j - y_j| = \frac{1}{2}(U - L) = 2$, for each j. Then the particles within one standard deviation of the search space center can be found anywhere in the circle

with radius

$$R = V_c \sqrt{\sum_{j=1}^{n} |\hat{y}_j - y_j|^2} \tag{5.18}$$

$$= V_c \sqrt{(2^2 + 2^2)} \tag{5.19}$$

$$= V_c 2\sqrt{2} \tag{5.20}$$

$$= V_c \sqrt{2} \frac{1}{2}(U - L) \tag{5.21}$$

$$= V_c \sqrt{2} |\hat{y}_j - y_j| \tag{5.22}$$

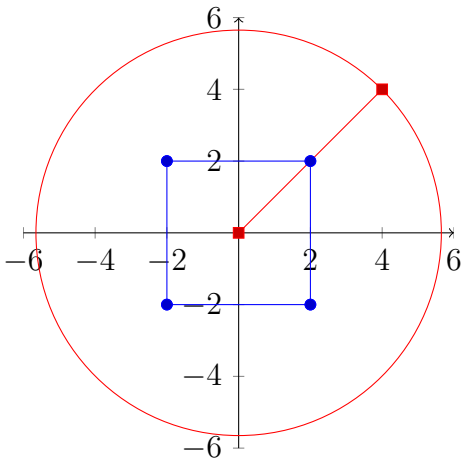


Figure 5.2: Hypersphere within one standard deviation of the search space centre (where $|\hat{y}_j - y_j| = 2$, which is half the search space)

Thus, the distance between the personal and global best positions in any dimension is approximately half of the search space and if the upper and lower bounds are the same in all dimensions, the length of the standard deviation is given by

$$V_c \sqrt{n} \frac{1}{2}(U - L) \tag{5.23}$$

If the radius of the search space is defined by

$$R_{ss} = \sqrt{\sum_{i=1}^{n} \left(\frac{U - L}{2}\right)^2} \tag{5.24}$$

$$= \frac{U - L}{2} \sqrt{n} \tag{5.25}$$

and is used as a measure of the search space's size, then the ratio between the length of the standard deviation and the search space's radius is given by V_c. The coefficients c and w, which determine V_c, thus play an important role in determining the range of the swarm's movement.

It should be noted that particles further than one standard deviation from the center of the search space are thus guaranteed to be out of bounds (i.e. 32% of the swarm will be out of bounds). In a worse scenario (if $|\hat{y}_j - y_j| > U_j - L_j$ or if V_c is much larger than one), then even more particles will be outside the search space.

Since the size of the standard deviation grows with problem dimensionality, the optimal γ to which to restrict the standard deviation may depend on n. If so, it may thus be possible to obtain coefficients that will negate the effect of dimensionality on the standard deviation's norm and restrict the particles' movement, hopefully preventing roaming.

5.2.2 Experimental Method

Table 5.1 summarizes all the c-w combinations that were tested and the corresponding γ values. Three different γ values were tested, i.e. $\gamma \in \{1.0, 0.75, 0.5\}$. For each γ, w was varied from 0.1 to 1.0 in increments of 0.05, so that

$$w \in \{0.1 + 0.05k \mid 0 \leq k \leq 18\} \tag{5.26}$$

For the chosen (γ, w) pair, the corresponding value for the acceleration coefficient was calculated according to equation (5.12). Except for $w = 1$, where the corresponding c values were unique (i.e. regardless of γ, if $w = 1$ then $c = 0$). Each configuration was run on the benchmark suite for 30 independent runs, as in the experiments in Chapter 3.

5.2.3 Variance and Dimensionality

Restricting the standard deviation influenced the swarm's average velocity magnitude, as expected. It may have been that the swarm was not roaming due to the distance between the particles' personal and global best positions, but rather because the swarm

Table 5.1: Parameter configurations for swarms with fractionally restricted standard deviations

$\gamma = 1.0$			$\gamma = 0.75$			$\gamma = 0.5$		
γ	c	w	γ	c	w	γ	c	w
1.0	1.7535	0.1	0.75	1.6998	0.1	0.5	1.5632	0.1
1.0	1.7943	0.15	0.75	1.7349	0.15	0.5	1.5851	0.15
1.0	1.8286	0.2	0.75	1.7633	0.2	0.5	1.6	0.2
1.0	1.8557	0.25	0.75	1.7841	0.25	0.5	1.6071	0.25
1.0	1.8747	0.3	0.75	1.7967	0.3	0.5	1.6059	0.3
1.0	1.8846	0.35	0.75	1.8	0.35	0.5	1.5955	0.35
1.0	1.8841	0.4	0.75	1.7929	0.4	0.5	1.575	0.4
1.0	1.8719	0.45	0.75	1.774	0.45	0.5	1.5435	0.45
1.0	1.8462	0.5	0.75	1.7419	0.5	0.5	1.5	0.5
1.0	1.8049	0.55	0.75	1.6947	0.55	0.5	1.4431	0.55
1.0	1.7455	0.6	0.75	1.6302	0.6	0.5	1.3714	0.6
1.0	1.6649	0.65	0.75	1.5457	0.65	0.5	1.2833	0.65
1.0	1.5592	0.7	0.75	1.4381	0.7	0.5	1.1769	0.7
1.0	1.4237	0.75	0.75	1.3034	0.75	0.5	1.05	0.75
1.0	1.2522	0.8	0.75	1.1368	0.8	0.5	0.9	0.8
1.0	1.0366	0.85	0.75	0.9322	0.85	0.5	0.7239	0.85
1.0	0.7664	0.9	0.75	0.6817	0.9	0.5	0.5182	0.9
1.0	0.4274	0.95	0.75	0.3754	0.95	0.5	0.2786	0.95
1.0	0.0	1.0	0.75	0.0	1.0	0.5	0.0	1.0

was attracted to a position near the boundary of the search space. Then the particles exploring the region near the global best may have been likely to be out of bounds in at least a few dimensions, due to the positioning of the attractor, even though the particles'

velocities were not very high. If this was the case, then restricting the variance would not reduce the swarm's average velocity.

Figures 5.3 to 5.7 illustrate the typical profile of the swarms' average velocity magnitude. Each figure is for a fixed inertia weight and shows the effect of varying γ. In all cases, the smaller standard deviation resulted in lower average velocity, as expected.

Figures 5.8 to 5.10 show the average velocity magnitude for a fixed γ as w varies. For each γ, lower inertia weights result in higher average velocities and vice versa. As γ becomes larger, the diversity increases asymptotically as w decreases. Thus, the movement of swarms with higher inertia weights is generally more restrained than swarms with lower inertia weights.

Restrictions to the standard deviation combined with high w-values cause the corresponding acceleration coefficients to have relatively low values. Such swarms will rely more heavily on momentum than on new information received from its neighbours, or learned throughout its own search. These particles will exhibit smooth trajectories due to the regularizing influence of the large momentum component and because any direction alterations introduced by their personal and global best positions will be small, since the acceleration coefficients are small.

If the inertia weight is low and the acceleration coefficients are high, then the particle is more likely to divert its course if its attractors are updated. If the location of the particle's attractors changes drastically, from one side of the search space to the other, this may lead to the particle taking huge steps, causing velocity explosion. If the inertia weight is high and the acceleration coefficients are low, then the pull of a particle's attractors will be exerted more gradually, preventing the particle from oscillating between the extremes of the search space due to updating attractors.

Figure 5.11 shows the score of a swarm with a given γ and w on the 1000 dimensional problem suites. The score of a configuration was calculated as described in section 4.2.1 using the swarm's best fitness achieved throughout the search. The color of a block shows its score, with lighter indicating a better score. Comparisons were done across all combinations of γ and w, so that the overall best combinations can be determined. Figure 5.12 visualizes the performance of all the c-w combinations. For every parameter configuration, the corresponding score of the configuration is reflected in the color of the

data point, where pink is the best possible score and blue is the worst score.

Table 5.2: Coefficient values for best-performing combinations

γ	c	w	Score
0.5	0.2786	0.95	162
0.5	0.5182	0.9	159
1.0	0.4274	0.95	156
0.75	0.3754	0.95	153
0.75	0.6817	0.9	150

Table 5.3: Coefficient values for worst-performing combinations

γ	c	w	Score
1.0	0.00	1.0	23
0.5	1.575	0.4	25
1.0	1.8462	0.5	25
0.5	1.2833	0.65	26
1.0	1.5592	0.7	28

The w, c and score of the five best combinations are given in table 5.2. In general, severe restrictions to the standard deviation (i.e. lower γ) and high inertia weights performed the best. Values for w below 0.75 generally did not perform well in comparison to the others. The best performing swarms varied across all three chosen γ, but all swarms had high inertia weights and relatively low acceleration coefficients.

Relying on inertia weight rather than immediately utilizing information regarding newly found good positions does not seem like exploitative behaviour in the sense that the particle is emulating the success of its attractors. However, it is a very granular form of searching. There is less danger of the particle taking huge steps from one end of the search space (due to the particle's attractors being updated) because the influence of the social and cognitive components is small. Instead, the particle's trajectory may be pulled towards the updated attractors gradually, without fear of velocity explosion (as discussed previously, based on figures 5.8 to 5.10).

The w, c and score of the five worst combinations are given in table 5.3. The worst performing swarms were either $\gamma = 1$ or $\gamma = 0.5$. The worst configuration used only inertia and had no social or cognitive component whatsoever. It is not surprising that the swarm did not perform well, since the particles could not incorporate any additional information into their search direction. The search will devolve into sampling a number of points along a line, where the line's direction was determined by the randomly initialized personal and global best positions. The other configurations that scored in the bottom five all had large acceleration coefficients and medium-range inertia weights. Disconcertingly, some of these values were not far away from the rule of thumb "good" parameters.

Simply restricting the standard deviation is thus not enough to ensure good performance: some of the worst-performers had severely restricted standard deviations. The swarm's performance is also dependent on the chosen values for w and c. This is not unexpected: just because a swarm's movement is restricted to be within the search space does not guarantee that the swarm will find a good solution within the search space.

Figure 5.13 shows the typical diversity profiles of the five best-performing configurations. In general, the configurations with better scores exhibited lower diversities. It is interesting to note that, although the chosen value for w and c plays a larger role in determining the swarm's behaviour, the role of γ is still visible here, even when comparing across different inertia weights: the swarms for which $\gamma = 1$ showed increasing diversity, in comparison to the configurations for which $\gamma < 1$, where the swarm diversity decreased or, at least, did not increase after the initial spike.

In general, larger γ values exhibited higher diversities and more dimensions out of bounds (see figures 5.13 and 5.14), although large values for c also played a role. For example, when $\gamma = 1$ and $c = 0.4274$, the diversity was lower than for $\gamma = 0.75$ and $c = 0.6817$. When c is larger, the local and global attractors exert more influence over the particle trajectories, allowing the particles to be diverted towards updated attractors and increasing the swarm's diversity. Usually, none of the configurations' diversities went to zero, indicating that the particles did in the vicinity of a single point, even for the configurations with very low acceleration coefficients.

From figures 5.11 and 5.12, there appear to be three points with low inertia weights

and high acceleration coefficients that also perform well (circled in red on figure 5.12). However, further investigation of these configurations yielded nothing interesting about these configurations. In general, the swarms were out of bounds for almost the entire search and their personal best positions were almost never updated. The behaviour of these three points did not seem very different from their surrounding neighbours, as can be seen in figures 5.15 and 5.16. Figure 5.15 plots the diversities of the three outlier points and figure 5.16 includes the diversities of the outlier's surrounding neighbours for comparison. Since the calculated scores are relative, it may simply be that these points scored slightly higher than the other parameter configurations in the same vicinity, but their performance was still poor.

5.2.4 Summary

Particle movement can successfully be restricted by choosing an inertia weight and acceleration coefficients to reduce the variance of particle positions. For a given inertia weight, decreasing γ caused the swarm to exhibit lower velocity magnitudes, thereby reducing the step sizes of the particles and encouraging a more granular search.

For a fixed γ, increasing the inertia weight caused the swarm's average velocity magnitude to decrease and vice versa. High inertia weights regularize the particle's movement because the particle is less likely to divert its course rapidly if the positions of its attractors changes. Thus, if the particle's attractors change drastically from one iteration to the next, a momentum-focused particle will be less prone to oscillation and velocity explosion than a particle that has high acceleration components.

It was found that high inertia weights and low acceleration coefficients perform the best in high dimensional spaces. Although such configurations are not "highly exploitative", such configurations are restrictive because particle trajectories can only change gradually (due to the low weighting of their social and cognitive components relative to their momentum). Thus, granular searching and smooth trajectories appear to be the key aspects to searching successfully in high dimensions.

The analysis performed in this section was based on empirical observations regarding why particles roam. The section that follows makes use of existing literature regarding the swarm's movement patterns to provide additional insight into the behaviour of PSO

in high dimensional spaces.

5.3 Frequency and Variance of Particle Positions

The value of the coefficients w, c_1 and c_2 play a large role in the search behaviour of the swarm and directly influences the patterns of movement exhibited by the particles [75, 7]. Trelea [75] categorized the movement patterns into four groups:

1. non-oscillatory - the particle's position does not oscillate throughout the search,
2. harmonic - the particle's position oscillates smoothly in a wave-like motion,
3. zigzagging - the particle's position oscillates significantly from one iteration to the next, and
4. harmonic-zigzagging - the particle's position displays a combination of harmonic and zigzagging motion.

Trelea [75] also divided the convergence region (for convergence in terms of the expectation of particle positions) into sub-regions, depending on what type of behaviour is brought about by the parameters w, c_1 and c_2 in those regions.

The movement pattern exhibited by a swarm may be characterized in terms of its range of movement and the smoothness of its particles' trajectories. Particle trajectories may be smooth (with positions strongly positively correlated between iterations) or oscillatory (with positions strongly negatively correlated between iterations). Particle trajectories can also exhibit various degrees of chaos, when subsequent positions are only weakly correlated or are independent.

This section examines different movement patterns and identifies behaviours that perform well in high dimensional spaces. The section is also intended to provide further insight into the findings from the previous section. The inertia weights and acceleration coefficients that performed well previously may correspond to certain classes of movement patterns, making it easier to interpret the swarm's behaviour. Section 5.3.1 introduces the concepts of the base frequency and range of movement of particle positions and discusses their influence on the swarm's movement patterns. Section 5.3.2

describes the settings of the experiment, which examines the influence of the base frequency and range of movement on the swarm's behaviour in high dimensional spaces. Lastly, section 5.3.3 examines the results of the empirical experiment and determines what swarm configurations and corresponding movement patterns are optimal in high dimensional spaces in terms of performance.

5.3.1 Base Frequency and Range of Movement

A particle's trajectory may be characterized in terms of the particle's range of movement and its base frequency. The base frequency of a particle, denoted by F, is defined to be the largest amplitude among the Fourier series coefficients of the particle's positions throughout the search [7]. The base frequency influences the oscillatory behaviour exhibited by a particle. Particles with small values for F typically exhibit smooth trajectories, while large F-values are prone to more oscillations with large steps between positions.

Range of movement refers to the size of the hyper-volume bounded by all the particles' positions from the start to the end of the search. The range of movement is characterized by the variance of a particle's position, denoted by σ^2 where σ is defined as in equation (5.5). The coefficient V_c is independent of the global and local best positions, and characterizes the swarm's ability to explore.

For a given F and V_c, corresponding velocity update coefficients, w, c_1 and c_2 can be calculated as described by [7] and the previous section. If the velocity update coefficients are assumed to be constant and $c_1 = c_2 = c$ and $w > 0$, then the simultaneous equations (5.27) and (5.28) are sufficient to calculate the corresponding velocity update coefficients:

$$c = 1 + w - 2cos(2\pi F)\sqrt{w} \tag{5.27}$$

$$= \frac{-48V_c w^2 + 48V_c}{28V_c + w - 20V_c w + 1} \tag{5.28}$$

Equations (5.27) and (5.28) allow the practitioner to choose a pattern of movement for particle trajectories that is most suited to the problem and to calculate values for w, c_1 and c_2 that bring about the desired behaviour.

The base frequency, F should be in the range $[0, 0.5]$ where $F = 0$ implies that the particle's position does not oscillate at all. On the other hand, $F = 0.5$ means that

the particle's position will oscillate with every iteration. Bonyadi and Michalewicz [7] examined the relationship between the base frequency and the correlation of particle positions. Figure 5.17 from [7] shows the correlation measure as a function of the base frequency. Each point is the correlation for a given frequency averaged across different variance values, with bars indicating the standard deviation. Bonyadi and Michalewicz observed that positions are positively correlated when $F < 0.25$, so the particle moves smoothly, with high dependence between subsequent positions. When $F > 0.25$, particle positions are negatively correlated implying no dependence between subsequent positions. When F is near 0.25, the correlation between particle positions is close to zero, so particle movement may be chaotic.

The variance of movement, V_c, can be used as a measure a particle's search range. Its value can be any value larger than zero. However, if the value of V_c is very small, then the particle is not guaranteed to sample positions outside the boundaries $[\min\{y, \hat{y}\}, \max\{y, \hat{y}\}]$, because the expected value of the particle's position is between its personal and global best positions [77]. Thus, the particle's exploration ability will be limited.

The following theorem from [7] (simplified for constant w, c_1 and c_2) provides a minimum bound for V_c to ensure that at least one of the particle's positions is not on the line between y and $\hat{y}$.

Theorem 5.1. *Assume that $y \neq \hat{y}$. If*

$$V_c > \left(\frac{\max\{c_1, c_2\}}{c_1 + c_2} \right)^2 \tag{5.29}$$

then for any distribution of x_i^t generated by the velocity update equation (2.7), the number of points generated by x_i^t outside the interval

$$(\mathbb{E}(x_i^t) - \min\{y, \hat{y}\}, \mathbb{E}(x_i^t) + \max\{y, \hat{y}\}) \tag{5.30}$$

is non-zero.

Bonyadi and Michalewicz [7] suggested that large V_c-values are generally the better choice, observing that small values of V_c prevent the particles from exploring the search space sufficiently. In high dimensional spaces, literature and the findings of this thesis suggest that search strategies which focus on local exploitation may be more effective

than searches that attempt exploration, due to the exponential growth of the search space with dimensionality [81, 82]. Thus, smaller values for V_c may prove better than large values, by curbing the particles' exploration.

The sections that follow investigate the relationship between particle movement patterns (brought about by different F and V_c values) and problem dimensionality.

5.3.2 Experimental Method

This section examines whether the movement patterns recommended by literature also perform well in high dimensional spaces. As before, Bonyadi and Michalewicz [7] observed that generally high values for V_c performed well by allowing particles to explore the search space. It was also observed that generally smaller values for F perform better than larger values, regardless of the choice of V_c. High V_c values imply a large range of particle movement and focus on exploration. Small F are associated with smooth particle trajectories with high correlation between subsequent positions.

The experiment was performed as in Chapter 3, but with varying values for c_1, c_2 and w. The experiment tested different values for the frequency and variance of movement with

$$F \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.25, 0.4, 0.45\} \tag{5.31}$$

$$V_c \in \{0.1, 0.4, 1.6, 6.4, 25.6\} \tag{5.32}$$

which is similar to the values tested in [7]. For each (F, V_c) pair, the corresponding values for the acceleration coefficients and inertia weight were calculated using equations (5.27) and (5.28). Setting (5.27) = (5.28) and solving for w yields a polynomial of order 4, which was solved using Matlab's *roots* function. The root with the smallest absolute value and no imaginary component was chosen as the w value. This value was then substituted into both equations (5.27) and (5.28) and the average of the two values was used as c (to compensate for any error in w).

To ensure that the results remain interpretable and are not skewed by the errors introduced when solving for w, the resulting w and c values were substituted back into equations (5.27) and (5.28) to solve for the frequency and variance. The recalculated frequency and variance, denoted by F' and V_c', capture the error on w and c. The resulting

error between the desired frequency and variance (F, V_c), and the actual frequency and variance (F',V_c') can then be calculated. The error on the frequency, E_F, was calculated as

$$E_F = |F - F'| \tag{5.33}$$

The frequency error was never larger than 0.005; that is, the error on the frequency was never larger than 10% of the tested frequency increment of 0.05. The error in frequency was thus deemed sufficiently negligible.

The relative error on the variance, E_{V_c}, was calculated using

$$E_{V_c} = \frac{|V_c - V_c'|}{V_c} \tag{5.34}$$

so that the error is normalized according to the size of the variance, since the tested variances had quite a large range. Figure 5.18 shows E_{V_c} for each F. Generally, the relative error increased as the frequency increased gradually until $F = 0.3$, then decreased again. For $V_c = 0.1$, the relative error became as high as 62%. The relative error of the other variances were lower and never went higher than 16%. The results for the following (V_c, F) pairs were deemed unacceptable due to high relative errors:

$$\{(0.1, 0.25), (0.1, 0.3), (0.1, 0.35)\} \tag{5.35}$$

Although the use of base frequency and variance of movement may prove a useful tool in examining the patterns of swarm behaviour, the error in calculating the corresponding parameter values is problematic. Future work may include finding a different formulation for these calculations that can be solved analytically or more readily by numerical methods.

5.3.3 Movement Patterns in High Dimensions

This section presents the results of the experiments described in the previous section. Section 5.3.3.1 examines all the tested swarm configurations and determines which were the best. The behaviour of the best swarm configurations are examined in detail and possible reasons for their success are presented. Section 5.3.3.2 examines the influence of variance on the swarm's movement.

5.3.3.1 Optimal Frequency and Variance of Movement

This section determines the optimal combinations for F and V_c for both a low and a high dimensional problem suite. From the frequency and variance values, it is possible to determine what kind of movement patterns are exhibited by the swarms that perform well. The section also shows that the optimal movement pattern for the swarm is different for high and low dimensional spaces.

Figures 5.19 and 5.20 show the score of all the (F, V_c) pairs on 10 and 1000 dimensional problem suites. The color of a block shows its score, with lighter indicating a better score. Comparisons were done across all combinations of F and V_c, so that the overall best combinations could be determined for each problem suite.

For the discussions that follow, recall that oscillatory behaviour due to negative correlation between positions occurs for $F \in [0.25, 5]$ whereas trajectories are smooth with positive correlation between positions when $F \in [0, 0.25)$. Chaotic behaviours occurs near 0.25, where correlation between particle positions is near zero.

The frequency-range combinations that perform well are quite different on the low and high dimensional suites. In low dimensions, the frequency $F = 0.05$ never performed well and generally, values $F \in [0.1, 0.2]$ provided good performance. Thus, swarms exhibiting smooth trajectories with some positive correlation between particle positions performed well, though parameters causing very strong correlation between particle positions did not perform well (the lowest F values did not perform as well as frequencies near $0.1 - 0.15$). Thus, some chaotic behaviour is beneficial to the swarm's ability to find good solutions. As variance increases, the performance of larger F-values improved, but never overtook the smaller values. This indicates that, as the range of the swarm's movement increases, some chaotic and oscillatory behaviour is beneficial.

For high dimensional problems, the best performing combinations had low frequency and low variance of movement. The swarms exhibiting oscillatory behaviour performed much worse than smooth trajectories ($F \geq 25$ performed badly in comparison with $F < 0.25$). The variance had less of an effect on swarm performance, but there was still a trend indicating that larger variance is detrimental to performance. Of the five best configurations, four had the smallest possible variance of 0.1 and one had a variance of 0.4. All five of the best performing configurations had frequencies below 0.25, where par-

ticles have smooth trajectories. The frequency and variance as well as the corresponding momentum and acceleration coefficients are given in table 5.4.

Thus particles with low range of movement and smooth trajectories showed the best performance. Low F-values imply that the particles' search was smooth and granular, and that particle positions were highly correlated. Low V_c values imply that the particles did not search in a very large range, thereby preventing the swarm from trying to explore the entire search space. For the lowest possible variance of 0.1, slightly higher frequencies (where position correlation is lower) performed better whereas for the other variances, the lowest possible frequency performed the best. This indicates that, when the particles' range of movement is sufficiently small, some chaos in movement may be beneficial; the small movement range is sufficient to restrict particle movement and small chaotic steps do not cause a velocity explosion. But as particle steps become larger, chaotic movement is detrimental to swarm performance and instead, highly correlated particle positions are required to prevent particles from roaming.

Observations regarding the good performance of smooth trajectories (and small frequencies) agree with previous literature. The observed benefit of using small variance differs from previous findings that were focused on low dimensional problems. However, this is in line with the hypothesis that fine-grained searching within small areas of the search space are the most effective in solving problems in high dimensions. Using small variance to limit particle exploration and small frequencies to ensure smooth, granular movement will bring about the desired behaviour for high dimensional spaces.

Four of the five top combinations had $V_c = 0.1$, which does not satisfy the relation in equation (5.29):

$$\left(\frac{\max\{c_1, c_2\}}{c_1 + c_2}\right)^2 = \left(\frac{c}{2c}\right)^2 = \frac{1}{4} > V_c = 0.1 \tag{5.36}$$

Thus, there was no guarantee that the particles sampled any points outside the region between their personal and global best positions. This does not imply that no points outside this region were sampled, but it does point towards exploitative behaviour by the particles that performed well.

Figures 5.21 and 5.22 show the optimal F-value for a given V_c. Comparisons were thus done within each column (as opposed to over all the possible pairs, as done in figures

5.19 and 5.20). On the low dimensional suite, the optimal frequency was usually around 0.15, which produces smooth particle trajectories, but particle positions are not highly correlated.

For the high dimensional case, the optimal frequency was almost always 0.05, the smallest possible value. Similar to the low dimensional case, $V_c = 0.1$ was the exception. In the high dimensional case, the best frequency for $V_c = 0.1$ was 0.15. For all values of V_c, the optimal frequency was smaller on the high dimensional problems than it was on the low dimensional problems. Thus, highly correlated particle positions and granular searching behaviour is a requirement for good performance in high dimensional spaces. Additionally, observe that, unlike the low dimensional case, none of the F-values that induce oscillatory behaviour performed well in high dimensions. This further supports the hypothesis that fine-grained exploitation performs better than exploration-focused behaviour in high dimensional spaces.

Tables 5.4 and 5.5 show the acceleration coefficient and inertia weight corresponding to the best and worst (F, V_c) combinations. In addition, figure 5.23 plots the c and w values that correspond to the (F, V_c) combinations tested in the experiments. The colour of each data point corresponds to its score in comparison to all the other swarm configurations, where pink dots have the highest score and light blue dots have the lowest score.

The best performing swarms generally had high momentum components and relatively low acceleration coefficients. This corresponds to the findings in figure 5.12, but the additional information provided by the corresponding base frequency and variance of movement has aided the discussion regarding the reasons why these swarms performed well. Figure 5.23 shows no configurations that perform well in the region $c \in (0.58, 1.88), w \in (0.15, 0.3)$, which supports the conclusion that the three outliers in figure 5.12 are not significantly better than their neighbours.

Figures 5.24 and 5.25 show typical diversity profiles of the five best performing swarm configurations. The configuration with the largest variance consistently exhibited higher diversity than the other configuration, which is expected given that the swarm has larger range of movement. The configuration with $V_c = 0.4$ never converged to a point, i.e. the swarm's diversity never went to zero. In contrast, all four of the other top performing

Table 5.4: Coefficient values for best-performing combinations

V_c	F	c	w	Score	Reference
0.1	0.15	0.74883	0.70521	132	Config A
0.1	0.05	0.099381	0.96941	129	Config B
0.4	0.05	0.097915	0.98631	126	Config C
0.1	0.1	0.37326	0.87483	123	Config D
0.1	0.2	1.0965	0.40746	120	Config E

Table 5.5: Coefficient values for worst-performing combinations

V_c	F	c	w	Score
0.4	0.25	1.5763	0.55579	21
1.6	0.2	1.2588	0.81365	23
1.6	0.4	1.8091	0.15443	24
6.4	0.25	1.7052	0.70415	25
6.4	0.3	1.9633	0.51717	25

swarms (all with $V_c = 0.1$) converged very close to a single point. The diversity of the best performing strategy, $V_c = 0.1$ and $F = 0.15$ decreased more slowly than the other strategies with $V_c = 0.1$, while the rest converged prematurely within the first 500 iterations. The c and w of the best configuration were nearly equal, indicating that the particles struck a nearly equal balance between restricting movement to the current trajectory and moving in the direction of good solutions.

According to theory, the correlation between particle positions begins to weaken when $F = 0.15$ as for configuration A. This concurs with the observed behaviour, since configuration A's diversity is higher than for $F = 0.05$ and $F = 0.1$, indicating more exploration and more chaotic particle movement. However, it would then be expected that configuration E's diversity would be even higher, since its particles' movement should be even more chaotic. However, this was not the case. Configuration E had higher acceleration components and a lower inertia weight than the other configurations that performed well ($c = 1.0965$ and $w = 0.40746$). The base frequency parameter thus fails to account for the behaviour of all the swarm configurations.

Figures 5.26 and 5.27 both plot the average number of personal best updates per iteration for configurations A to E. This measure indicates whether the swarm is actively searching and improving its solutions or the swarm has stagnated. These figures varied a lot between benchmark functions, which is to be expected since the measure is highly problem dependent. Generally, configuration A consistently showed improvements throughout the search, with the number of personal best updates decreasing towards the end of the search. Such behaviour adheres to the usual recipe of first exploring and then exploiting, and fits well with the swarm's diversity profiles (though it should be noted that exploration takes place in a limited capacity, due to the inertia weight and acceleration coefficients being nearly equal).

Although configuration B seemed to have converged according to the diversity plot, the swarm was usually actively improving for most of the search, often displaying higher update counts than configuration A, especially later in the search. The swarm's personal best update counts increased just before its diversity dropped, implying that the swarm was exhibiting exploitative behaviour.

Configuration C usually exhibited improvement throughout the search, but with the update counts increasing as the search progressed, showing that the swarm was still exploring and moving towards more fruitful regions (where the other swarms were already exhibiting exploitative behaiour). Configuration C's low acceleration coefficients may explain why the swarm required so many iterations to begin exploiting: since the influence of a particle's attractors was so low, it may take many iterations for a particle's trajectory to move in the direction of its updated attractors.

Configuration D's behaviour varied, sometimes peaking early in the search and then converging prematurely. Other times, configuration D continued to improve throughout the search with update counts just below that of configuration B. Lastly, configuration E's update counts consistently peaked within the first 500 iterations and then went almost to zero, indicating premature convergence.

Of the five most successful strategies, most prevented particles from roaming outside the search space. Configuration C, with the largest variance, occasionally failed to return more than 10% of the swarm's particles to the search space (see figure 5.28). But in general, almost all of the particles returned to the search space throughout the search

(see figures 5.29 and 5.30).

From table 5.4, it is apparent that particle movement can be restricted successfully by having high inertia weights and low acceleration coefficients. Configuration E is the exception, with a low inertia weight and a relatively large acceleration coefficient (though still smaller than the values that are generally accepted as "rule of thumb"). It should be noted that configuration E exhibited worse performance than configuration A to D and was also more prone to premature convergence. It may be that the swarm moved too rapidly towards its attractors, not allowing for sufficient exploration on the way. This prevented the particles from being pulled to and fro between oscillating attractors (and thus preventing the velocity explosion), but it also caused premature convergence.

Based on the number of high-inertia configurations to low-inertia configurations that performed well, it may be easier to find a configuration with a high inertia weight that performs well. The observation that configurations with high inertia weights and low acceleration components perform well concurs with the observations of section 5.2. Incorporating the information provided by the corresponding base frequency and variance values, it can be concluded that swarms with low variance and smooth trajectories perform the best in high dimensional spaces.

5.3.3.2 Analysis of Variance

The diversity profiles of each swarm configuration was very similar across all benchmark functions. Thus, the analysis below for F17 (a randomly selected function) is applicable to the rest of the benchmark suite.

Figures 5.31 to 5.35 plot swarm diversity for a fixed variance as the base frequency varies. Observe that for $V_c = 0.1$, the configurations with $F \geq 0.25$ exhibit high diversity which increases with the frequency. This is not unexpected since $F \geq 0.25$ induces oscillation. The configurations with $F \leq 0.25$ exhibited low diversities that went to zero.

Compare this to figure 5.32 for $V_c = 0.4$. The configurations with $F \leq 0.3$ exhibit low diversities, whereas the configurations with $F > 0.3$ have high diversities. Thus, diversity increases with frequency, until $F = 0.2$, then diversity decreases until $F = 0.3$. After this, the diversity suddenly becomes high and again increases with frequency.

This behaviour is not predicted by the theory or by previous studies. Further, stronger theoretical investigation may be required to explain the observed relationship between the base frequency and swarm diversity.

Figure 5.33 plots the diversity for $V_c = 1.6$. Even fewer configurations exhibit high diversities. As for $V_c = 0.4$, the swarm diversity increases with frequency until $F = 0.2$, i.e. once again, the "turning point" is at $F = 0.2$. After that, increasing frequency leads to decreasing diversities, until $F = 0.4$. For $F = 0.45$, the swarm diversity is considerably higher than for the other configurations.

This behaviour continues as V_c increases (as shown for $V_c = 6.4$ in figure 5.34 and $V_c = 25.6$ in figure 5.35), until eventually none of the frequencies exhibit high diversities. For these configurations, diversity increases with frequency until $F = 0.2$, then decreases as the frequency increases.

It is unexpected that the swarm diversity should decrease for larger frequencies and that this behaviour is exacerbated as the variance increases. The opposite makes more intuitive sense: as the swarm's range of movement increases and particles exhibit more oscillatory behaviour, the swarm diversity should become larger.

Since there is no body of theory that provides much information regarding the swarm's distribution throughout the search, a plausible but unproven explanation is provided here. It may be that, in general, the particles are normally distributed. However, as the variance increases, the distribution becomes wider but also distorted, so that there is a higher probability of particles being at the very centre of the distribution. Thus, for small variances, the overwhelming shape of the distribution is still approximately normal and the diversity increases as the oscillatory behaviour of the particles increases (i.e. as F increases). However, when the variance increases, the distribution becomes distorted and the particles oscillate between two positions that are both near the center (explaining why increasing the frequency decreases the swarm's diversity). If the swarm's oscillatory behaviour is sufficiently large and the variance is not very high, then particles manage to oscillate between distant positions, again causing the diversity to be large (as was the case for $F \in [0.35, 0.45]$ when $V_c = 0.4$). But as the variance becomes very large ($V_c = 6.4$ or $V_c = 25.6$), the shape of the distribution is overwhelmingly distorted and the diversity decreases as oscillatory behaviour increases. Although the

discussion above explains the observed diversity profiles, further theoretical investigation is required.

5.4 Summary

This chapter examined the use of the inertia weight and acceleration coefficients to control the variance of particle positions. Restricting the variance by a fraction successfully reduced the swarm's velocity. As the variance was restricted more severely, the average velocity magnitude of the swarm decreased, as desired. However, simply decreasing the variance of positions is not sufficient to guarantee good performance. Even swarms that were restricted to the same variance on position movements exhibited very different behaviour. Unsurprisingly, the chosen values for the inertia weight and acceleration coefficients were also an important factor.

It was found that swarms with high inertia weights and low acceleration coefficients performed the best in high dimensions. High inertia weights have the effect of regularizing a particle's trajectory, making it smooth and granular. The particle is less likely to rapidly divert its direction when its attractors are updated, since the influence of its momentum is stronger than the influence of its attractors. Thus, if the particle's attractors change drastically from one iteration to the next, a momentum-focused particle will be less prone to oscillation and velocity explosion than a particle that has high acceleration coefficients. Configurations with higher inertia weights consistently exhibited lower velocities than configurations that had high acceleration components (for a fixed variance).

Although such configurations are not exploitative as such, their movement is restricted because particle trajectories can only change gradually (due to the low weighting of their social and cognitive components relative to their momentum). Thus, granular searching and smooth trajectories appear to be the key aspects to searching successfully in high dimensions.

The effect of different movement patterns was also studied. Inertia weights and acceleration coefficients that influence the swarm's range of movement and characteristics of the particles' trajectories (such as smoothness, oscillation and degree of correlation

between consequent positions) were tested on high dimensional problems.

It was found that small base frequencies and low variance were key factors in the swarms that performed well. Small base frequencies are associated with smooth, granular particle trajectories with weak to strong correlation between particle positions. Low variance is associated with restricted range of movement. These experiments also confirmed that swarms with high inertia weights and low acceleration coefficients generally perform better than those with low inertia weights and high acceleration coefficients.

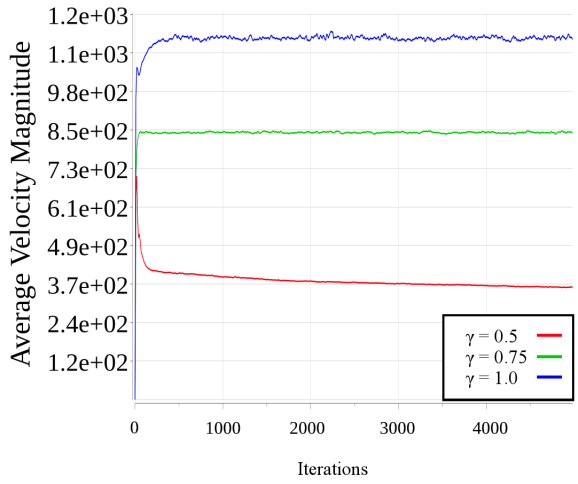


Figure 5.3: Average velocity magnitude for $w = 0.95$ on F7 with $n = 1000$

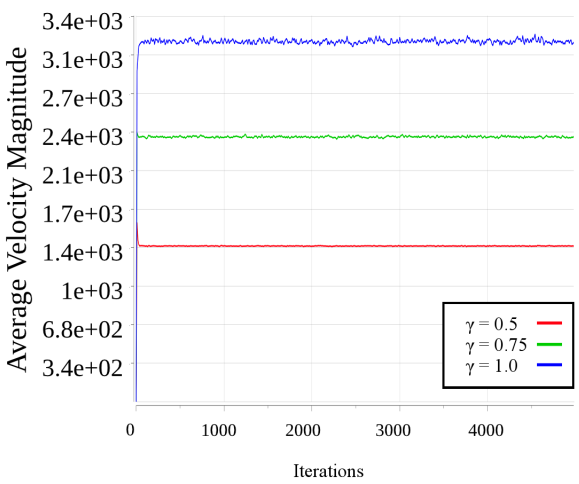


Figure 5.4: Average velocity magnitude for $w = 0.75$ on F7 with $n = 1000$

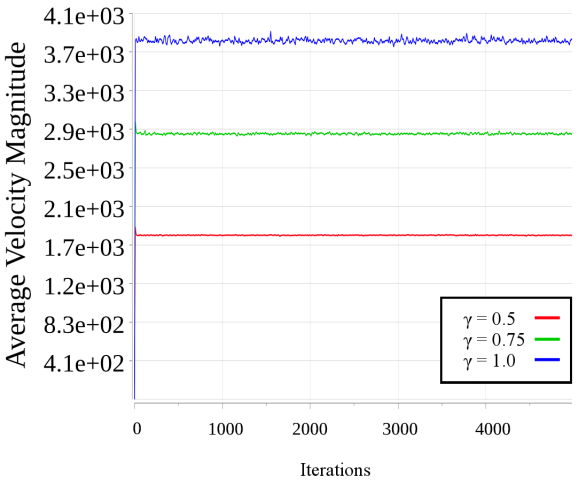


Figure 5.5: Average velocity magnitude for $w = 0.55$ on F7 with $n = 1000$

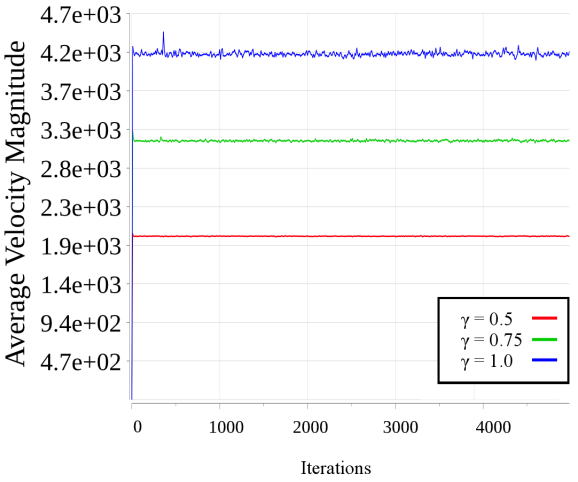


Figure 5.6: Average velocity magnitude for $w = 0.35$ on F7 with $n = 1000$

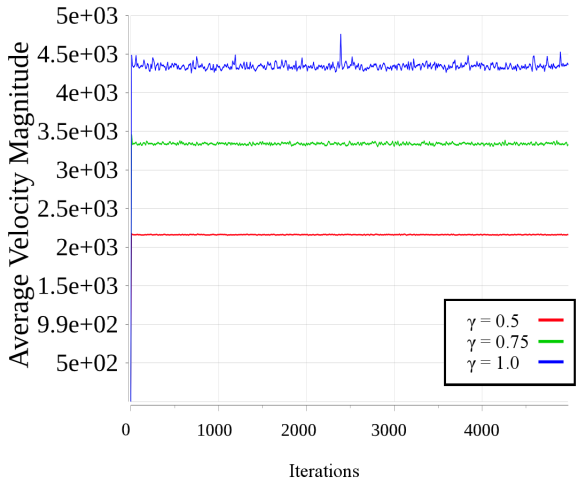


Figure 5.7: Average velocity magnitude for $w = 0.15$ on F7 with $n = 1000$

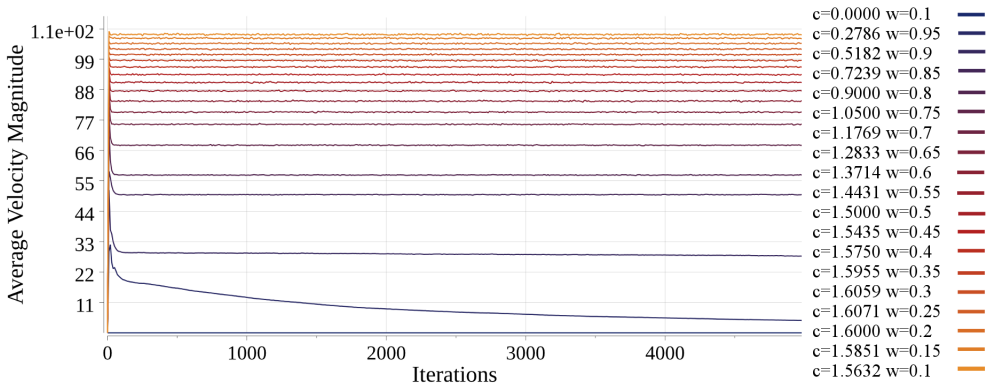


Figure 5.8: Average velocity magnitude for $\gamma = 0.5$ on F2 with $n = 1000$

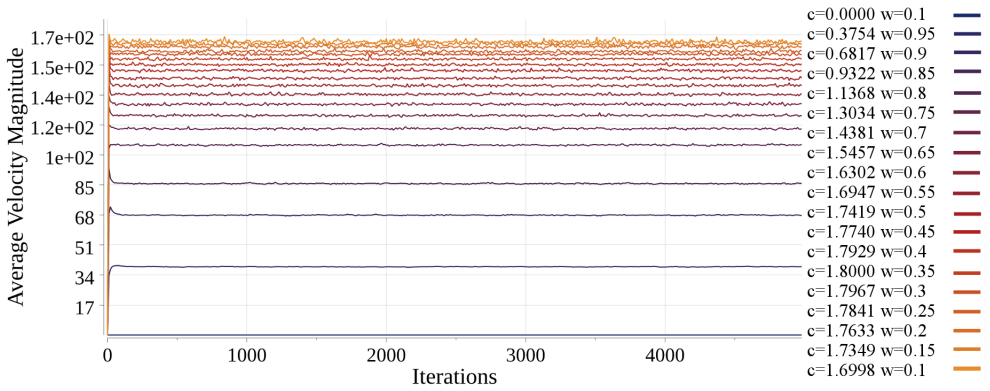


Figure 5.9: Average velocity magnitude for $\gamma = 0.75$ on F2 with $n = 1000$

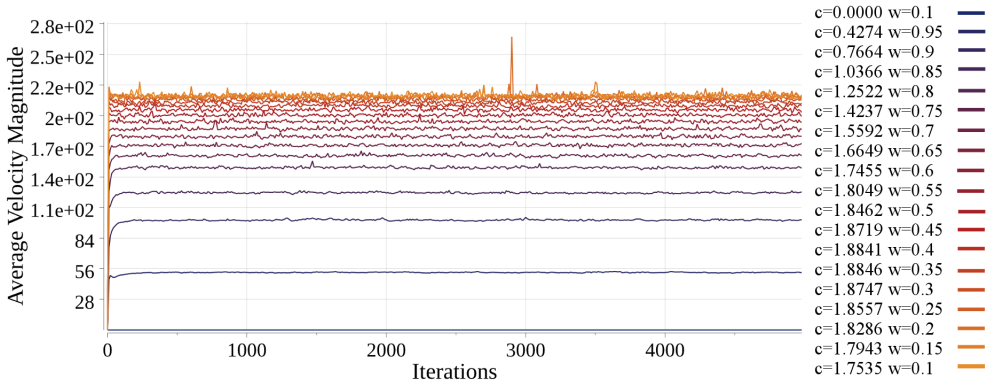


Figure 5.10: Average velocity magnitude for $\gamma = 1.0$ on F2 with $n = 1000$

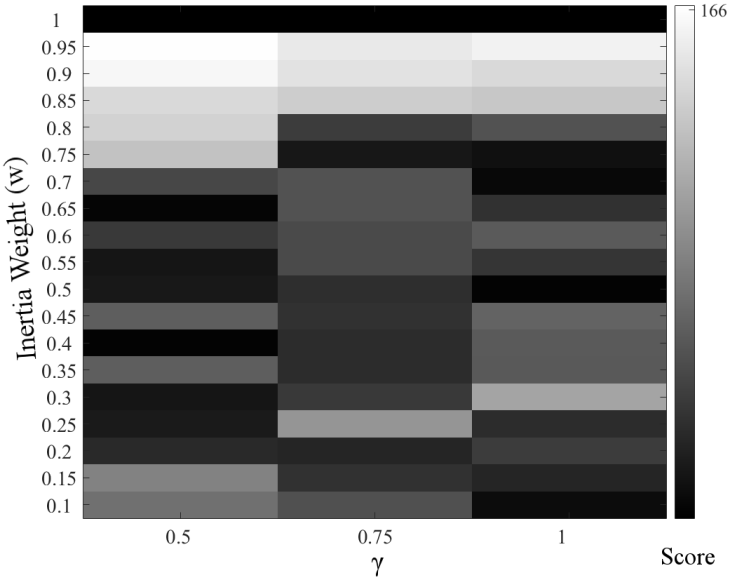


Figure 5.11: Performance of swarm configuration produced by restricting standard deviation of positions

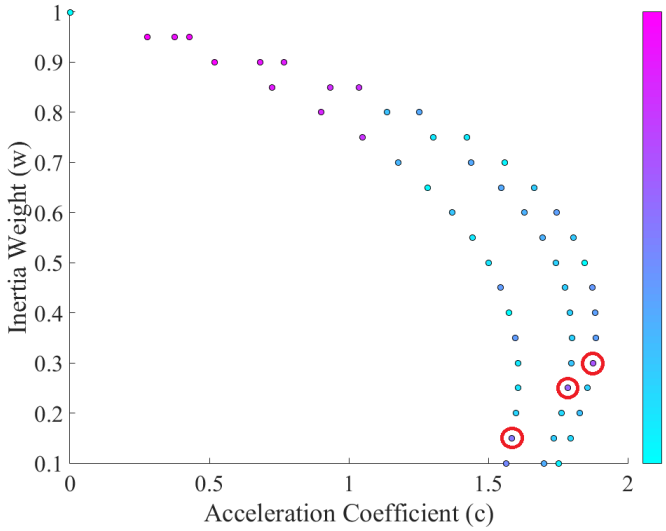


Figure 5.12: Score of acceleration coefficients and inertia weights produced by restricting standard deviation of positions

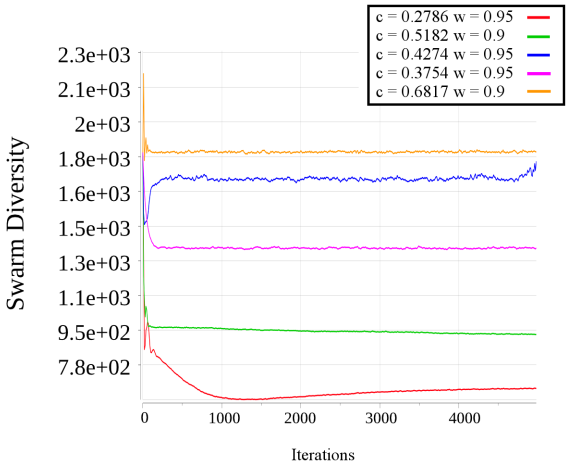


Figure 5.13: Swarm diversity on F18 with $n = 1000$ (5 best configurations)

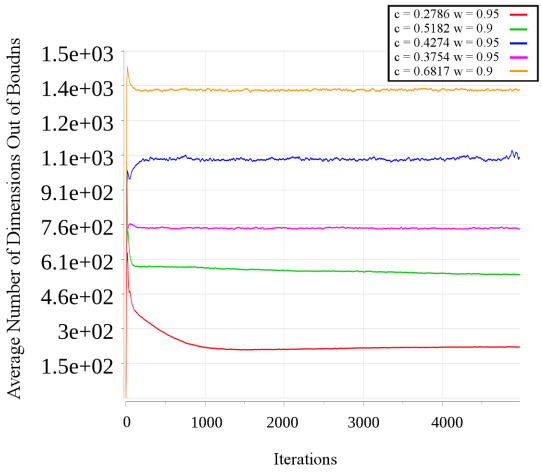


Figure 5.14: Average number of dimensions out of bounds on F18 with $n = 1000$ (5 best configurations)

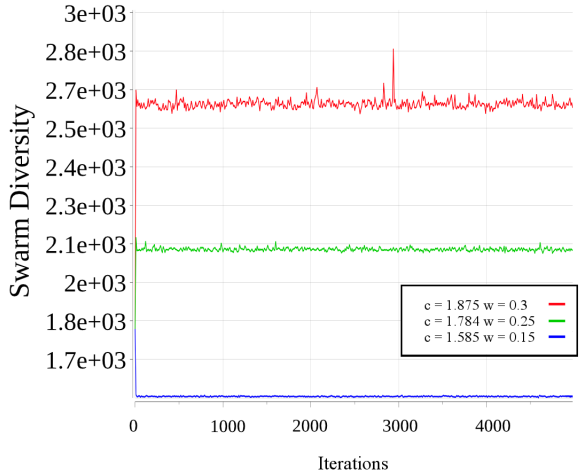


Figure 5.15: Swarm diversity of outlier configurations on F9 with n=1000

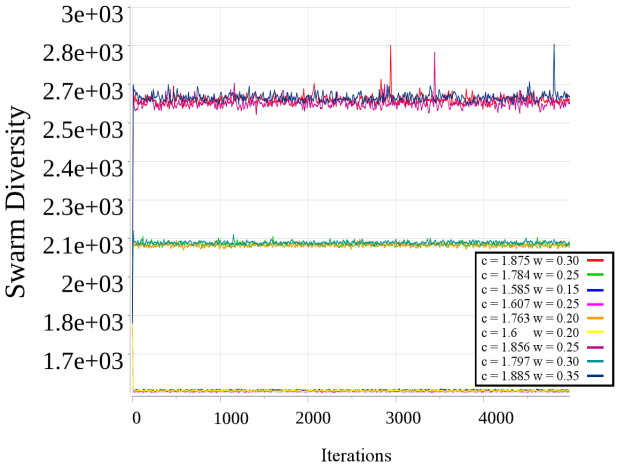


Figure 5.16: Swarm diversity of outlier configurations and surrounding configurations on F9 with n=1000

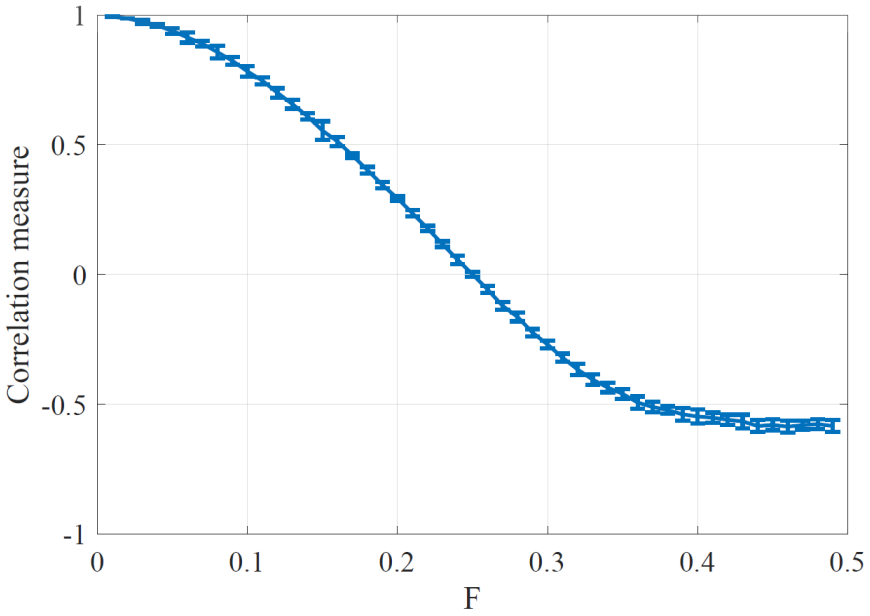


Figure 5.17: Relationship between base frequency and correlation of particle positions

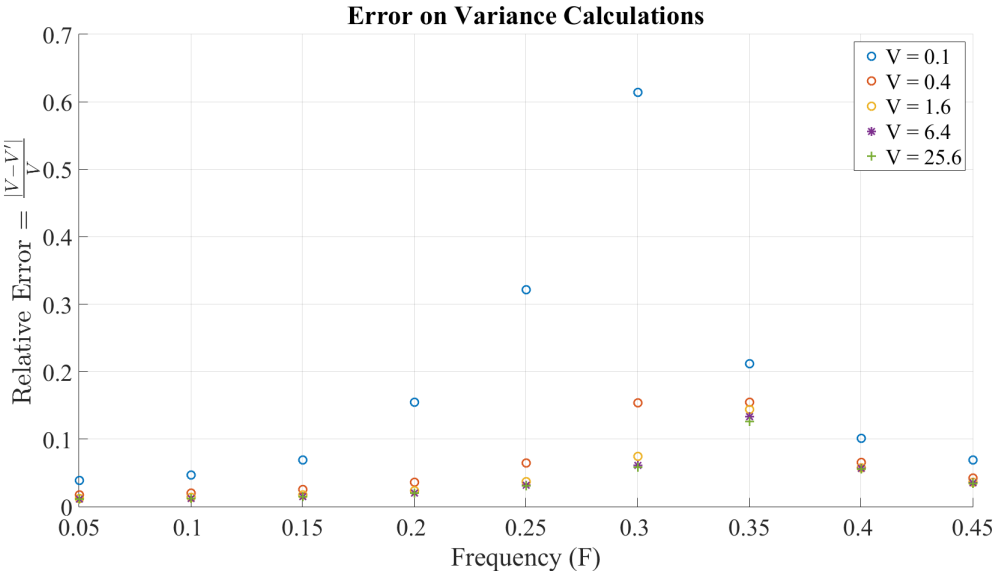


Figure 5.18: Relative error on variance across frequencies

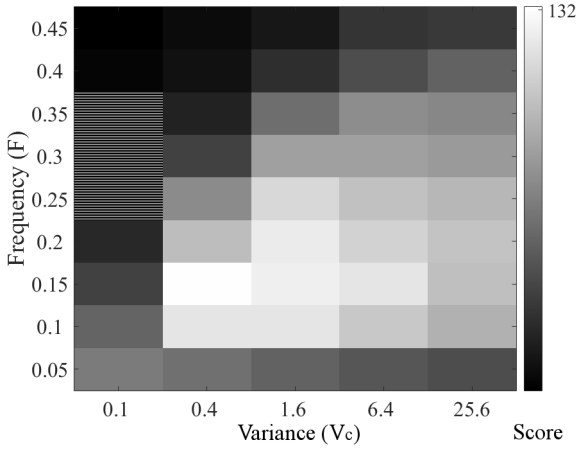


Figure 5.19: Optimal frequency-variance combinations (n=10)

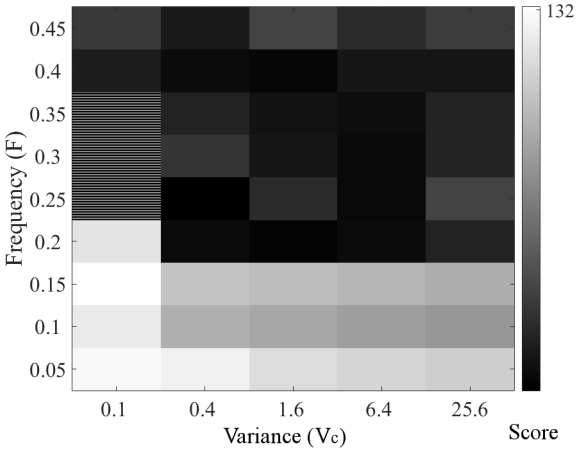


Figure 5.20: Optimal frequency-variance combinations (n=1000)

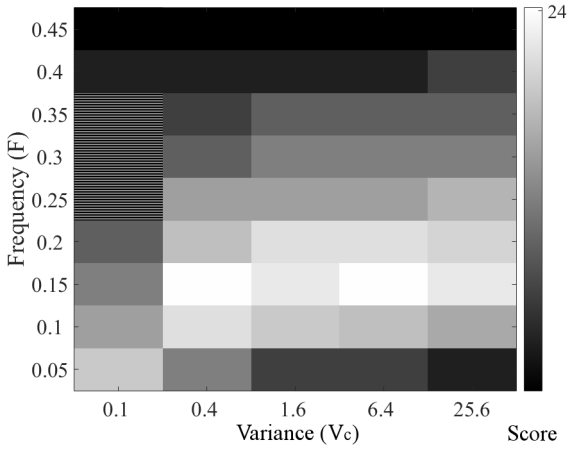


Figure 5.21: Optimal frequency for given variance (n=10)

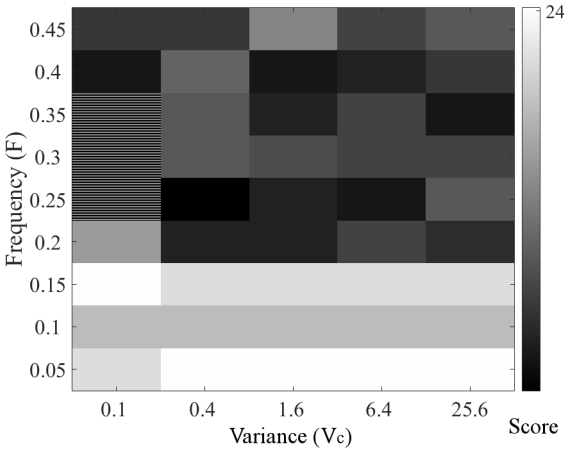


Figure 5.22: Optimal frequency for given variance (n=10)

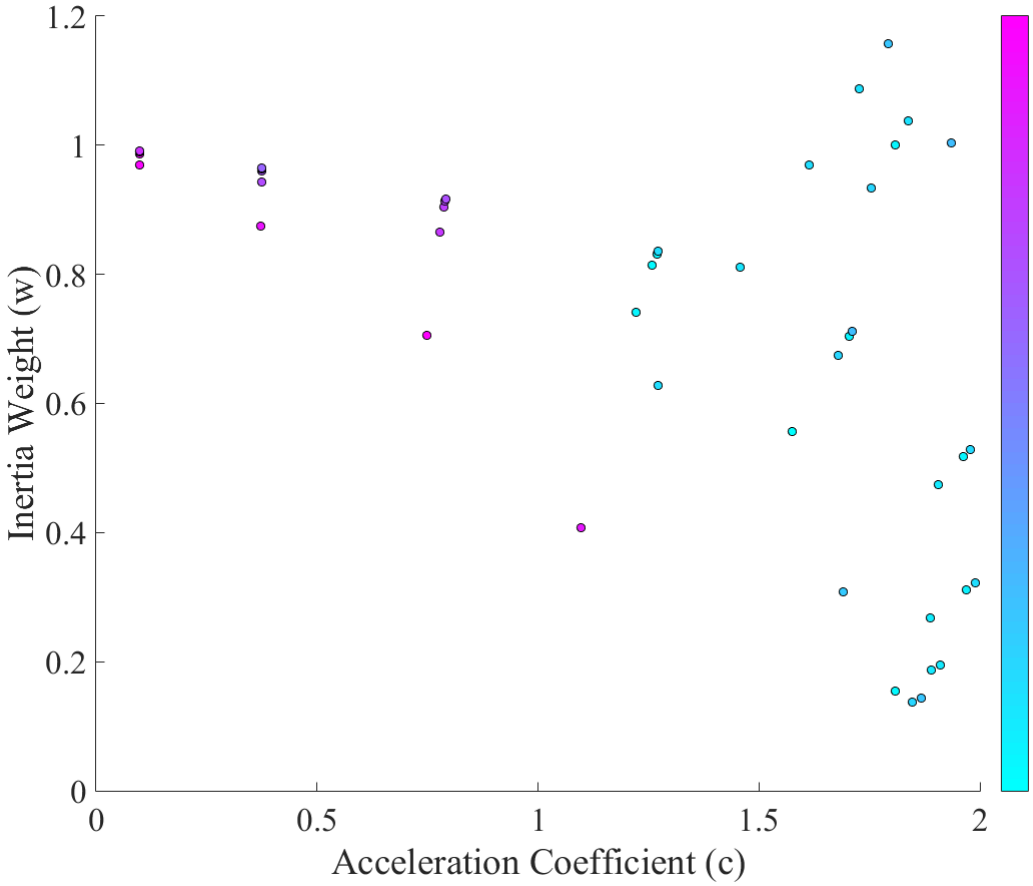


Figure 5.23: Score of acceleration coefficients and inertia weights produced by frequency and variance

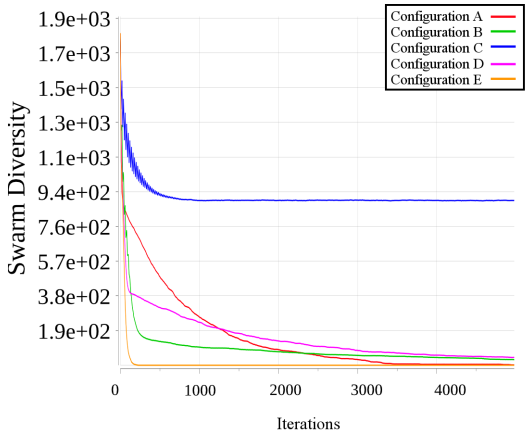


Figure 5.24: Swarm diversity on F7 with $n = 1000$ (best 5 configurations)

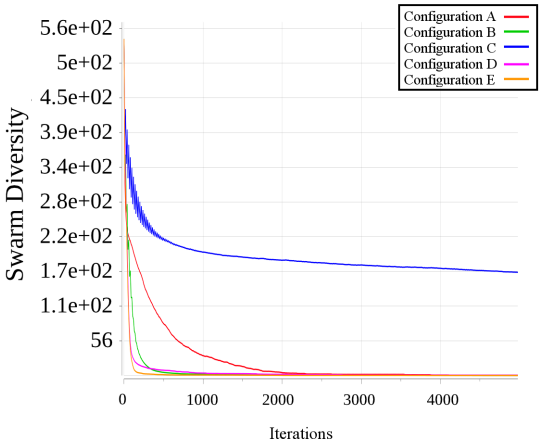


Figure 5.25: Swarm diversity on F11 with $n = 1000$ (best 5 configurations)

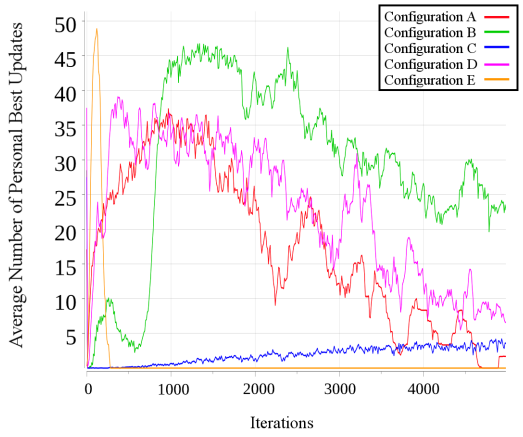


Figure 5.26: Average number of personal best updates on F7 with $n = 1000$ (best 5 configurations)

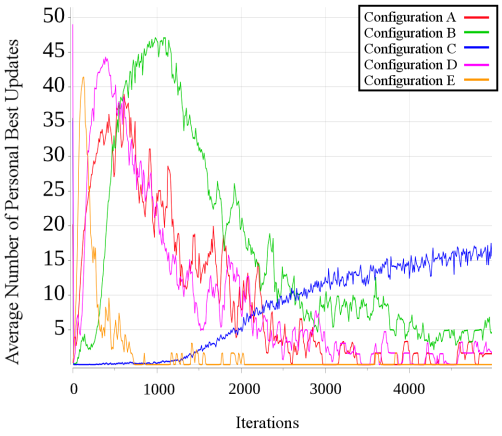


Figure 5.27: Average number of personal best updates on F11 with $n = 1000$ (best 5 configurations)

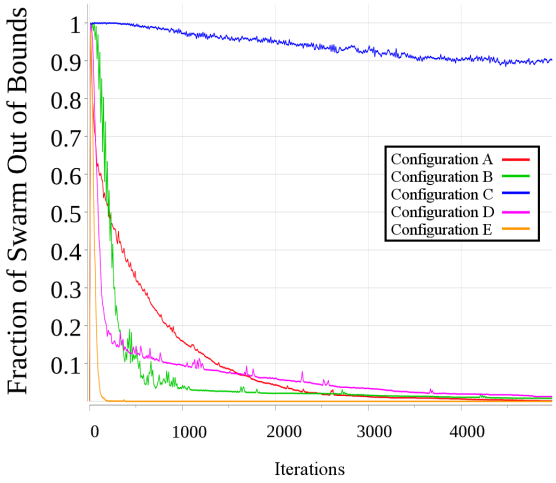


Figure 5.28: Fraction of swarm out of bounds on F8 with $n = 1000$ (best 5 configurations)

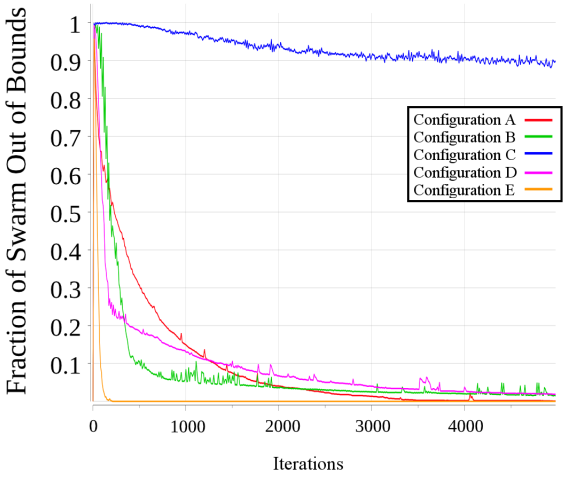


Figure 5.29: Fraction of of swarm out of bounds on F7 with $n = 1000$ (best 5 configurations)

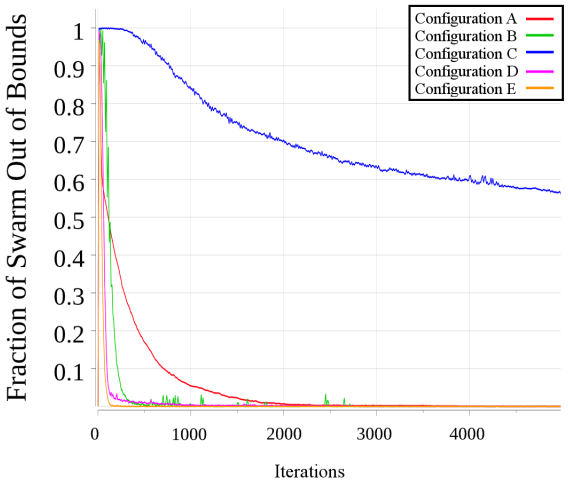


Figure 5.30: Fraction of of swarm out of bounds on F11 with $n = 1000$ (best 5 configurations)

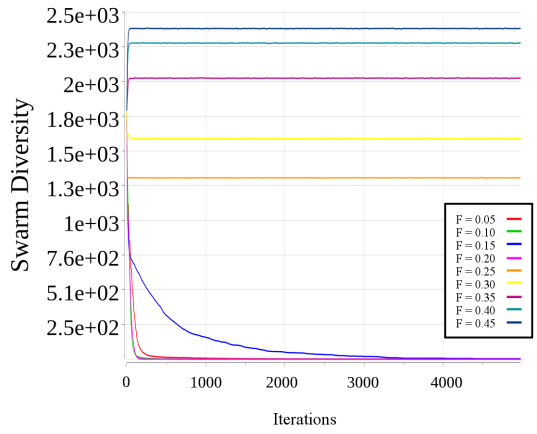


Figure 5.31: Swarm diversity when $V_c = 0.1$ on F17 with $n = 1000$

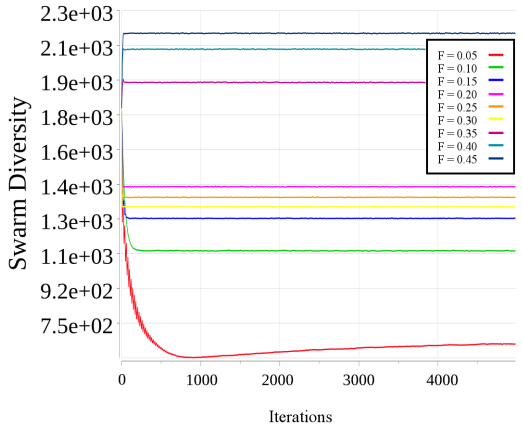


Figure 5.32: Swarm diversity when $V_c = 0.4$ on F17 with $n = 1000$

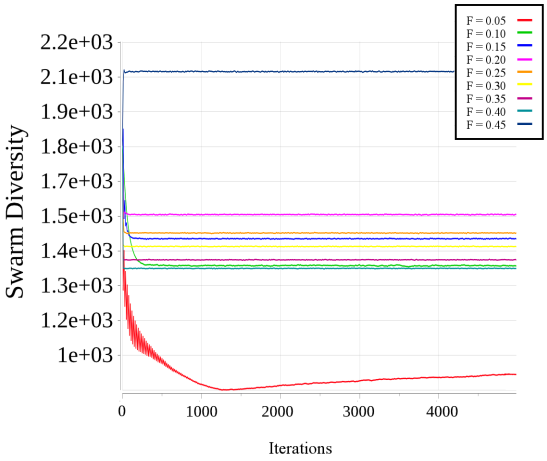


Figure 5.33: Swarm diversity when $V_c = 1.6$ on F17 with $n = 1000$

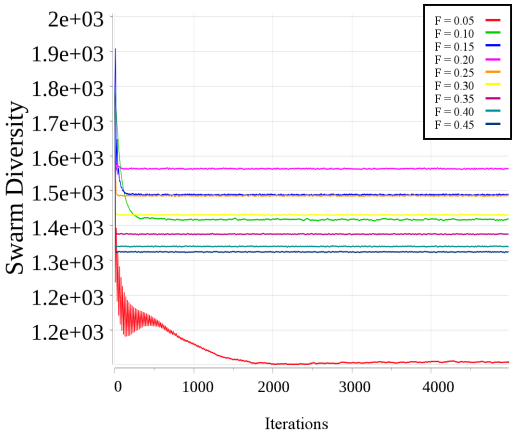


Figure 5.34: Swarm diversity when $V_c = 6.4$ on F17 with $n = 1000$

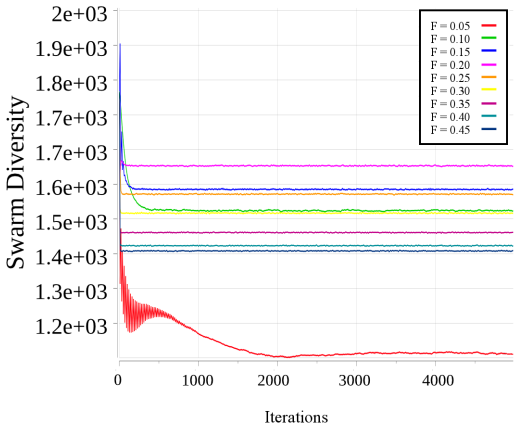


Figure 5.35: Swarm diversity when $V_c = 25.6$ on F17 with $n = 1000$

Chapter 6

Grouping Stochastic Scalars

This chapter examines the effect of the stochastic scaling components on the swarm's ability to explore and exploit the search space. The chapter begins by explaining the difference between scaling the social and cognitive components using component-wise multiplication of random vectors and using random scalars in section 6.1. Next, section 6.2 proposes a new movement strategy that changes the way in which random vectors are used to scale the global and local search information by grouping problem dimensions and applying the same random scaling values to components in the same group. Section 6.3 tests the grouping strategies against a standard inertia weight PSO and compares the behaviours brought about by each grouping strategy. Section 6.4 summarizes the observed results and ties it into the overarching discussion of PSO's behaviour in high dimensional search spaces.

6.1 Stochastic Scaling: Vector or Scalar

This section explains the difference between scaling the social and cognitive components using component-wise multiplication of random vectors and using random scalars. Section 6.1.1 provides an in-depth discussion and a theoretical proof regarding the limitations imposed on the swarm's movement when using r_1 and r_2 as random scalars. Section 6.1.2 empirically determines the extent to which the swarm is able to escape its initial subspace when using component-wise random scaling (where $\mathbf{r}_1$ and $\mathbf{r}_2$ are

vectors).

6.1.1 Scaling with Scalars

Consider Equation (2.7), the velocity update equation from Chapter 2. Suppose that instead of $\mathbf{r}_1$ and $\mathbf{r}_2$ being random vectors, r_1 and r_2 are random scalars sampled from a uniform distribution with range $(0, 1)$ as shown below:

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1 r_1(\mathbf{y}_i^t - \mathbf{x}_i^t) + c_2 r_2(\hat{\mathbf{y}}^t - \mathbf{x}_i^t) \tag{6.1}$$

The right hand side of Equation (6.1) can be rewritten in the standard form of a linear combination, as illustrated below:

$$\begin{aligned} \text{RHS} &= a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3 \\ \text{with } a_1 &= w \text{ and } \mathbf{z}_1 = \mathbf{v}_i^t \\ a_2 &= c_1 r_1 \text{ and } \mathbf{z}_2 = (\mathbf{y}_i^t - \mathbf{x}_i^t) \\ a_3 &= c_2 r_2 \text{ and } \mathbf{z}_3 = (\hat{\mathbf{y}}^t - \mathbf{x}_i^t) \end{aligned}$$

Thus, the velocity of any particle i at iteration $t + 1$ is simply a linear combination of its previous velocity, its social component and its cognitive component. In turn, the social and cognitive components can be expressed as a linear combination of the particle's position, its personal best position, and the global best position. A particle's velocity can thus be written as a linear combination of its previous velocity, personal best position, current position, and the global best position, as below:

$$\begin{aligned} \text{RHS} &= b_1\tilde{\mathbf{z}}_1 + b_2\tilde{\mathbf{z}}_2 + b_3\tilde{\mathbf{z}}_3 + b_4\tilde{\mathbf{z}}_4 \\ \text{with } b_1 &= w \text{ and } \tilde{\mathbf{z}}_1 = \mathbf{v}_i^t \\ b_2 &= c_1 r_1 \text{ and } \tilde{\mathbf{z}}_2 = \mathbf{y}_i^t \\ b_3 &= c_2 r_2 \text{ and } \tilde{\mathbf{z}}_3 = \hat{\mathbf{y}}^t \\ b_4 &= -(c_1 r_1 + c_2 r_2) \text{ and } \tilde{\mathbf{z}}_4 = \mathbf{x}_i^t \end{aligned}$$

The position attained by particle i is thus a linear combination of its previous position, previous velocity, personal best position, and the global best position as shown below:

$$\begin{aligned}\mathbf{x}_i^{t+1} &= \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \\ &= \mathbf{x}_i^t + b_1\tilde{\mathbf{z}}_1 + b_2\tilde{\mathbf{z}}_2 + b_3\tilde{\mathbf{z}}_3 + b_4\tilde{\mathbf{z}}_4 \\ &= b_1\tilde{\mathbf{z}}_1 + b_2\tilde{\mathbf{z}}_2 + b_3\tilde{\mathbf{z}}_3 + (b_4+1)\tilde{\mathbf{z}}_4\end{aligned}$$

Any position that can be attained by particle i depends on these four vectors. If the swarm's velocity and position initialization strategy is known, then it is possible to express exactly which values the particles' positions may attain, i.e. where the particles can and can not go within the search space. Particularly, the region of the search space that can be reached by the particles is contained in the span of their initial velocities, positions, and personal best positions. This has been proved in literature [54], but a brief proof by induction is also provided here. For completeness, linear dependence, linear independence and the *span* of a set of vectors are defined below.

Definition 6.1. A set of vectors $\mathcal{I} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_M\} \subset \mathbb{R}^n$ is *linearly dependent* if there exists a finite number of distinct vectors, $\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_K \in \mathcal{I}$, and scalars, $a_1, a_2, ..., a_K \in \mathbb{R}$, not all zero, such that

$$a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_K\mathbf{z}_K = \mathbf{0} \tag{6.2}$$

Since at least one scalar is non-zero, say $a_1 \neq 0$, the vector $\mathbf{z}_1$ can be expressed as a linear combination of the other vectors:

$$\mathbf{z}_1 = -\frac{a_2}{a_1}\mathbf{z}_2 - ... - \frac{a_K}{a_1}\mathbf{z}_K \tag{6.3}$$

Thus, the set $\mathcal{I}$ is *linearly dependent* if and only if at least one element in $\mathcal{I}$ can be written as a linear combination of the other elements in $\mathcal{I}$.

Definition 6.2. A set of vectors $\mathcal{I} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_M\} \subset \mathbb{R}^n$ is *linearly independent* if the equation,

$$a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_M\mathbf{z}_M = \mathbf{0} \tag{6.4}$$

can only be satisfied by $a_1 = a_2 = ... = a_M = 0$. Thus no element in $\mathcal{I}$ can be written as a linear combination of other elements from $\mathcal{I}$.

Definition 6.3. Let $\mathcal{I}$ be a non-empty set of vectors from $\mathbb{R}^n$ (i.e. $\emptyset \neq \mathcal{I} \subset \mathbb{R}^n$). Then the *span* $\mathcal{I}$ is the smallest subspace $W \subseteq \mathbb{R}^n$ that contains $\mathcal{I}$. Thus $span(\mathcal{I}) = W$. The subspace W consists of all linear combinations of elements of $\mathcal{I}$, given by

$$\text{span}(\mathcal{I}) = \left\{ \sum_{k=1}^{|\mathcal{I}|} a_k \mathbf{z}_k \;\middle|\; \mathbf{z}_k \in \mathcal{I}, a_k \in \mathbb{R}, k \in \mathbb{N} \right\} \tag{6.5}$$

where $|.|$ denotes set cardinality.

Definition 6.4. A non-empty set of vectors, $\mathcal{I} = \{\mathbf{z}_1, ..., \mathbf{z}_M\}$, is a spanning set for a subspace $W \subseteq \mathbb{R}^n$ if and only if any element in W can be expressed as a linear combination of elements in $\mathcal{I}$. In other words, for any $\mathbf{z} \in W$, there exist scalars $a_1, a_2, ..., a_M$ with at least one $a_i \neq 0$ such that

$$\mathbf{z} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_M\mathbf{z}_M \tag{6.6}$$

Definition 6.5. A *basis* for a subspace S of $\mathbb{R}^n$ is a set of vectors $\mathcal{B}$ in S such that

- $\mathcal{B}$ is a spanning set for S, and
- $\mathcal{B}$ is linearly independent.

Armed with these definitions, it is proven below that if r_1 and r_2 are scalars, then the positions of any particle at any iteration of the search space must be a linear combination of their initial positions, best positions, and velocities. The theorem below is for an *lBest PSO*, since a *gBest PSO* is simply a local best PSO where the neighbourhood is the entire swarm.

Theorem 6.1. *At any iteration $t \geq 0$, the position $\mathbf{x}_i^t$ of any particle i is in the span of $\mathcal{I}$ where $\mathcal{I} = \{\mathbf{x}_1^0, \mathbf{y}_1^0, \mathbf{v}_1^0, ..., \mathbf{x}_{n_s}^0, \mathbf{y}_{n_s}^0, \mathbf{v}_{n_s}^0\}$.*

Proof. Suppose that particle velocities, positions and personal best positions are initialized randomly within the search space. Let the set of all these initial points be given by $\mathcal{I} = \{\mathbf{x}_1^0, \mathbf{y}_1^0, \mathbf{v}_1^0, ..., \mathbf{x}_{n_s}^0, \mathbf{y}_{n_s}^0, \mathbf{v}_{n_s}^0\}$. Assume that all the elements in $\mathcal{I}$ are unique and nonzero. These are reasonable assumptions: the probability of obtaining a zero vector from uniform initialization is zero, since the probability of a continuous random variable

being a particular constant is zero. Similarly, the probability of sampling two equal vectors is zero because the set of such points have zero measure.

Since the position of any particle at $t = 0$ is in $\mathcal{I}$ by the definition of $\mathcal{I}$, the position of any particle is also in the span of $\mathcal{I}$. Thus the hypothesis holds for the case $t = 0$.

At iteration $t = 1$, the position of any particle i is given by

$$\mathbf{x}_i^1 = \mathbf{x}_i^0 + \mathbf{v}_i^1 \tag{6.7}$$

Since $\mathbf{x}_i^0 \in \mathcal{I}$, it is only necessary to prove that $\mathbf{v}_i^1 \in \mathit{span}(\mathcal{I})$ for $\mathbf{x}_i^1$ to be in the span of $\mathcal{I}$. According to the velocity update equation,

$$\mathbf{v}_i^1 = w\mathbf{v}_i^0 + c_1 r_1(\mathbf{y}_i^0 - \mathbf{x}_i^0) + c_2 r_2(\hat{\mathbf{y}}_i^0 - \mathbf{x}_i^0) \tag{6.8}$$

$$= w\mathbf{v}_i^0 + c_1 r_1 \mathbf{y}_i^0 + c_2 r_2 \hat{\mathbf{y}}_i^0 - (r_1 c_1 + r_2 c_2)\mathbf{x}_i^0 \tag{6.9}$$

By definition, $\mathbf{v}_i^0, \mathbf{y}_i^0, \mathbf{x}_i^0 \in \mathcal{I}$. Additionally, the neighbourhood best position is chosen from among the personal best positions of the other particles in the neighbourhood, so that $\hat{\mathbf{y}}_i^0 \in \mathcal{I}$. Thus, if particle i is not the neighbourhood best, then $\mathbf{v}_i^1$ is a linear combination of four distinct elements from $\mathcal{I}$. If particle i is the neighbourhood best, then $\mathbf{y}_i^0 = \hat{\mathbf{y}}_i^0$ and $\mathbf{v}_i^1$ is a linear combination of three distinct elements from $\mathcal{I}$. In either case, $\mathbf{v}_i^1 \in \text{span}(\mathcal{I})$ by definition 6.3. The fact that $\mathbf{v}_i^1 \in \text{span}(\mathcal{I})$ will be referred to as (*). Therefore, since $\mathbf{x}_i^1$ is the sum of two elements in $\mathit{span}(\mathcal{I})$, $\mathbf{x}_i^1$ is in the span of $\mathcal{I}$. Since this is true for any particle i, all the particles' positions at iteration i must be in the span of $\mathcal{I}$ - this fact will be referred to as (**).

At iteration $t = 2$, the position of any particle i is given by

$$\mathbf{x}_i^2 = \mathbf{x}_i^1 + \mathbf{v}_i^2 \tag{6.10}$$

where $\mathbf{x}_i^1 \in \mathit{span}(\mathcal{I})$ as proved for the case when $t = 1$. It thus remains to prove that $\mathbf{v}_i^2$ is in the span of $\mathcal{I}$:

$$\mathbf{v}_i^2 = w\mathbf{v}_i^1 + c_1 r_1(\mathbf{y}_i^1 - \mathbf{x}_i^1) + c_2 r_2(\hat{\mathbf{y}}_i^1 - \mathbf{x}_i^1) \tag{6.11}$$

$$= w\mathbf{v}_i^1 + c_1 r_1 \mathbf{y}_i^1 + c_2 r_2 \hat{\mathbf{y}}_i^1 - (r_1 c_1 + r_2 c_2)\mathbf{x}_i^1 \tag{6.12}$$

where $\mathbf{v}_i^1 \in \mathit{span}(\mathcal{I})$ by (*) and $\mathbf{x}_i^1 \in \mathit{span}(\mathcal{I})$ by (**). It is thus only necessary to prove that $\mathbf{y}_i^1$ and $\hat{\mathbf{y}}_i^1$ are in the span of $\mathcal{I}$.

Between iterations 0 and 1, the personal best position either updated or did not update. If the personal best position did not update, then $\mathbf{y}_i^1 = \mathbf{y}_i^0 \in \mathcal{I}$. If the personal best position did update, then $\mathbf{y}_i^1 = \mathbf{x}_i^1$, since that is the only other position that the particle has encountered. By (**), $\mathbf{x}_i^1 \in \mathit{span}(\mathcal{I})$. Thus, whether the personal best position updated or not, $\mathbf{y}_i^1 \in \mathit{span}(\mathcal{I})$.

Similarly, the neighbourhood best position may or may not have updated. If it did not update, then $\hat{\mathbf{y}}_i^1 = \hat{\mathbf{y}}_i^0 \in \mathcal{I}$. If the neighbourhood best position did update, then there must be a particle j in i's neighbourhood so that $\hat{\mathbf{y}}_i^1 = \mathbf{x}_j^1$. But by (**), any particle's position at iteration $t = 1$ was in the span of $\mathcal{I}$. Thus $\hat{\mathbf{y}}_i^1$ must be in the span of $\mathcal{I}$.

Therefore, $\mathbf{v}_i^2$ is a linear combination of elements in the span of $\mathcal{I}$, so $\mathbf{v}_i^2$ must be in the span of $\mathcal{I}$.

Suppose for all iterations $s \leq t$, that the positions of all the particles are in the span of $\mathcal{I}$. It will now be proved that the positions of all the particles must still be in the span of $\mathcal{I}$ at iteration $t + 1$. The argument is similar to the case for $t = 2$. The position of any particle i is given by the position update equation:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{6.13}$$

where $\mathbf{x}_i^t \in \mathit{span}(\mathcal{I})$ by virtue of the inductive assumption. It thus remains to prove that $\mathbf{v}_i^{t+1}$ is in the span of $\mathcal{I}$:

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1 r_1(\mathbf{y}_i^t - \mathbf{x}_i^t) + c_2 r_2(\hat{\mathbf{y}}_i^t - \mathbf{x}_i^t) \tag{6.14}$$

$$= w\mathbf{v}_i^t + c_1 r_1 \mathbf{y}_i^t + c_2 r_2 \hat{\mathbf{y}}_i^t - (r_1 c_1 + r_2 c_2)\mathbf{x}_i^t \tag{6.15}$$

where $\mathbf{x}_i^t \in \mathit{span}(\mathcal{I})$ by the inductive assumption. It thus remains to prove that $\mathbf{v}_i^t$, $\mathbf{y}_i^t$ and $\hat{\mathbf{y}}_i^t$ are in the span of $\mathcal{I}$.

According to the position update equation,

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t \tag{6.16}$$

$$\implies \mathbf{v}_i^t = \mathbf{x}_i^t - \mathbf{x}_i^{t-1} \tag{6.17}$$

In other words, $\mathbf{v}_i^t$ is a linear combination of $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t-1}$, both of which are elements in the span of $\mathcal{I}$ by the inductive assumption. Thus, $\mathbf{v}_i^t$ is also in the span of $\mathcal{I}$. The personal best position of any particle i can only be equal to one of the particle's previous

positions. But, by the inductive assumption, all the particle's previous positions were in the span of $\mathcal{I}$. Thus, $\mathbf{y}_i^t$ must be in the span of $\mathcal{I}$. Similarly, the neighbourhood best position must be equal to a previous position of some particle in i's neighbourhood, all of which are in the span of $\mathcal{I}$ by the inductive assumption. Therefore, the velocity and also the position of particle i at iteration $t+1$ must be in $span(\mathcal{I})$.

□

Theorem 6.1 implies that, when r_1 and r_2 are random scalar values, the positions of the particles are limited to be in the span of their initial velocities, positions and personal best positions. If either of the assumptions on $\mathcal{I}$ does not hold (e.g. some vectors are multiples of another or some are zero), then the positions of the particles are limited further to being linear combinations of all non-zero, linearly independent initial velocities, positions and personal best positions. The question arises whether any point in the search space can be expressed in terms of such linear combinations.

If $\mathcal{I}$ constitutes a spanning set for the search space (i.e. the span of $\mathcal{I}$ is larger than the search space or equal to), then any point in the search space can theoretically be reached by the particles. However, if $\mathcal{I}$ is not a spanning set of the search space (i.e. the span of $\mathcal{I}$ is a subspace within the search space), then the particles can not reach every position in the search space. If the global optimum happens to be outside the span of $\mathcal{I}$, then the particles will never be able to find it. Since initial particle positions and personal best positions are typically generated randomly, this is a realistic scenario in high dimensional spaces.

It is a fundamental result of linear algebra that any two bases for some subspace of $\mathbb{R}^n$ must contain the same number of vectors [57]. Observe that if the search space, S, is a subset of $\mathbb{R}^n$, then the basic unit vectors

$$\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n =< 1, 0, 0, ..., 0 >^T, < 0, 1, 0, 0, ... >^T, ..., < 0, ..., 0, 1 >^T \tag{6.18}$$

form a basis for S. Thus, for $\mathcal{I}$ to span S, it must contain at least n linearly independent vectors. However, the search space $S = [L, U]^n$ does not constitute a subspace of $\mathbb{R}^n$, since it is not closed under addition and scalar multiplication. Theorem 6.4 proves the required result for this special case. The proof requires the use of the fundamental

theorem of invertible matrices [57] (which is given below) and the result is given in lemma 6.3.

Theorem 6.2 (Fundamental Theorem of Invertible Matrices)**.** *Let A be an n-by-m matrix. Then the following statements are equivalent:*

1. *A is invertible*
2. *$A\mathbf{x} = \mathbf{0}$ has only the trivial solution*
3. *The column vectors of A are linearly independent*
4. *The column vectors of A span $\mathbb{R}^n$*

Lemma 6.3. *Let $S = [L, U]^n$. There exists a set $\tilde{E}$ that forms a "basis" for S, in the sense that $\tilde{E}$'s elements are linearly independent, any $\mathbf{x} \in S$ can be expressed as a linear combination of elements in S and $\tilde{E} \subset S$.*

Proof. Let the center of the search space be denoted by

$$\mathbf{M} =< m_1, m_2, ..., m_n >^T = \left\langle \frac{L+U}{2}, \frac{L+U}{2}, ..., \frac{L+U}{2} \right\rangle^T \tag{6.19}$$

Since the search space is the same in every dimension, $\mathbf{M} =< m, m, ..., m >^T$ where $m = \frac{L+U}{2}$.

Let $\tilde{E} = \{\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, ..., \tilde{\mathbf{e}}_n\}$ where

$$\begin{aligned} \tilde{\mathbf{e}}_1 &=< m + 0.5m, m, ..., m >^T \\ \tilde{\mathbf{e}}_2 &=< m, m + 0.5m, ..., m >^T \\ &... \\ \tilde{\mathbf{e}}_n &=< m, m, ..., m + 0.5m >^T \end{aligned} \tag{6.20}$$

so that for any $\tilde{\mathbf{e}}_i$, all coordinates except the i-th coordinate are equal to m. The i-th coordinate will be equal to $1.5m$. Clearly, $\tilde{E} \subset S$. Let A be the n by n matrix with column vectors $\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, ..., \tilde{\mathbf{e}}_n$.

We prove that $A\mathbf{x} = \mathbf{0}$ has only the trivial solution. Then, by theorem 6.2, it will follow that:

1. A's column vectors are linearly independent. In other words, $\tilde{E}$ is linearly independent.

2. The column vectors of A span $\mathbb{R}^n$, which means that E spans $\mathbb{R}^n$. Since $S \subset \mathbb{R}^n$, any element in S can thus be expressed as a linear combination of elements from E.

It will now be proved that $A\mathbf{x} = \mathbf{0}$ has only the trivial solution, thereby proving the required properties of $\tilde{E}$. The system in $A\mathbf{x} = \mathbf{0}$ can be written as a system of linear equations as illustrated below:

$$\begin{array}{rcrcccrcl} (m+0.5m)x_1 & + & mx_2 & + & ... & + & mx_n & = & 0 \\ mx_1 & + & (m+0.5m)x_2 & + & ... & + & mx_n & = & 0 \\ & & & \vdots & & & & & \\ mx_1 & + & mx_2 & + & ... & + & (m+0.5m)x_n & = & 0 \end{array} \tag{6.21}$$

were the j-th equation is formed from the j component of the system $A\mathbf{x} = \mathbf{0}$. In turn, the system of linear equations can be represented in augmented matrix form as below:

$$\begin{pmatrix} 1.5m & m & m & ... & m & 0 \\ m & 1.5m & m & ... & m & 0 \\ ... & ... & ... & ... & ... & ... \\ m & m & m & ... & 1.5m & 0 \end{pmatrix} \tag{6.22}$$

where the i-th column contains the coefficients of x_i. The final column contains the right hand side of all the equations in the linear system (in this case, the zero vector). The augmented matrix is manipulated by means of matrix row operations, which take one of the following forms:

1. Switch the i-th and j-th rows (denoted by $R_i \leftrightarrow R_j$, where R_i denotes the i-th row).

2. Multiply the i-th row by c, a non-zero constant (denoted by $R_i \rightarrow cR_i$).

3. Add row j to row i (denoted by $R_i \rightarrow R_i + R_j$).

From these basic operations, more complex operations can be constructed such as subtracting one row from another ($R_i \to R_i - R_j$). For the sake of brevity, the notation $\forall_i R_i \to cR_i$ denotes that for each row i, the row is multiplied by a constant c. Similarly, $\forall_i R_i \to R_i - R_j$ indicates that the j-th row is subtracted from each of the other rows in turn. The system $A\mathbf{x} = \mathbf{0}$ will now be solved below.

$$\left(\begin{array}{ccccc|c} 1.5m & m & m & ... & m & 0 \\ m & 1.5m & m & ... & m & 0 \\ ... & ... & ... & ... & ... & ... \\ m & m & m & ... & 1.5m & 0 \end{array}\right) \forall_i R_i \to \frac{1}{m} R_i \left(\begin{array}{ccccc|c} 1.5 & 1 & 1 & ... & 1 & 0 \\ 1 & 1.5 & 1 & ... & 1 & 0 \\ ... & ... & ... & ... & ... & ... \\ 1 & 1 & 1 & ... & 1.5 & 0 \end{array}\right)$$

$$\forall_i R_i \to R_i - R_n \left(\begin{array}{ccccc|c} 0.5 & 0 & 0 & ... & -0.5 & 0 \\ 0 & 0.5 & 0 & ... & -0.5 & 0 \\ ... & ... & ... & ... & ... & ... \\ 1 & 1 & 1 & ... & 1.5 & 0 \end{array}\right) \forall_i R_i \to 2R_i \left(\begin{array}{ccccc|c} 1 & 0 & 0 & ... & -1 & 0 \\ 0 & 1 & 0 & ... & -1 & 0 \\ ... & ... & ... & ... & ... & ... \\ 1 & 1 & 1 & ... & 1.5 & 0 \end{array}\right)$$

$$R_n \to R_n - R_1 - R_2 - ... - R_{n-1} \left(\begin{array}{ccccc|c} 1 & 0 & 0 & ... & -1 & 0 \\ 0 & 1 & 0 & ... & -1 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 1.5 - (-1)(n-1) & 0 \end{array}\right)$$

Therefore, $x_n(n + 0.5) = 0$. Since $n \geq 1$, the equation can only have the trivial solution, $x_n = 0$. But then, for every $i = 1, ..., n - 1$,

$$\begin{aligned} & x_i - x_n = 0 \\ \implies & x_i - 0 = 0 \\ \implies & x_i = 0 \end{aligned} \tag{6.23}$$

Therefore, $A\mathbf{x} = \mathbf{0}$ has only the trivial solution. The existence of the required set $\tilde{E}$ is thus proved. □

Theorem 6.4. *Suppose $\mathcal{I}$ contains m linearly independent vectors and $S = [L, U]^n$. If $m < n$ then $span(\mathcal{I}) \cap S \subsetneq S$. Thus $\mathcal{I}$ can only be a spanning set of S if it contains at least n linearly independent elements.*

Proof. By lemma 6.3, there exists a set $\tilde{E}$ that forms a "basis" for S. The set $\tilde{E}$ will

be used to prove that if $\mathcal{I}$ contains m linearly independent vectors and $m < n$, then $\mathcal{I}$ spans only a subset of S.

Let $m < n$. Towards a contradiction, suppose that $\mathcal{I}$ and $\tilde{E}$ both form "bases" for S. In other words, $span(\mathcal{I}) \cap S = span(\tilde{E}) \cap S = S$.
Consider the equation,

$$c_1\tilde{\mathbf{e}}_1 + c_2\tilde{\mathbf{e}}_2 + ... + c_n\tilde{\mathbf{e}}_n = 0 \tag{6.24}$$

where $\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, ..., \tilde{\mathbf{e}}_n \in \tilde{E}$ and $c_1, c_2, ..., c_n \in \mathbb{R}$. Since each $\tilde{\mathbf{e}}_i \in S$ and any element in S can be expressed as a linear combination of vectors in $\mathcal{I}$, each $\tilde{\mathbf{e}}_i$ can be written as

$$\begin{aligned}
\tilde{\mathbf{e}}_1 &= a_{11}\mathbf{z}_1 + a_{12}\mathbf{z}_2 + ... + a_{1m}\mathbf{z}_m \\
&\vdots \\
\tilde{\mathbf{e}}_i &= a_{i1}\mathbf{z}_1 + a_{i2}\mathbf{z}_2 + ... + a_{im}\mathbf{z}_m \\
&\vdots \\
\tilde{\mathbf{e}}_n &= a_{n1}\mathbf{z}_1 + a_{n2}\mathbf{z}_2 + ... + a_{nm}\mathbf{z}_m
\end{aligned}$$

Thus, equation (6.24) can be rewritten as follows:

$$\begin{aligned}
& c_1(a_{11}\mathbf{z}_1 + a_{12}\mathbf{z}_2 + ... + a_{1m}\mathbf{z}_m) + ... \\
&\qquad + c_i(a_{i1}\mathbf{z}_1 + a_{i2}\mathbf{z}_2 + ... + a_{im}\mathbf{z}_m) + ... \\
&\qquad + c_n(a_{n1}\mathbf{z}_1 + a_{n2}\mathbf{z}_2 + ... + a_{nm}\mathbf{z}_m) \qquad (6.25) \\
&= (c_1a_{11} + c_2a_{21} + ... + c_na_{n1})\mathbf{z}_1 + ... \\
&\qquad + (c_1a_{1j} + c_2a_{2j} + ... + c_na_{nj})\mathbf{z}_j + ... \\
&\qquad + (c_1a_{1m} + c_2a_{2m} + ... + c_na_{nm})\mathbf{z}_m \\
&= \sum_{i=1}^{n} c_ia_{i1}\mathbf{z}_1 + \sum_{i=1}^{n} c_ia_{i2}\mathbf{z}_2 + ... + \sum_{i=1}^{n} c_ia_{im}\mathbf{z}_m \qquad (6.26)
\end{aligned}$$

Now, $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_m$ are linearly independent. Therefore, equation (6.26) has only the trivial solution and $\sum_{i=1}^{n} c_ia_{ij}$ must equal zero for all $j = 1, ..., m$. This can be written as a homogeneous system of m equations, each with n variables, $c_1, ..., c_n$. Since $m < n$, there are more variables than equations, so there must be infinitely many solutions. Particularly, there must be a non-trivial solution. But, this gives a non-trivial dependence

relation in equation (6.24). By definition, $\tilde{E}$ must thus be a linearly dependent set of vectors. But this is a contradiction, since $\tilde{E}$ is linearly independent. Therefore, $\mathcal{I}$ spans a strict subset of S. □

Theorem 6.4 proves that if r_1 and r_2 are scalars and the optimization function is high dimensional, then particles may not be able to reach the optimum solution because the solution can not even be expressed in terms of their initial positions, personal best positions, and velocities. If velocities are initialized to zero and the personal best positions are initialized to be equal to the initial positions, then the portion of the search space that can be reached by the particles is even smaller. It should also be noted that it is possible for the swarm to lose degrees of freedom throughout the search. Since the algorithm is executed on a computer with limited precision, it may occur that some of the vectors in $\mathcal{I}$ are cancelled out further in the search. Though unlikely, the span of the swarm may in fact decrease as the search progresses. Thus, if the size of the swarm is much smaller than the dimensionality of the search space, then the swarm will be unable to reach a large part of the search space. Unfortunately, simply increasing the number of particles in the swarm is not an adequate solution, because it greatly increases the computational cost. Additionally, the swarm size parameter influences the swarm's searching behaviour in other ways so changing the swarm size drastically may have unintended consequences [25, 47].

Using scalar values for r_1 and r_2 should thus only be done if it is known that the optimal solution lies in the span of the particles' initial positions. In fact, scalar values for r_1 and r_2 have been successfully employed to solve problems with equality constraints in the form $A\mathbf{x} = \mathbf{b}$ [54]. The particles are initialized in the feasible region and since r_1 and r_2 are scalars, the particles can not leave the hyperplane in which they were initialized and are guaranteed to produce feasible solutions.

6.1.2 Component-Wise Scaling with Vectors

The previous section explained the repercussions of choosing scalar values for r_1 and r_2. If, instead, the social and cognitive terms are scaled by vectors of random values in a component-wise manner, the swarm is able to escape its initial subspace and explore the

remainder of the search space [24].

Although it is theoretically possible for particles to reach any point in the search space when $\mathbf{r}_1$ and $\mathbf{r}_2$ are vectors, it may be unclear that the non-linearity introduced by stochastic vector scaling is sufficient for the particles to effectively express any solution. Chen *et. al.* [10] found that although particles do leave the search space within which they are initialized, the particles focus their search on the subspace within the initial subspace.

In order to measure a particle's activity inside and outside the initial subspace, Chen *et. al.* suggested projecting the particle's velocity vector onto two components: one component inside the initial subspace and the other component orthogonal to the initial subspace. This gives an idea of how much the positions change from one iteration to the next inside the initial subspace in comparison to outside the initial subspace. The result of the projection is then scaled according to the number of linearly independent vectors that formed each subspace. The experiments from [10] are reproduced here, for the PSO settings used throughout the thesis (see Chapter 3). A particle's step size within the initial subspace is measured by

$$S_I = \frac{\|P\mathbf{v}_i^t\|}{\sqrt{|\mathcal{I}|}} \tag{6.27}$$

where P is a projection matrix onto the initial subspace. P is given by

$$P = AA^T \tag{6.28}$$

where A is an n-by-$\|\mathcal{I}\|$ matrix, whose columns form an orthonormal basis for the initial subspace. The orthonormal basis may be obtained by applying the Gram-Schmidt method to $\mathcal{I}$, the set of initial particle positions, velocities, and personal best positions. Note that A^T denotes the transpose of the matrix A. Further discussion of the Gram-Schmidt method is deferred to section 7.2 in Chapter 7.

A particle's step outside the initial subspace is given by

$$S_O = \frac{\|P'\mathbf{v}_i^t\|}{\sqrt{n - |\mathcal{I}|}} \tag{6.29}$$

where P' is a projection matrix onto the space orthogonal to the initial subspace. P' may be obtained by finding an orthonormal basis for the initial subspace's orthogonal

complement and applying equation (6.28). The step size within a given subspace is normalized by the square root of the number of basis vectors for that subspace ($(\sqrt{|\mathcal{I}|}$ for the initial space and $\sqrt{n - |\mathcal{I}|}$ for the space orthogonal to the initial space). This allows direct comparison between S_I and S_O to determine how much movement took place in dimensions inside and outside the initial subspace. The square root was used because distance (i.e. step size) will grow proportionally to $\sqrt{k}$ as k increases, where k is the number of dimensions in the space. Note that the original literature did not specify how the scaling should be performed, but later results indicate that scaling was proportional to k (instead of $\sqrt{k}$). The measures proposed in equations (6.27) and (6.29) are thus slight alterations of the original measures proposed in [10].

The results shown below are not consistent with findings in literature, which found that step sizes inside the initial subspace are much larger than step sizes outside the initial subspace. In figure 6.1, the step sizes inside and outside the search space are both illustrated for a single run of a standard inertia weight PSO. Although the step size inside the initial space is very large for the first few iterations, the step size decreases sharply and after the first 50 iterations, the step sizes S_I and S_O are nearly identical for the remainder of the search.

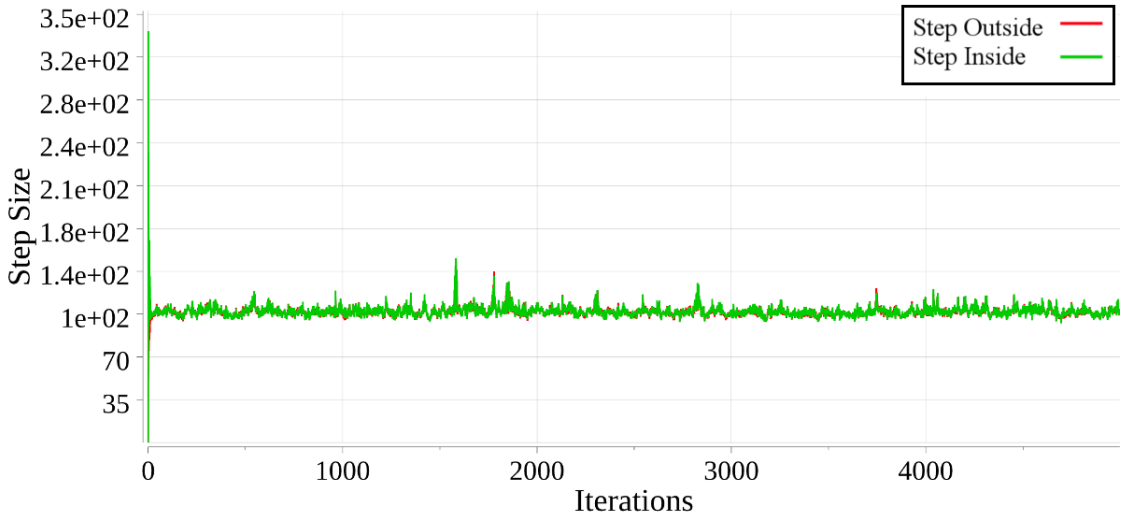


Figure 6.1: Step size inside initial subspace vs step size outside initial subspace of standard PSO on F17

The difference between the findings presented here and the results of [10] is likely due to scaling. The manner in which the projections were scaled was not specified in [10] and it may be that the step sizes were scaled by k instead of $\sqrt{k}$. This would unfairly penalize the larger dimensional space. In this case, the space orthogonal to the initial subspace is much larger than the initial subspace. Scaling by $n - |\mathcal{I}|$ would thus create the impression that movement in the space orthogonal to the initial subspace is much less than it actually is, because the step sizes were scaled by a number that is too large. Indeed, upon rescaling the step sizes proportionally to k instead of $\sqrt{k}$ (where k is the number of dimensions in the space, as before), then figure 6.2 is obtained, which is consistent with literature.

Figure 6.2: Step size inside initial subspace vs step size outside initial subspace of standard PSO on F17 (incorrect scaling)

Figure 6.2 incorrectly implies that the step size inside the initial subspace is consistently about 2 to 3 times that outside the initial subspace (across all the other benchmark functions). However, as figure 6.1 showed, this is not the case. The swarm focuses almost equally on the initial subspace and the subspace orthogonal to the initial subspace. The step size measure described in this section will be applied again to examine the behaviour of the grouping strategies in later sections.

6.2 Grouping Stochastic Scalars

As discussed in Chapter 2, the search process can be divided into two different stages: exploration and exploitation. During *exploration*, the swarm attempts to cover large parts of the search space in order to find good regions and, hopefully, avoid becoming trapped in a local optimum. After the swarm has explored sufficiently, it should focus more on *exploitation* and search in a more fine-grained fashion so that the accuracy of the solution may be improved. It is important to balance exploration with exploitation to ensure that the PSO does not become trapped in local optima, but does still manage to find an accurate solution.

The choice of using scalars or vectors for r_1 and r_2 affects the exploration to exploitation ratio. If r_1 and r_2 are scalars, the particles are forced to remain inside the subspace in which they were initialized, effectively limiting the swarm's exploration ability and tilting the scale towards exploitation. On the other hand, component-wise vector scaling allows particles to roam the search space freely, thereby placing more emphasis on exploration than exploitation. The search space grows exponentially with problem dimensionality, so at high dimensions it becomes impossible for the swarm to explore the search space effectively. It may be more effective to bias the swarm towards exploitation rather than exploration, thereby forcing the swarm to perform granular searching on a smaller area rather than speed around the search space in search of a global solution.

This section introduces a strategy for reducing the swarm's degrees of freedom by grouping the problem variables and applying the same random scalars for all variables in a group. This constrains the swarm's movement so that it exploits within a smaller region rather than attempting to explore large regions. The effect of using vector or scalar random numbers to scale the cognitive and social components has previously been considered in [39, 45, 54]. However, these studies performed no further investigation into the effects of an intermediate approach, where scaling is not performed by component-wise multiplication nor by scalar multiplication, but instead by means of such a group-based scaling approach.

Sections 6.2.2 to 6.2.4 discuss strategies for modifying appropriate group sizes, including fixed-size groups, increasing the number of groups and decreasing the number of groups.

6.2.1 Groups of Variables

The proposed strategy groups the problem dimensions and applies the same random number to all of the problem dimensions within the group. If a set of decision variables are always assigned to the same group, the values of those decision variables will always be a combination of the swarm's initial values for those variables. The solution vector's values for dimensions within the same group will thus be constrained to the subspace generated by the initial particle positions (for those dimensions). This forces the swarm to perform fine-grained searching within each "group subspace". The more groups there are, the closer the search is to a canonical PSO in terms of allowed exploration. If fewer groups are imposed, the space of solutions that can be reached by the swam will become smaller. The swarm's behaviour will become increasingly like one using scalar r_1 and r_2 values, and lean more towards fine-grained searching in a small region or exploitation.

Before introducing the different grouping strategies, the notion of variable grouping will be formalized. Towards this end, consider the definition of a partition:

Definition 6.6. Let B be any set. Then the family of sets $\mathcal{G}$ is a *partition* of B if and only if all the following statements hold:

1. The union of the sets in $\mathcal{G}$ is equal to B (i.e. $\cup_{G\in\mathcal{G}} G = B$).
2. The intersection of any two sets in $\mathcal{G}$ is empty (i.e. $\forall_{G_1,G_2\in\mathcal{G}}$ if $G_1 \neq G_2$ then $G_1 \cap G_2 = \emptyset$).
3. $\mathcal{G}$ does not contain the empty set ($\emptyset \notin \mathcal{G}$).

Thus, $\mathcal{G}$ partitions B if every element of B is a member of exactly one set in $\mathcal{G}$ and $\mathcal{G}$ does not contain the empty set.

Consider a swarm consisting of n_s particles, each of n dimensions. Let $\mathcal{P}$ denote a random permutation of the set $\{1, 2, ..., n\}$. A grouping $\mathcal{G}$ of the problem variables is defined as

$$\mathcal{G} = \{G_1, G_2, ..., G_q\} \tag{6.30}$$

where every G_i is a set of integers so that $\mathcal{G}$ partitions $\mathcal{P}$ into q sets where G_1 contains the first $|G_1|$ elements of $\mathcal{P}$, G_2 contains the next $|G_2|$ elements of $\mathcal{P}$ and so forth (where

$|.|$ denotes set cardinality). Note that different groups may be of different sizes so that $|G_k|$ varies across $k \in \{1, 2, ..., q\}$.

A grouping strategy produces a grouping of the problem variables for every iteration and q may vary between iterations. A grouping strategy is called *static* if $\mathcal{P}$ is the same across all iterations of the search and is called *dynamic* if $\mathcal{P}$ is re-generated with every iteration. Each of the grouping strategies introduced in the following subsections can be dynamic or static.

Upon updating particle i's velocity, $\mathbf{r}_1$ and $\mathbf{r}_2$ will be vectors of length q where every element is drawn from a uniform distribution with range $(0, 1)$ as usual. For every problem dimension j there exists a partition $\mathcal{G}_k$ so that $j \in \mathcal{G}_k$. The velocity update equation for particle i at iteration $t + 1$ will then be given by

$$v_{ij}^{t+1} = wv_{ij}^t + c_1 r_{1k}(y_{ij}^t - x_{ij}^t) + c_2 r_{2k}(\hat{y}_j - x_{ij}^t) \tag{6.31}$$

6.2.2 Fixed Group Number

The fixed group number strategy depends on a parameter g, which specifies how many groups the variables will be divided into. Let $\bmod$ denote the modulus operator. $\mathcal{P}$ is divided into g groups, where the first $n \bmod g$ groups will contain $\frac{n}{g} + 1$ variables and the remaining $g - (n \bmod g)$ groups will contain $\frac{n}{g}$ variables.

Upon updating the velocity of particle i, the random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ will be of length g, one element for each group. The variables in group 1 will be scaled by the first elements in $\mathbf{r}_1$ and $\mathbf{r}_2$, the variables in group 2 will be scaled by the second elements and so forth so that the variables in group $q = g$ are scaled by the last elements in $\mathbf{r}_1$ and $\mathbf{r}_2$. Thus the velocity in the j^{th} dimension will be calculated by using the k^{th} element in $\mathbf{r}_1$ and $\mathbf{r}_2$ if $j \in G_k$.

The number of groups produced by this strategy is fixed across all iterations by the parameter g. This strategy and to some extent, the other strategies to be discussed may be considered as a divide and conquer method. High dimensional search spaces are generally too large to explore effectively. Restricting the swarm's movement in this fashion essentially divides the search space up into smaller sections. Unfortunately, the selected regions are essentially random and may thus not contain the global optimum (or even a good local optimum).

6.2.3 Decreasing Group Number

The decreasing group number strategy linearly decreases the number of groups from iteration to iteration. The strategy starts with the largest possible number of groups and linearly decreases the number of groups to the minimum. The maximum number of groups is clearly n, where every variable is the only member in its own group. The first iteration of this strategy will thus produce velocities exactly like those produced by the normal PSO velocity update equation (2.7). The minimum number of groups is 1, in which all the variables belong to the same group. The velocities calculated in the last iteration will thus be a linear combination of the swarm's positions in the second last iteration. If the maximum number of iterations is denoted by t_{max}, then the number of groups g for iteration $t \in \{1, ..., t_{max}\}$ is calculated by:

$$g(t) = n - \left\lfloor \frac{(t-1)}{t_{max} - 1}(n-1) \right\rfloor \tag{6.32}$$

This strategy attempts to balance the exploration and exploitation of the swarm by first allowing the particles to move with the maximum degrees of freedom, then slowly constraining their movement more and more, gradually encouraging a focus on exploitation within the reachable search space instead.

6.2.4 Increasing Group Number

The increasing group number strategy also varies the number of groups from iteration to iteration, but in this case the number of groups increases linearly with every iteration. The number of groups starts at the minimum, namely 1, where all the variables are members of the same group. The first iteration will thus produce velocities that are in the subspace generated by the swarm's initial positions. The number of groups at the last iteration, t_{max} will be the n, so that each variable is scaled by a different random value. The number of groups g for iteration $t \in \{1, ..., t_{max}\}$ is calculated by:

$$g(t) = \left\lfloor \frac{(t-1)}{t_{max} - 1}(n-1) \right\rfloor + 1 \tag{6.33}$$

Perhaps counter-intuitively, this method initially constrains particle movement to the subspace within which the swarm was initialized. The particles are slowly allowed more

freedom of movement as g increases with t. This strategy can be thought of as a beam search, where the particles move towards the more promising regions of the reachable subspace and the reachable subspace is grown with every iteration. In high dimensional spaces, where the search space grows exponentially with n and exploration is difficult, directing the search in such a manner may be efficient.

6.3 Results and Discussion

The grouping strategies discussed in the previous section were compared against each other and a standard inertia PSO to analyze their influence on the swarm's exploration behaviour. Section 6.3.1 provides a brief comparison between swarms implementing the grouping strategy and the standard PSO to prove that the swarm behaviour is affected positively. Section 6.3.2 considers the difference between the static and dynamic strategies and provides an in-depth explanation for the difference in behaviour. Next, section 6.3.3 performs an empirical analysis for the optimal number of groups when applying the strategy with a fixed number of groups. Finally, section 6.3.4 compares and analyzes the behaviour of the different grouping strategies.

The experiments were performed in the same manner as those described in Chapter 3, except for the application of variable grouping strategies.

6.3.1 Comparison to Standard Inertia PSO

The swarms that performed group-based stochastic scaling were compared to a standard PSO with inertia weight, that used the usual velocity update equation as in Equation (2.7). As before, comparisons were done by applying Friedman tests with a p-value of 0.05 to detect statistically significant differences among the best fitnesses achieved by the algorithms. If the Friedman test indicated significant differences, further pairwise comparisons were done by means of Mann-Whitney U tests with a p-value of 0.05. For the fixed group number strategy, the value $g = 10$ was used, which was found to work well empirically. As can be seen from Table 6.1, the swarms with group-based stochastic scaling almost always performed significantly better than the simple swarm, except for the dynamic fixed group number strategy (which will be discussed in the

following section). As explained in the motivation for these strategies in section 6.2.1, this is not unexpected in high dimensional search spaces, where grouping the variables will allow the swarm to employ more of a divide-and-conquer strategy. The swarm will be able to focus on thoroughly searching the subspaces to which it is restricted, rather than attempting to explore a very large search space.

Table 6.1: Best fitness comparison of swarm with grouping-based stochastic scaling against simple swarm

	> Simple	=	< Simple
Static Fixed	20	0	0
Dynamic Fixed	2	18	0
Static Decreasing	20	0	0
Dynamic Decreasing	20	0	0
Static Increasing	20	0	0
Dynamic Increasing	20	0	0

6.3.2 Static vs Dynamic Grouping Strategies

The different grouping strategies were examined and compared amongst one another (using Friedman and Mann-Whitney U tests on the best fitness achieved by the swarms, as described in 6.3.1). The investigation began by comparing the static and dynamic grouping strategies, so that the static version of a given grouping strategy is compared to its dynamic version. As can be seen in Table 6.2, the static version of a grouping strategy always performed significantly better than the dynamic version on all of the benchmark functions.

The static methods fix the permutation of the variable indices, $\mathcal{P}$, at the beginning of the search. Consider the fixed number strategy and suppose, without loss of generality, that group 1 is given by $\mathcal{G}_1 = \{1, 2, ..., k\}$. Then every possible value that the first k variables can attain must be a linear combination of elements from

$$\mathcal{I}|_{\mathcal{G}_1} = \mathcal{I}|_{\mathbb{R}^k} = \{\mathbf{P}_{\mathbb{R}^k}\mathbf{z} \big| \mathbf{z} \in \mathcal{I}\} \tag{6.34}$$

Table 6.2: Best fitness comparison between static and dynamic grouping strategies

	Dynamic > Static	=	Dynamic < Static
Fixed	0	0	20
Decreasing	0	0	20
Increasing	0	3	17

where $\mathbf{P}_{\mathbb{R}^k}\mathbf{z}$ denotes the vector formed by projecting $\mathbf{z}$ onto $\mathbb{R}^k$. This reduces the reachable regions of the search space considerably.

By contrast, the dynamic strategy recalculates $\mathcal{P}$ randomly with every iteration. So if, as in the static example, group 1 contains indices $1, 2, ..., k$ at iteration 0, chances are that many of those indices will be distributed amongst the other groups in a following iteration. The values of these variables are thus no longer restricted to some linear combination of $\mathcal{I}|_{\mathbb{R}^k}$ and the swarm easily escapes its initial subspaces - defeating the entire purpose of the proposed strategies. Figure 6.3 provides a typical illustration of the diversity behaviour of a static grouping strategy against its dynamic version. As can be seen, the diversity of the dynamic grouping method behaves almost identically to the standard PSO, supporting the statement that dynamic strategies are not sufficiently restrictive. Figure 6.4 plots the ratio between the swarm's average step size inside the initial subspace and the average step size outside the initial subspace. Clearly, the static strategy performs more searching inside the initial subspace than the dynamic strategy, also indicating that the dynamic strategy is less restrictive. This is not unexpected, given the theoretical justification in section 6.1.

The static increasing and decreasing group number strategies are less strict than the static fixed group strategy. Since the number of groups changes, every variable is in its own group for at least one iteration, and there is thus no set of variables that will always be a combination of their initial values. These strategies thus allow the swarm to leave the subspace within which it was initialized to a much larger extent than the fixed group strategy, albeit more gradually than a standard PSO. The discussion above was for the fixed group strategies. The decreasing and increasing group number strategies exhibited a less radical difference between the static and dynamic versions. Each strategy is discussed below.

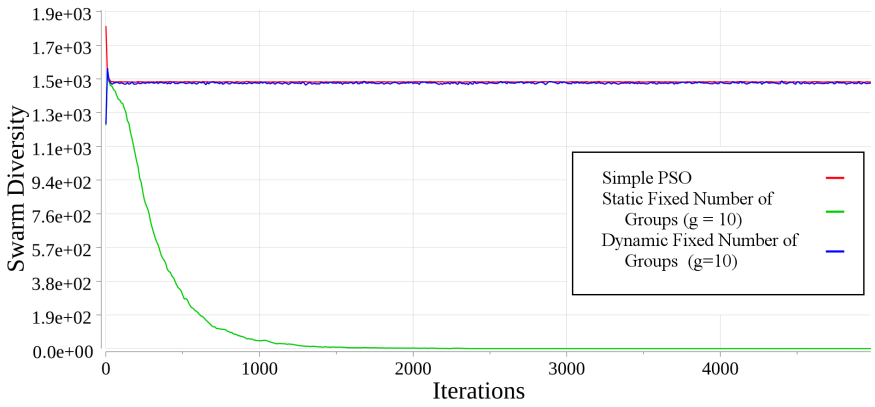


Figure 6.3: Swarm diversity per iteration of standard PSO, static fixed group number and dynamic fixed group number with g = 10 on F1 for 1000 dimensions over 5000 iterations

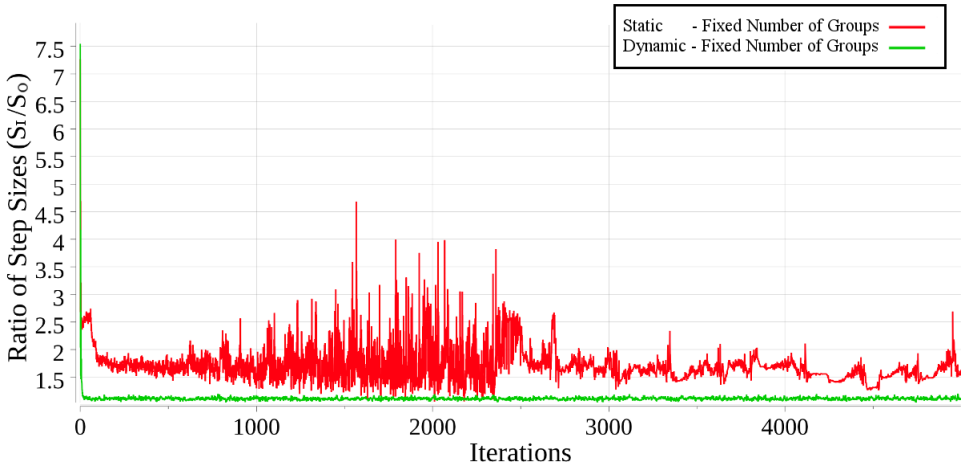


Figure 6.4: Ratio of step sizes, $\frac{S_I}{S_O}$ inside and outside initial subspaces for static vs dynamic strategy with fixed number of groups (where $g = 10$ on F3 for $n = 1000$, over 5000 iterations, one run)

Although not immediately clear from figure 6.5, both static and dynamic versions of the decreasing group number strategy exhibited high initial swarm diversity, which settled to a high value for most of the search. Towards the end of the search, as the number of groups became very small and swarm movement became strongly constrained, the diversity of the swarm decreased sharply. Once strongly restrained, the static strategy's

diversity decreased faster and remained lower than the diversity of the dynamic strategy, confirming the hypothesis that dynamic strategies allow greater freedom of movement. For most of the search, both the static and dynamic versions of the decreasing group strategy behaved very similar to a standard PSO (as shown in figure 6.6, which overlays the diversity profile of a standard PSO on the diversities of the static and dynamic decreasing group strategy PSOs). Figure 6.7 shows the ratio of step sizes inside and outside the initial space for both the static and dynamic decreasing group strategies. After a large initial spike (similar to that exhibited by the standard PSO), both versions had a ratio of approximately one for most of the search. Towards the end of the search, as swarm movement became more strongly constrained, the ratio increased. It is unexpected that the swarm should focus on the initial subspace at the end of the search, especially given that the particles escaped the initial subspace and also explored the orthogonal space for most of the search. This phenomenon is discussed further in section 6.3.4.

The difference between the static and dynamic version of the increasing group number strategy is shown in figures 6.8 and 6.9. Figure 6.8 illustrates the typical diversity profiles of the static and dynamic versions of the increasing group number strategy. Similar to the fixed group strategy, the decreasing strategy's static version exhibited lower diversity than the dynamic version throughout the search. However, unlike the fixed group strategy, the dynamic version of the increasing group number strategy did exhibit decreasing diversity, indicating that the dynamic increasing group number strategy was able to search to some extent, i.e. the swarm's behaviour was not like a standard PSO. Figure 6.9 shows that both the dynamic and fixed versions of the increasing group number strategy exhibited a very high initial spike in step size ratio, which is expected, since the swarm was strongly constrained during the initial stages of the search. Figure 6.10 plots the step size ratio from iteration 50 onwards to give an indication of the swarm's behaviour for the remainder of the search. The mean step size ratio is one for the majority of the search, indicating that the swarm escaped the initial subspace and also explored the rest of the search space. The static version of the strategy exhibited slightly elevated step size ratios in the first thousand iterations of the search, showing that it was more focused on the initial subspace than the dynamic strategy, as expected (since

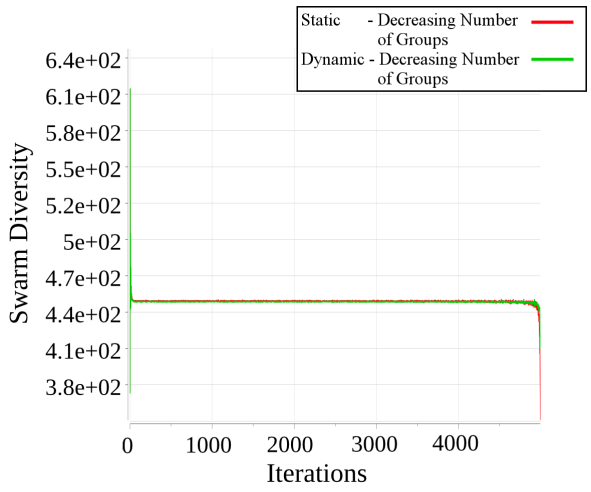


Figure 6.5: Swarm diversity for static vs dynamic strategy with decreasing number of groups on F3 (for $n = 1000$, over 5000 iterations)

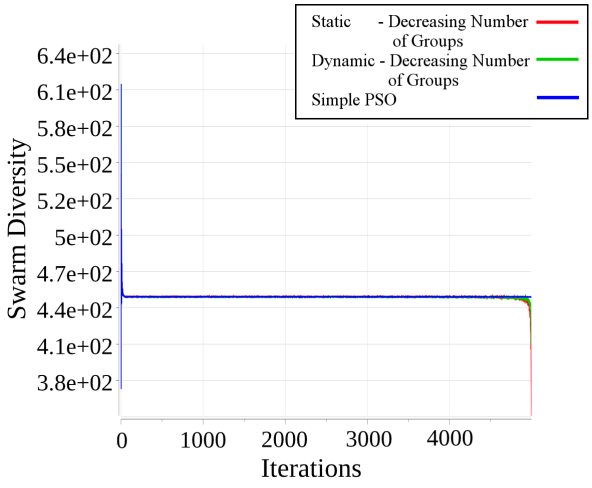


Figure 6.6: Comparison of swarm diversity profiles of standard PSO and decreasing number of groups PSO, both static and dynamic versions on F3 (for $n = 1000$, over 5000 iterations)

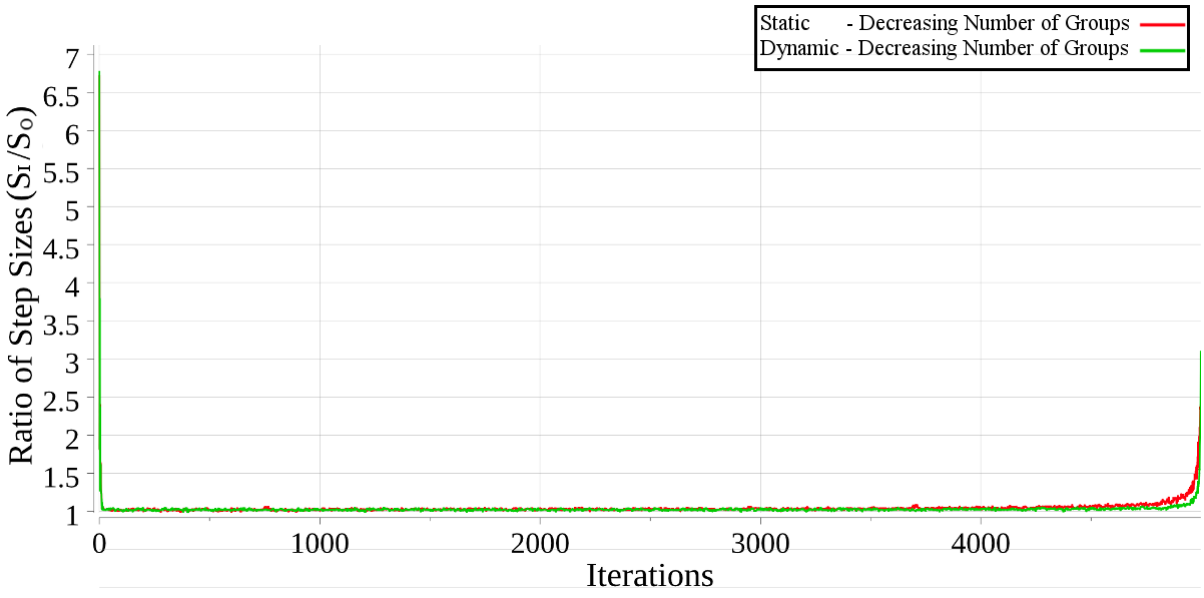


Figure 6.7: Ratio of step sizes, $\frac{S_I}{S_O}$ inside and outside initial subspaces for static vs dynamic strategy with decreasing number of groups on F3 (for $n = 1000$, over 5000 iterations)

the dynamic strategy is less constraining than the static strategy).

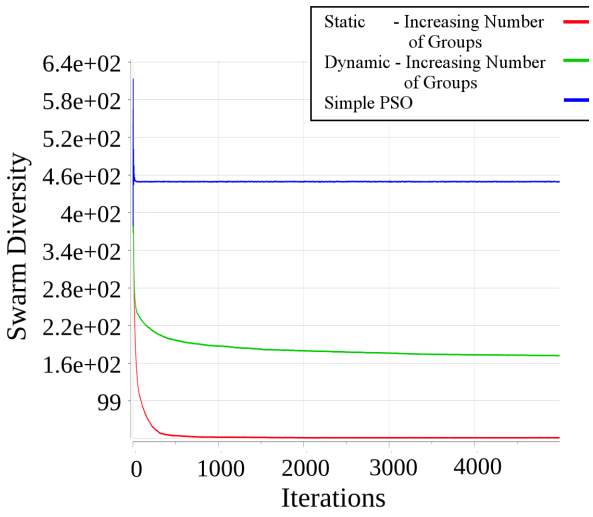


Figure 6.8: Swarm diversity for standard PSO and both static and dynamic versions of the increasing number of groups strategy on F3 (for $n = 1000$, over 5000 iterations

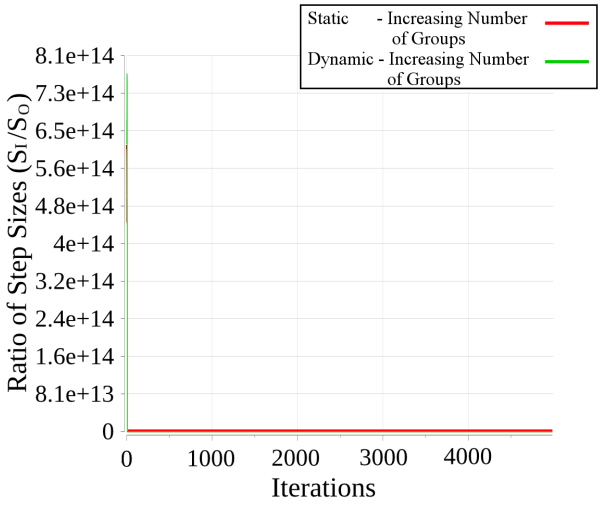


Figure 6.9: Ratio of step sizes, $\frac{S_I}{S_O}$ inside and outside initial subspaces for standard PSO and both static and dynamic strategy with increasing number of groups on F3 (for $n = 1000$, over 5000 iterations)

6.3.3 Fixed Group Numbers

A brief empirical investigation of the g parameter for the fixed group number strategy now follows. The parameter g determines how many groups the variables are divided into. A number of different values were tested for g, namely $\{2, 5, 10, 50, 100, 500\}$. The value of g that performed the best was $g = 10$, as indicated in Table 6.3. Table 6.3 compares the performance of swarms with $g = 10$ to that of the swarms with other values for g.

If the value of g is low, then the movement of the swarm is severely restricted, as discussed in the previous section. As can be seen in figure 6.11, the severe restriction on the swarm's movement causes the swarm's diversity to drop early in the search, possibly causing premature convergence. As the value of g increases, the swarm becomes more and more like a standard PSO. This is also illustrated in figure 6.11 where it can be seen

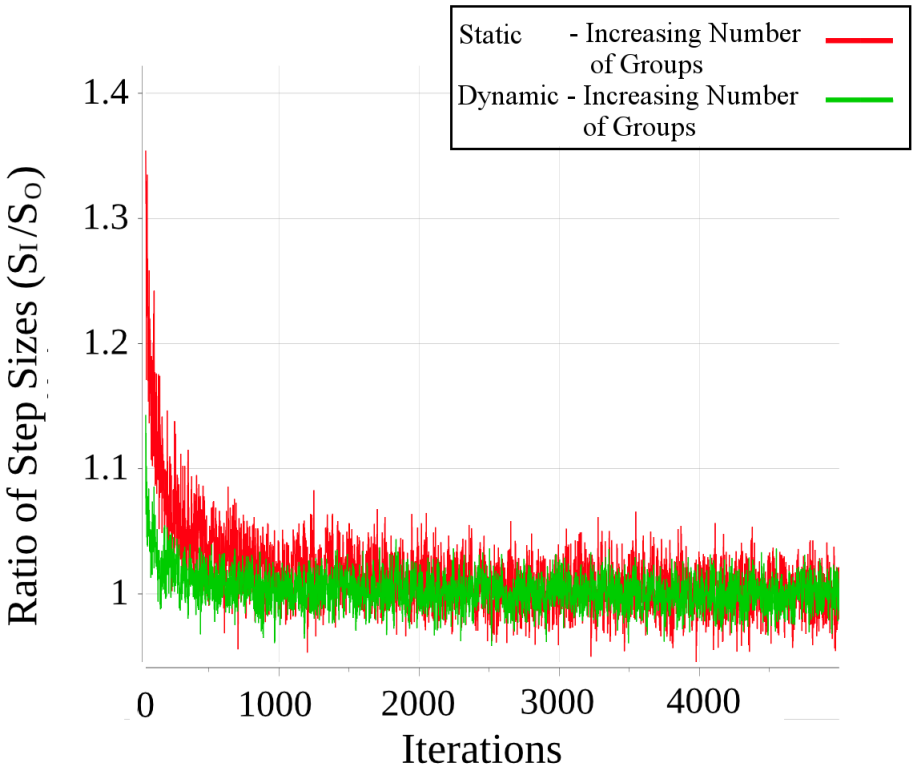


Figure 6.10: Ratio of step sizes, $\frac{S_I}{S_O}$ inside and outside initial subspaces for standard PSO and both static and dynamic strategy with increasing number of groups on F3 (for $n = 1000$, from iteration 50 onwards)

that, for values of g which are too large, the swarm's diversity profile becomes almost identical to that of a standard inertia PSO (to the point where it may be difficult to distinguish the standard PSO's diversity from that of $g \in \{50, 100, 500\}$ in the figure).

6.3.4 Comparison of Grouping Strategies

The fixed, increasing and decreasing group number strategies were compared amongst one another. Upon comparison, the decreasing group number strategy performed significantly worse than the fixed and increasing group number strategies, as can be seen in Table 6.4. Upon comparing the increasing and fixed group number strategies in Table 6.5, it is found that the increasing group number strategy is the most effective and

Table 6.3: Performance Of static fixed group number strategy for varying g-parameter

	$> g = 10$	$=$	$< g = 10$
$g = 2$	2	3	15
$g = 5$	2	4	14
$g = 50$	0	0	20
$g = 100$	0	0	20
$g = 500$	0	0	20

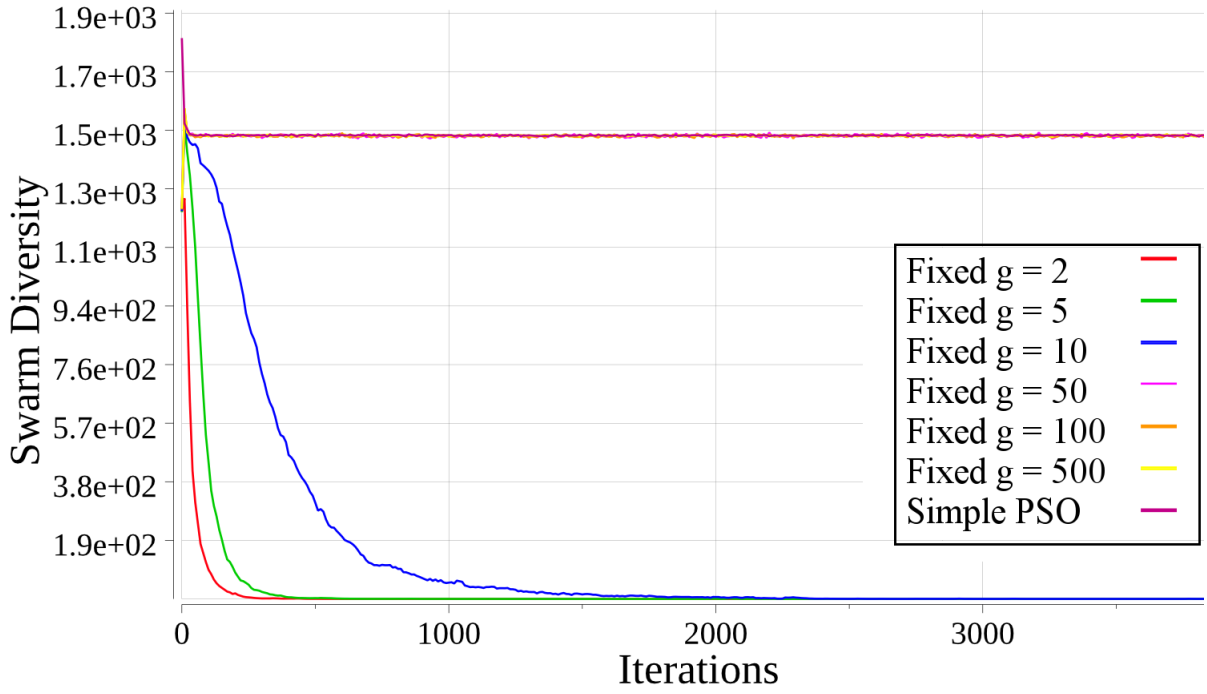


Figure 6.11: Swarm diversity per iteration of static fixed group number PSO with varying g values on F1 for 1000 dimensions over 2000 iterations

significantly outperformed the fixed group strategy on 85% the benchmark functions.

Consider figure 6.12 which plots the ratio between a swarm's average step size inside the initial subspace and outside. The figure allows inferences to be made about a swarm's ability to escape the initial subspace and to determine how effective a grouping strategy

Table 6.4: Comparison of decreasing group number strategy with fixed and increasing group number strategies

	> Decreasing	=	< Decreasing
Fixed	20	0	0
Increasing	20	0	0

is in controlling its exploration or exploitation. Note that the figure omits the first 2 iterations of the search because the ratio between the initial and outside subspaces for the increasing group number strategy was very high in those iterations, making it impossible to distinguish the swarms' behaviour for the remainder of the search. The step sizes of the swarms in the first few iterations were already discussed in detail in section 6.3.2.

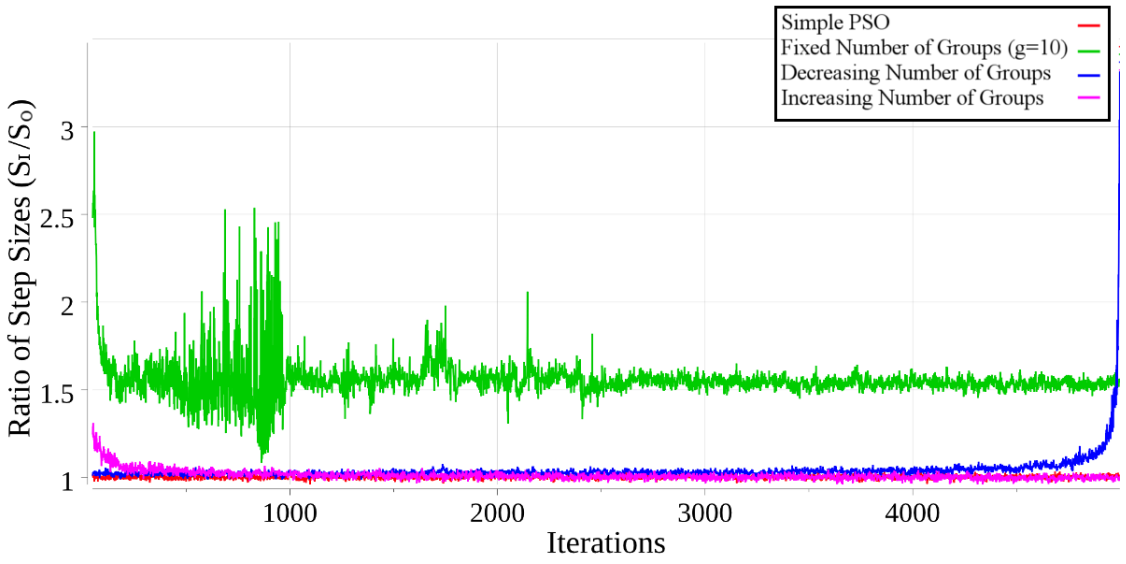


Figure 6.12: Ratio of step sizes, $\frac{S_I}{S_O}$ inside and outside initial subspaces on F3 (for $n = 1000$, from iteration 50 onwards)

The decreasing strategy allows the swarm to escape the initial subspace immediately, as can be seen in the rapid decrease of step size ratio. The strategy slowly restricts the swarm's movement, forcing the particles to exploit rather than explore as the search progresses. By the time the swarm's freedom is reduced, the swarm has already left the initial subspace to the same extent as a standard inertia weight PSO. Thus, the decreas-

Table 6.5: Comparison of increasing group number strategy with fixed and decreasing group number strategies

	$>$ Increasing	$=$	$<$ Increasing
Fixed	0	3	17
Decreasing	0	0	20

ing strategy allows as much initial exploration as a standard PSO, which causes it to face similar problems to that faced by a standard PSO in high dimensions. The standard PSO and the decreasing strategy had very similar step size ratios, both exhibiting more focus outside the initial subspace than the fixed and increasing strategies. Towards the end of the search, the step size ratios increased sharply, indicating that the swarm was taking larger steps within the initial subspace than the rest of the search space. This is unexpected, since the step size ratio for most of the search indicates that the swarm has successfully escaped the initial subspace. Since the result plotted in figure 6.12 is for a single run (due to the computational complexity of measuring the step sizes), it may be that the space in which the swarm was located contained many components from the initial subspace, biasing its movement towards the initial subspace once its movement became increasingly restricted.

For the strategy with a fixed number of groups, every group of variables will only be able to attain values in the span of the swarm's initial values for those variables. In other words, if a group q is such that $G_q = \{\mathcal{P}_r, \mathcal{P}_{r+1}, ..., \mathcal{P}_s\}$ where $\mathcal{P}_r$ denotes the r_{th} element in $\mathcal{P}$, then all the positions reachable by the swarm will have values in the span of $\mathcal{I}_{\mathcal{G}_q} = \{\mathbf{z}[\mathcal{P}_r : \mathcal{P}_s] | \mathbf{z} \in \mathcal{I}\}$ for all the variables indexed by G_q. Thus, the swarm's movement will not be linear (and it will be able to escape the initial subspace), but it will still be restricted to a large extent. This is reflected in figure 6.12. The initial drop in step size ratio shows the swarm partially leaving the initial subspace. Due to the movement restriction, the mean step size ratio remained approximately the same for the remainder of the search (which supports the theoretical discussion about the possible values of variables in the same groups).

The increasing strategy intends to impose strong restrictions on particle movement for the first few iterations, thereby focusing the search strongly inside the initial subspace.

Then as the search progresses, it is expected that the step size ratio will decrease as the swarm's degrees of freedom increases. As expected, the increasing strategy showed step size ratios greater than one in the first 250 iterations of the search. Later, the step size ratios decreased as the swarm's movement restrictions relaxed and the swarm was able to explore regions outside the initial subspace (more so than the fixed grouping strategy, as would be expected).

Other aspects of the swarm's behaviour for the different grouping strategies are examined in the remainder of the section. As can be seen in figure 6.13, the decreasing strategy's particles left the search space almost immediately and then failed to return to the search space within the allowed iterations. The decreasing strategy thus "gets lost" in the large search space, rather than searching within a small region, like the other two strategies.

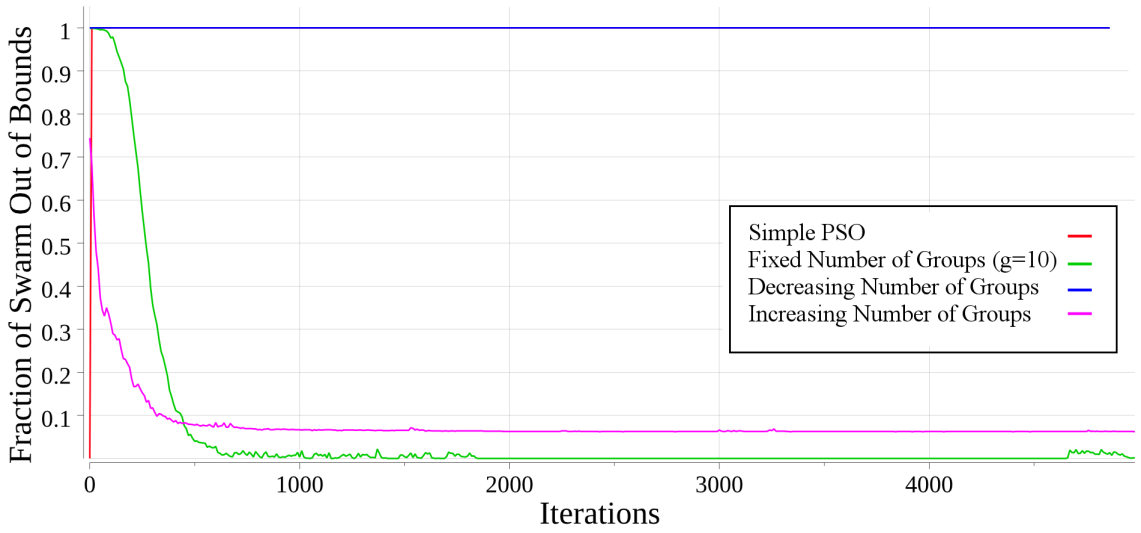


Figure 6.13: Fraction of particles out of bounds per iteration on F3 for 1000 dimensions over 5000 iterations

As can be seen in figure 6.14, which illustrates typical swarm diversity profiles on the benchmark suite, the diversity of the decreasing group number strategy fails to decrease, similar to the behaviour observed from the standard inertia weight PSO. This is not unexpected, considering that most of the swarm failed to return to the search space for

the remainder of the search.

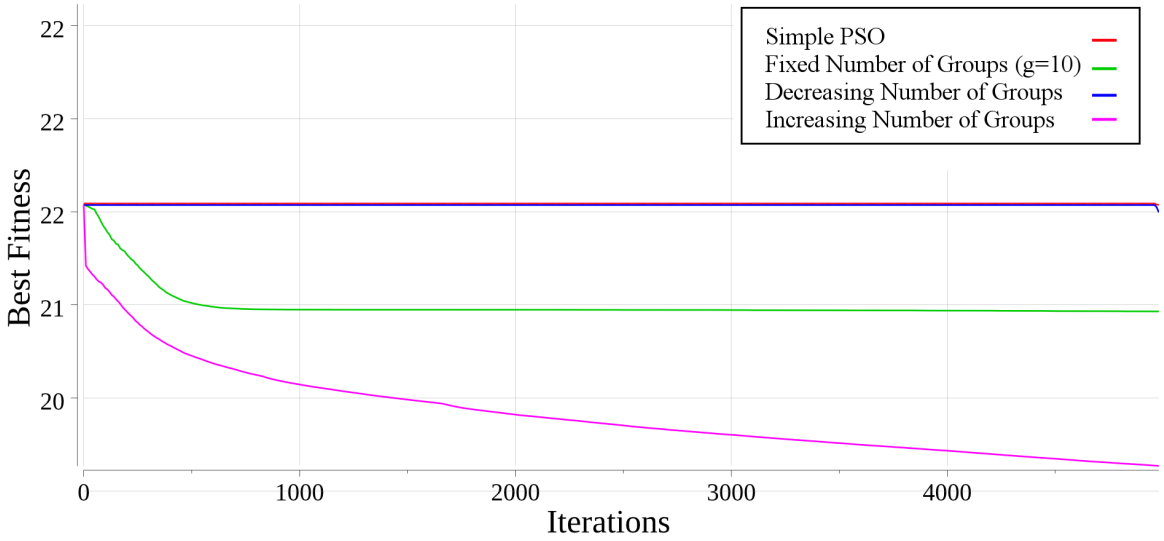


Figure 6.14: Best score per iteration of PSO with fixed, decreasing and increasing group numbers on F3 for 1000 dimensions over 5000 iterations

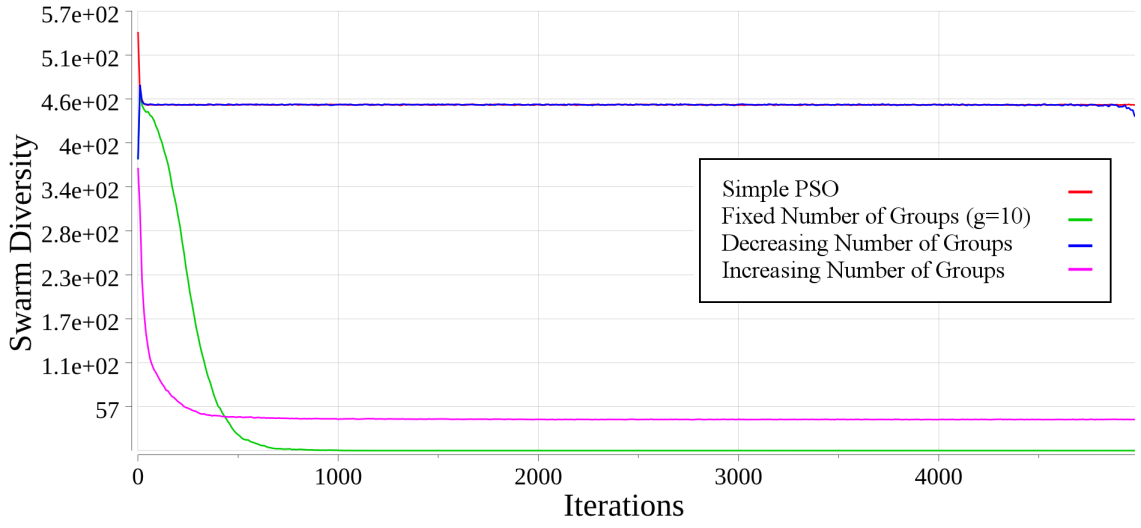


Figure 6.15: Swarm diversity per iteration of PSO with fixed, decreasing and increasing group numbers on F3 for 1000 Dimensions over 5000 Iterations

The diversity of the fixed group number strategy decreases gradually, exhibiting a beautiful balance between exploration and exploitation. As can be seen from figure 6.15, the swarm converged to the best solution that it could find approximately halfway into the allowed number of iterations. On the other hand, the diversity of the increasing group number strategy decreases steeply at first, but eventually becomes constant. Even though the swarm's diversity did not decrease, figure 6.15 illustrates that the swarm did not converge when its diversity stopped changing. Instead, the swarm continued to search locally and find better solutions until the very end of the allowed iterations. This shows that the fixed number strategy converged prematurely, whereas the increasing group strategy was able to continue improving its solution. Premature convergence from the fixed strategy could be expected, since the swarm's movement is limited. The increasing group number strategy continues to perform localized exploration as its degrees of freedom increases with every iteration, thereby preventing it from getting trapped in local minima. The increasing group number strategy also does not exhibit the unfortunate behaviour of the simple and decreasing group number swarms, where the particles immediately leave the search space due to the initial restrictions on the swarm's movement.

The fixed strategy imposes far more stringent constraints on swarm movement than the decreasing strategy (at the start of the search). If the initial swarm positions happen to be unfavourable, the strategy may prevent the swarm from being able to escape the initial subspace and find a good solution. The increasing group number strategy on the other hand, will allow the swarm to gradually leave an initial, unfavourable subspace in search of better solutions elsewhere. This guides the search, allowing more fine-grained exploration from the start. The increasing grouping strategy thus provides a good balance between exploration and exploitation in high dimensional search spaces: exploration is sufficient to escape unfruitful subspaces, but not to the extent that the swarm wastes time trawling the tremendous search space.

6.4 Summary

This chapter explained the significance of component-wise stochastic scaling in allowing the particles to reach any point in the search space. It was shown that it may be efficient to limit the swarm's reachable space in order to facilitate exploitation. The chapter introduced variable grouping strategies, which control the swarm's degrees of freedom by dividing the problem variables into groups and applying the same stochastic scalars to each group. Three different grouping strategies were considered: using a fixed number of groups, linearly decreasing the number of groups and linearly increasing the number of variable groups.

Using a fixed number and linearly increasing the number of groups were both effective in limiting the swarm's exploration. Decreasing the number of groups was more efficient than a standard inertia weight PSO, but its behaviour was generally similar to the standard PSO. Dynamic strategies that shuffle the variables assigned to a group with each iteration were tested, but were not found to be particularly effective.

Decreasing the number of groups as the search progresses initially allows particles to move freely, as in a simple swarm, then slowly restricts swarm movement. Although this follows the typical profile of exploration and exploitation, it was not effective. The initial exploration phase failed to identify fruitful regions for further exploitation, as is the case for a standard inertia PSO. Instead, the strongly restrictive approaches were more effective. The most effective approach was to restrict the swarm's movement initially and then gradually allow more freedom to encourage continued improvement.

The next chapter discusses various particle initialization strategies. On the one hand, it seems that initializing for maximum search space coverage will be advantageous, since it allows the swarm to obtain a more representative sample of the search space. On the other hand, when considering the counter-intuitive behaviour exhibited by the increasing groups strategy, it may be efficient to restrict the swarm's movement, at least initially. Thus, initialization that covers only a small part of the search space might perform well in high dimensional spaces. Swarm initialization is thus the topic of the next chapter.

Chapter 7

Swarm Initialization

In low dimensional search spaces, it is advantageous for the particles' initial positions to be spread across the search space. If the particles are well distributed throughout the search space, it decreases the chance of skipping over a good region in which the global optimum might be located. The position initialization significantly influences the likelihood of the optimum being found. On the other hand, if the particles are distributed unevenly, it may happen that the global optimum lies in a region far from all the particles' initial position. The only way in which the particles will be able to locate the region containing the optimum is if they are carried there by momentum or if the region lies in the general direction of their local and global best positions.

Usually, search space coverage is achieved by initializing the particles uniformly throughout the search space so that the initial diversity of the swarm is high. Although this initialization approach does not necessarily produce an even spread of particles [65], it performs well enough for most applications.

However, for high dimensional problems, the region of the search space that can be effectively covered by the particles' initial positions is very small because the search space grows exponentially with dimensionality. Even if the particles are distributed uniformly throughout the search space, much of the search space remains effectively unseen due to its sheer size.

Helwig and Wanka [33] have shown that, in high dimensional spaces, particles that are initialized uniform randomly are located arbitrarily close to the boundary with over-

whelming probability (i.e. becomes almost certain as $n \to \infty$). Such particles are likely to leave the search space in the following iteration. Uniform random initialization may thus exacerbate particle roaming behaviour in high dimensional spaces.

This chapter shows that, in high dimensions, it is more effective to initialize the particles within a small area of the search space rather than attempting to spread the particles as evenly as possible. A number of initialization strategies are examined, most of which attempt to maximize initial search space coverage. These strategies are compared to a novel strategy which initializes the swarm inside a small subspace of the search space. The proposed strategy was introduced in [81] and was shown to perform very well in comparison to other strategies for the high dimensional problems.

The proposed initialization strategy recognizes that effective exploration in high dimensional search spaces is practically impossible. So rather than distributing the swarm as uniformly as possible throughout the entire search space, the strategy initializes the swarm in a small subspace of the search space. The swarm does not wander far from the initialized region and is thus able to find a - potentially local - optimum within that region. But, as the results show, the local optimum found by such a swarm is usually better than the optimum found by a swarm that attempts to explore the entire search space.

The chapter begins by examining a number of existing initialization strategies from literature in section 7.1. Next, section 7.2 discusses a new initialization strategy, which initializes the swarm within a small initial subspace. Thus, instead of attempting to promote exploration, the initialization strategy forces the swarm to focus more on exploitation or rather, exploration of only a small part of the search space. Section 7.3 provides a quick overview of the experimental method and section 7.4 discusses the the results of the experiments. Reasons for the observed behaviour are also discussed. Finally, section 7.5 concludes the chapter.

7.1 Initialization Methods

This section introduces different swarm initialization strategies that have been used in literature. Section 7.1.1 discusses uniform random initialization. Next, section 7.1.2

introduces initialization using the Sobol quasi-random number generator. 7.1.1 discusses how Voronoi tesselations can be used as a particle initialization strategy. Lastly, section 7.1.4 discusses the nonlinear simplex method and how it can be used to generate particle positions.

7.1.1 Uniform Random Initialization

Usually [23], the initial positions of the particles are obtained by sampling from a uniform distribution as follows:

$$\mathbf{x}_i^0 \sim U(L, U)^n \tag{7.1}$$

where n is the dimensionality of the search space and $U, L \in \mathbb{R}$ denote the upper and lower bounds of the search space, respectively. A pseudo-random number generator is usually used to generate these positions. It has been proved theoretically that, as the problem dimensionality increases, the probability of a particle being initialized arbitrarily close to the boundary in at least one dimension goes to one (i.e. becomes certain) [33]. Thus, uniform random initialization may not be the best choice when the problem has many dimensions.

7.1.2 Sobol Sequences

It has been found by Schoemann and Engelbrecht [65] that using quasi-random number generators to generate initial particle positions can improve PSO performance. Quasi-random number generators exhibit low discrepancy: for any given set, the proportion of points generated by a quasi-random number generator is approximately proportional to the measure of the set. In other words, the point set generated by such a sequence does not have gaps or clusters.

In order to provide a formal definition of a low-discrepancy sequence, the notion of discrepancy must be formalized. Consider the definition of *discrepancy* below, as in [17]. Note that the definitions below consider points that are generated in the half-open interval $[0, 1)$ and can be scaled as necessary.

Definition 7.1. Consider the n-dimensional half-open unit cube, $\mathbb{I}^n = [0, 1)^n$ where $n \geq 1$. For s-many points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_s \in \mathbb{I}^n$ and a sub-interval J of $\mathbb{I}^n$, let $A(J)$ count

the number of points $\mathbf{p}_i \in J$ and let $V(J)$ denote the volume of J. The *discrepancy*, $\mathcal{D}(J, s)$, is defined as

$$\mathcal{D}(J, s) = \left| \frac{A(J)}{s} - V(J) \right| \tag{7.2}$$

Thus, the discrepancy is the difference between the proportion of points in the subinterval J and the volume of J (where the proportion of the points in J is relative to the total number of points in the unit cube $\mathbb{I}^n$ and the volume of J is relative to the volume of the unit cube, which has volume of one). Next, consider the definition of worst-case discrepancy:

Definition 7.2. The *worst-case discrepancy* of a set of points $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_s\} \in \mathbb{I}^n$ is defined as

$$\mathcal{D}^*(s) = \max_J |\mathcal{D}(J, s)| \tag{7.3}$$

The worst-case discrepancy may also be called the star-discrepancy [48]. A low-discrepancy sequence is defined in terms of the star-discrepancy, as can be seen in the definition that follows.

Definition 7.3. A *low-discrepancy sequence* in $\mathbb{I}^n$ minimizes $\mathcal{D}^*(s)$. Thus, it is a sequence of points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_s \in \mathbb{I}^n$ for which the worst case discrepancy is as small as possible.

Quasi-random number generators thus ensure that the swarm is initially distributed evenly throughout the search space, providing the swarm with relatively good search space coverage.

The Sobol sequence is one such quasi-random sequence. In particular, it is a digital low-discrepancy sequence, meaning that the sequence of numbers is constructed by performing binary operations on binary expansions. Further information about the implementation of Sobol sequences is provided by Joe and Kuo in [37].

7.1.3 Centroidal Voronoi Tesselations

Richards and Ventura suggested Voronoi tesselations as a particle initialization strategy in [60]. Voronoi tesselations partition the search space into a number of small cells.

These cells should be approximately the same size so that their centers are distributed uniformly throughout the search space.

Such cells are produced by a set of generators. Each generator is assigned all the points in the search space that are closer to that generator than to any of the other generators. In this way, the search space is partitioned into a number of cells, each around one of the generators. For centroidal Voronoi tesselations (CVT), the generators are at the center of their cells.

The CVTs were calculated as described by Ju *et. al.* [38]. Initially, the set of generators are chosen randomly within the search space. The positions of the generators are then adjusted over a number of iterations as follows: At every iteration, a number of sample points are chosen in the search space. Each sample point is allocated to the closest generator. Calculate $\mathbf{a}_i$, the centroid of the sample points assigned to the i_{th} generator. Then move every generator $\mathbf{g}_i$ closer to $\mathbf{a}_i$ by some fraction of the distance between $\mathbf{g}_i$ and $\mathbf{a}_i$.

The pseudo-code for CVT is described in algorithm 7.1.

7.1.4 Nonlinear Simplex Method

Parsopoulos and Vrahatis [55] used the non-linear simplex method (NSM) to initialize particle swarms. The NSM was introduced by Nelder and Mead [50] as a means of function minimization. An n-dimensional simplex is a geometrical figure with $n + 1$ vertices. The vertices of a simplex are initialized to be points in the search space, each with a corresponding score according to the fitness function. For a number of iterations, the simplex "walks" around the search space, each time moving the vertex with the lowest score to a point with a higher score. Every step that the simplex takes can be described in terms of reflections, contractions and expansions.

Let $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denote the $n + 1$ vertices of a simplex. Let the vertex with the best score be denoted by $\mathbf{g}$. Let the vertex with the worst score be denoted by $\mathbf{b}$. Also, let $\bar{\mathbf{p}}$ denote the centroid calculated using all the vertices except $\mathbf{b}$.

A reflection of p is defined by

$$\mathbf{p}^* = (1 + \alpha)\bar{\mathbf{p}} - \alpha\mathbf{p} \tag{7.4}$$

//Produce s-many vectors in the $[L, U]^n$
for all *generators* $i = 1, 2, 3, ..., s$ **do**
Choose $\mathbf{g}_i$ randomly within search space as i_{th} generator
end for
for all *iterations* $k = 1, 2, 3, ..., MAX\ ITERATIONS$ **do**
Build a set, Q, of sample points by choosing q-many points randomly within the search space
for all *sample points* $\mathbf{p} \in Q$ **do**
Find $\mathbf{g}_i$, the generator closest to $\mathbf{p}$
Add $\mathbf{p}$ to G_i, the subset of Q containing the points that are closer to $\mathbf{g}_i$ than any other generator
end for
for $i \in 1, 2, 3..., s$ **do**
Calculate $\mathbf{a}_i$, the arithmetic mean of all the points in G_i
Move $\mathbf{g}_i$ closer to $\mathbf{a}_i$ by some fraction of the distance between $\mathbf{g}_i$ and $\mathbf{a}_i$
end for
end for
return $\{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_s\}$

Algorithm 7.1: Position generation by centroidal voronoi tessellations

where α is a positive constant known as the reflection coefficient. An expansion of $\mathbf{p}$ is defined by

$$\mathbf{p}^{**} = \gamma\mathbf{p} + (1 - \gamma)\bar{\mathbf{p}} \tag{7.5}$$

where γ is known as the expansion coefficient and is a constant greater than one. A contraction of $\mathbf{p}$ is defined by

$$\mathbf{p}^{***} = \beta\mathbf{p} + (1 - \beta)\bar{\mathbf{p}} \tag{7.6}$$

where β is between zero and one and is known as the contraction coefficient.

At every iteration, $\mathbf{b}$ is reflected to $\mathbf{b}^*$. There are three possible cases:

1. If the score of the new position is neither better than $\mathbf{g}$ nor worse than $\mathbf{b}$, $\mathbf{b}$ is replaced by $\mathbf{b}^*$ and the iteration ends.

2. If the score of $\mathbf{b}^*$ is better than $\mathbf{g}$, $\mathbf{b}^*$ is expanded to $\mathbf{b}^{**}$. If the score of $\mathbf{b}^{**}$ is better than $\mathbf{g}$, then $\mathbf{b}^{**}$ replaces $\mathbf{b}$, otherwise the expansion fails, and $\mathbf{b}^*$ replaces $\mathbf{b}$ and the iteration ends.

3. Lastly, if $\mathbf{b}^*$ is worse than all the other points (except possibly $\mathbf{b}$), then $\mathbf{b}$ is replaced by $min\{\mathbf{b}, \mathbf{b}^*\}$. The new $\mathbf{b}$ is then contracted to $\mathbf{b}^{***}$.
   If the score of $\mathbf{b}^{***}$ is better than $\mathbf{b}$, $\mathbf{b}^{***}$ replaces $\mathbf{b}$. Otherwise, the contraction fails and all $\mathbf{p}_i$ are replaced by $\frac{\mathbf{p}_i - \mathbf{g}}{2}$ and the iteration restarts.

The initial suggestion for using the NSM applied it to problems where the swarm size, s, was larger than the problem dimension, n. In this case, the simplex was initialized randomly in the search space and allowed to step $s - (n + 1)$ many times. The points through which the simplex's vertices moved were used as the initial particle positions.

However, some of the problems examined in this thesis were of dimensionality far higher than that of the swarm size. For these problems, the simplex was allowed to take s steps and the best s points through which the simplex's vertices had moved were selected as the initial particle positions. This approach may cause the swarm to converge prematurely, by essentially initializing the entire swarm in a local optimum. However, given that an exploitation-focus search is more efficient in high dimensional spaces, this may be an advantage.

7.2 Proposed Initialization Strategy

The proposed swarm initialization technique initializes the swarm in a small subspace of the entire search space, thereby forcing the swarm to focus on exploitation within the small subspace rather than attempting to explore the expanse of the search space.

The initialization strategy makes use of a "seed set", which is simply a set of u, randomly generated, n-dimensional orthogonal unit vectors. The seed set is obtained by means of the modified Gram-Schmidt (MGS) [52], where the initial set of linearly independent vectors are generated randomly. The smaller the seed set, the fewer linearly independent vectors are used to initialize $\mathcal{G}$, the set of all initial particle positions, velocities, and personal best positions. As discussed in Chapter 6, the subspace within

//Given a set of n-dimensional input vectors, $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_u\}$

for $i \in \{1, 2, 3, ..., u\}$ **do**

$\mathbf{a}_i = \mathbf{v}_i - \sum_{j=1}^{i-1} \frac{<\mathbf{v}_i, \mathbf{a}_j>}{<\mathbf{a}_j, \mathbf{a}_j>} \mathbf{a}_j$

$\mathbf{b}_i = \frac{\mathbf{a}_i}{||\mathbf{a}_i||}$

end for

return $\{\mathbf{b}_1, ..., \mathbf{b}_u\}$

Algorithm 7.2: Gram-Schmidt method of orthogonalization

which the swarm is initialized will be small if the seed set contains a small number of vectors.

For completeness, a brief discussion of the Gram-Schmidt method follows. The Gram-Schmidt method receives u-many, n-dimensional vectors as input from which it generates a set of orthogonal vectors. If the input vectors are all linearly independent, then the Gram-Schmidt method will produce $\min\{u, n\}$ many orthogonal vectors.

The set of orthogonal vectors is generated stepwise. To produce the i_{th} orthogonal vector $\mathbf{b}_i$, the i_{th} input vector $\mathbf{v}_i$ is projected onto the subspace generated by all of the orthogonal vectors already calculated, $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1}$. The orthogonal vector $\mathbf{b}_i$ is then defined to be the difference between $\mathbf{v}_i$ and its projection. Note that all the $\mathbf{b}_i$ for $i > n$ will be zero vectors.

Intuitively, the projection step finds the components of the input vector that is shared by the orthogonal vectors calculated thus far. The seed vector $\mathbf{b}_i$ is then found by removing all the shared components from the input vector. The Gram-Schmidt method is given in algorithm 7.2.

In the algorithm above, $< \mathbf{a}, \mathbf{b} >$ denotes the inner product of the two vectors $\mathbf{a}$ and $\mathbf{b}$.

The Gram-Schmidt method as discussed above is susceptible to rounding errors. The modified Gram Schmidt method computes the projection step differently so that the process is numerically stable. Usually, $\mathbf{a}_i$ is calculated in one step as in equation (7.7)

given below:

$$\mathbf{a}_i = \mathbf{v}_i - \sum_{j=1}^{i-1} \frac{< \mathbf{v}_i, \mathbf{a}_j >}{< \mathbf{a}_j, \mathbf{a}_j >} \mathbf{a}_j \tag{7.7}$$

Instead, the MGS method calculates $\mathbf{a}_i$ iteratively according to

$$\begin{aligned}
\mathbf{a}_i^{(1)} &= \mathbf{v}_i - \frac{< \mathbf{v}_i, \mathbf{a}_1 >}{< \mathbf{a}_1, \mathbf{a}_1 >} \mathbf{a}_1 \\
\mathbf{a}_i^{(2)} &= \mathbf{a}_i^{(1)} - \frac{< \mathbf{a}_i^{(1)}, \mathbf{a}_2 >}{< \mathbf{a}_2, \mathbf{a}_2 >} \mathbf{a}_2 \\
&\dots \\
\mathbf{a}_i^{(i-2)} &= \mathbf{a}_i^{(i-3)} - \frac{< \mathbf{a}_i^{(i-3)}, \mathbf{a}_{(i-2)} >}{< \mathbf{a}_{(i-2)}, \mathbf{a}_{(i-2)} >} \mathbf{a}_{(i-2)} \\
\mathbf{a}_i^{(i-1)} &= \mathbf{a}_i^{(i-2)} - \frac{< \mathbf{a}_i^{(i-2)}, \mathbf{a}_{(i-1)} >}{< \mathbf{a}_{(i-1)}, \mathbf{a}_{(i-1)} >} \mathbf{a}_{(i-1)}
\end{aligned}$$

The vector, $\mathbf{a}_i^{(k)}$, generated at step k is orthogonal to the vector generated at the previous step, $\mathbf{a}_i^{(k-1)}$, so that $\mathbf{a}_i^{(k)}$ is also orthogonalized against any errors introduced at the previous step due to limited precision. The implementation of the initialization technique thus uses the MGS method instead.

In order to produce the linearly independent seed set, the required number of vectors were sampled randomly from $[L, U]^n$. If any two of the seed vectors, $\mathbf{v}_i$, $\mathbf{v}_j$ were linearly dependent (where $i < j$), then $\mathbf{a}_j = 0$. This condition can be tested for and if it occurs, a new random vector can be sampled to replace $\mathbf{v}_j$. In practice, this almost never occurred since it is unlikely for two randomly sampled vectors to be linearly dependent (even when considering the loss of precision due to computer representation). There is thus very little computational cost involved in finding the seed set.

The initialization strategy generates an n-dimensional point which can be used either as a particle's initial position, velocity, or personal best position. The point is selected randomly on a line, L, that passes through the centre of the search space. The direction, $\mathbf{d}$, of the line is determined by the seed set.

The direction of the line is a random linear combination of all the vectors in the seed set, calculated as

$$\mathbf{d} = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \dots + c_u\mathbf{b}_u \tag{7.8}$$

where each $c_i \sim U(-1, 1)$ and each $\mathbf{b}_i$ is an element from the seed set.

The point, $\mathbf{p}$, is generated on a line with the equation,

$$\mathbf{p} = t\mathbf{d} + \mathbf{c} \tag{7.9}$$

where $\mathbf{c}$ denotes the centre of the search space and t is a scalar. The value for t is drawn from a uniform distribution with bounds (t_{min}, t_{max}). Let the bounds of the j-th dimension of the search space be (L_j, U_j). Then the bounds for t are calculated according to algorithm 7.3.

The algorithm is illustrated in figure 7.1 for a 2-dimensional search space centred at $(0, 0)$. According to the diagram, the algorithm will return bounds $(t_{min}, t_{max}) = (\frac{y_{min}}{d_y}, \frac{y_{max}}{d_y})$.

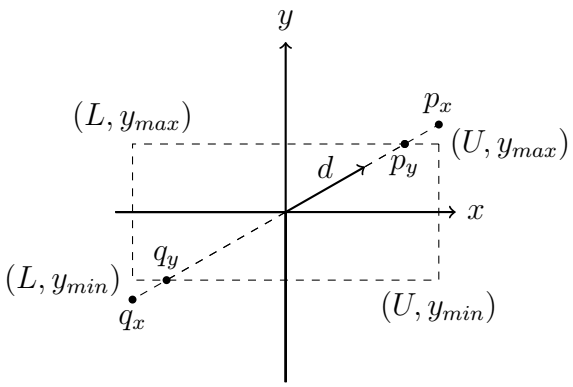


Figure 7.1: Determining bounds for the scalar t in a 2-dimensional space centred at $(0, 0)$, given a direction vector $\mathbf{d}$

Since the relative sizes of the direction vector's components differ, the bounds (t_{min}, t_{max}) may define a very small range that causes the generated positions to be near the origin in all dimensions except the few for which the direction vector is relatively large. In order to prevent all of the generated points from being concentrated near the centre of the search space, the points were allowed to be generated outside of the search space in a given dimension by a small margin, denoted by ϵ in algorithm 7.3.

```
//Calculate t-Bounds
    for each dimension j ∈ {1, 2, 3, ..., n} do
        if d_j == 0
            continue
        endif
        // Find the point p, the intersection between L and the maximum
           boundary of the j_th dimension
        t_1 = (x_{j,max} − c_j) / d_j
        p = t_1 d + c
        if !InBounds(p, ε)
            continue
        endif
        // Find the point q, the intersection between L and the minimum
           boundary of the j_th dimension
        t_2 = (x_{j,min} − c_j) / d_j
        q = t_2 d + c
        if !InBounds(q, ε)
            continue
        endif
        return t_min = min(t_1, t_2) and t_max = max(t_1, t_2)
    end for
//Where the InBounds function is defined by
    function InBounds(p, ε)
        for each dimension j ∈ {1, 2, 3, ..., n}
            if |p_i − x_{i,max}| or |p_i − x_{i,min}| > ε
                return false
            endif
        end for
      return true
```

Algorithm 7.3: Algorithm for calculating t-bounds

7.3 Experimental Procedure

The experiment tested the proposed initialization strategy against the other strategies discussed in section 7.1. The experiments followed the same general procedure as outlined in Chapter 3, except that the particles were initialized using different strategies.

Since the proposed initalization strategy initializes particles on lines that pass through the centre of the search space, it would not be representative to use benchmark functions with optima near the centre of the search space. Recall that the benchmark functions are all shifted by a random vector, sampled from $U(L, U)^n$. The proposed initialization method is thus not biased towards the benchmark suite by initializing the particles near the center of the search space.

Some implementation details for the different initialization strategies are provided below:

- The quasi-random number generator used for the Sobol sequence initialization strategy was implemented by generating multiple large point sets in Matlab with *skip*= 1024 and *leap*= 100. *Skip* specifies the how many of the initial points in the sequence will be ignored. For example, the sequence $\{5, 3, 8, 2, 6, 7, 1, 6\}$ with $skip = 2$ will transform the sequence into $\{8, 2, 6, 7, 1, 6\}$. *Leap* specifies how many points in the sequence should be ignored for every point that is selected. For example, the sequence $\{5, 3, 8, 2, 6, 7, 1, 6\}$ with $skip = 2$ and $leap = 1$ will be emitted as $\{8, 6, 6\}$.

- The CVT strategy was allowed to run for 5 iterations. With every iteration, 50000 sample points were generated as suggested in [60]. Every generator was moved 0.6 of the distance between it and the centroid of the sample points assigned to it.

- The NSM was implemented as described in the previous section with values $\alpha = 1.0$, $\gamma = 2.0$ and $\beta = 0.5$ as suggested in [50].

The proposed initialization strategy was implemented as described in section 7.2. Particle positions were generated with an ϵ of one tenth the range of the search space, $(0.1(U - L))$. Personal best positions were generated with an ϵ of zero to ensure that

the particle's attractor remains inside the search space. Five different values for the size of the seed set, u, were tested, namely $\{1, 5, 25, 50, 100\}$.

7.4 Results and Discussion

This section provides comparative results of the simulations described in section 7.3, as well as a detailed discussion of the observed outcomes. Section 7.4.1 discusses the general behaviour of all the initialization strategies as the problem dimensionality increases. Section 7.4.2 examines the influence of the seed set size on the swarm's behaviour and performance. Section 7.4.3 observes that the more successful initialization algorithms both exhibit very low initial swarm diversity and tests whether low initial swarm spread is the only factor contributing to their success. Lastly, section 7.4.4 examines the roaming behaviour exhibited by the two most successful initialization strategies.

7.4.1 Initialization and Dimensionality

Figure 7.2 gives an indication of the best-performing initialization strategies for each dimension. Figure 7.2 was generated in the same manner as figures 4.2 and 4.3 from Chapter 4, where the color of a block represents how well the initialization strategy, determined by the block's row, performed on the problem suite with dimensionality n, determined by the block's column. The lighter a strategy's block is, the better the strategy performed in comparison with the other strategies for the given dimensionality. Note that it is not possible to have a seed set that contains more vectors than the problem dimensionality. Thus, for $n = 10$, there are no simulations for seed sets larger than 5. Similarly, for $n = 50$, there are no simulations for $u = 100$. An interpretation of figure 7.2 follows below. These results are supported by tables 7.1 to 7.6 which show the result of pair-wise Mann-Whitney U tests, which compare the subspace-based strategy with the best-performing u for the given dimensionality to all the other strategies.

On the low dimensional problems, i.e. $n = 10$ and $n = 50$, the subspace-based initialization strategy shows no advantage. In fact, most of the other initialization strategies performed better, including the uniform random method. However, as the dimensionality increases, the performance of the uniform random, Sobol sequence and NSM strategies

Table 7.1: Comparison with subspace-based PSO on $n = 10$

	> Subspace u=5	=	< Subspace u=5
Uniform Random	2	18	0
Sobol Sequence	2	18	0
NSM	0	18	2
CVT	2	18	0

Table 7.2: Comparison with subspace-based PSO on $n = 50$

	> Subspace u=5	=	< Subspace u=5
Uniform Random	4	15	1
Sobol Sequence	1	19	0
NSM	3	16	1
CVT	3	16	1

Table 7.3: Comparison with subspace-based PSO on $n = 100$

	> Subspace u=5	=	< Subspace u=5
Uniform Random	0	14	6
Sobol Sequence	0	16	4
NSM	3	10	7
CVT	1	19	0

Table 7.4: Comparison with subspace-based PSO on $n = 500$

	> Subspace u=25	=	< Subspace u=25
Uniform Random	0	0	20
Sobol Sequence	0	0	20
NSM	0	0	20
CVT	7	12	1

Table 7.5: Comparison with subspace-based PSO on $n = 750$

	> Subspace u=25	=	< Subspace u=25
Uniform Random	0	0	20
Sobol Sequence	0	0	20
NSM	0	0	20
CVT	5	15	0

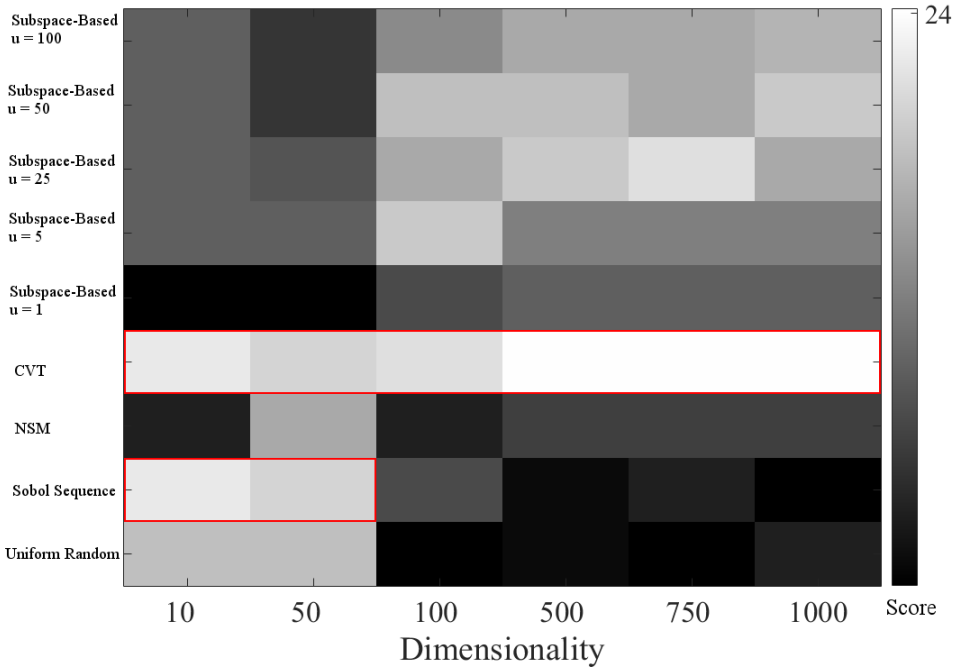


Figure 7.2: Performance of different initialization strategies as dimensionality varies (lighter is better)

deteriorate and the relative performance of the subspace-based methods increases. The CVT method consistently provided good results across all dimensions (highlighted with red) and as n increases, its performance becomes noticeably better in comparison to the other strategies. [1]

For the lowest dimensional case ($n = 10$), the Sobol sequence strategy performed better than the uniform random strategy. The Sobol sequence initialization strategy ensures that the particle positions are initialized in such a way that the particles do

[1]Similar experiments have been performed before in [81], but on a smaller range of problem dimensionalities with a larger benchmark suite and fewer particles. When comparing CVT and the subspace-based method, CVT performed significantly better on the benchmark suite used for the thesis in comparison to the benchmark suite used in [81], where the subspace-based method performed significantly better. Although the best performing algorithm differs slightly between benchmark suites, the general trend in algorithm behaviour remains consistent regardless of the chosen benchmark suite. It may also be noted that the worse performing algorithms remained the same between benchmark suites.

Table 7.6: Comparison with subspace-based PSO on $n = 1000$

	$>$ Subspace u=50	$=$	$<$ Subspace u=50
Uniform Random	0	0	20
Sobol Sequence	0	0	20
NSM	0	0	20
CVT	4	16	0

not cluster together and that there are no large gaps between them initially. Based on the Sobol sequence's performance in comparison with the uniform random strategy, spreading out evenly is a useful strategy in the lower dimensional cases. But as the dimensionality increases (from $n = 500$ onwards), the Sobol sequence did not perform better than the uniform random strategy and performed almost as poorly as the uniform random strategy.

As the problem dimensionality increases, the swarms are no longer able to explore the exponentially growing search space effectively and the swarm's performance deteriorates. A typical profile of best score achieved by the swarm thus far in the search is shown in figure 7.3 for every iteration. The figure shows that the best solution found by the Sobol sequence was discovered within the first few iterations and then never improved upon (figure 7.4 supports this assertion, with a plot of the Euclidean distance of the best solution from the optimum); the good solutions found by the swarm were thus not due to gradual exploration of the search space, but rather because the solution found by the swarm was serendipitously near one of the swarm's initial points. Such propitious initialization becomes less likely as the problem dimensionality increases.

The observed behaviour can be explained by the hypothesis that for high dimensional problems, initializing particles evenly throughout the search space may not be beneficial. As explained in section 7.2, it is proposed that forcing the swarm to search within a small part of the search space may yield better results than attempting to explore the entire search space.

Since the Sobol sequence swarm and the uniform random PSO swarm are initialized evenly throughout the search space, there will be large distances between the initial

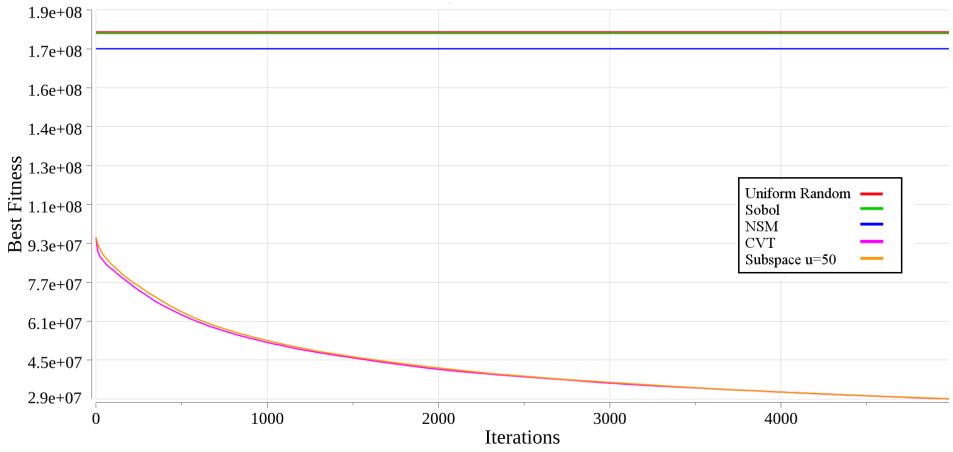


Figure 7.3: Best score per iteration for F17 with $n = 1000$

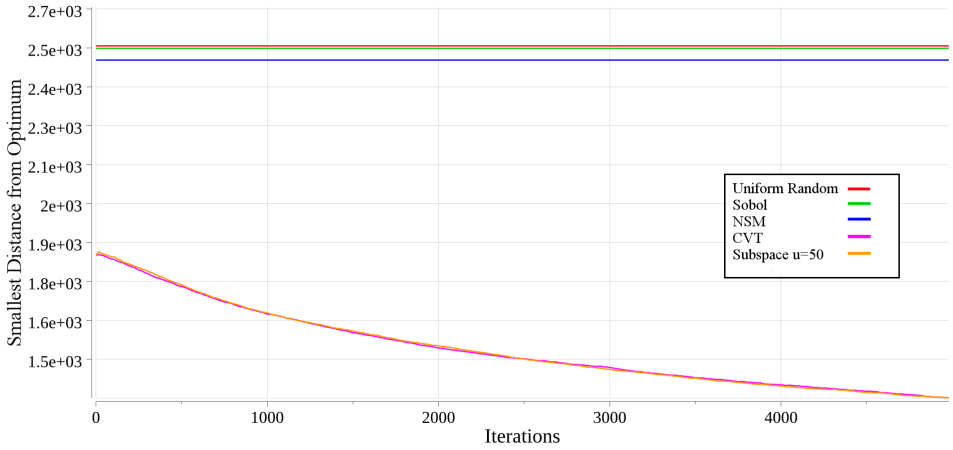


Figure 7.4: Distance of best solution from the optimum for F17 with $n = 1000$

positions of the particles. Although the NSM is not initialized evenly throughout the search space, the initial particle positions are the s points with the highest scores through which the n dimensional simplex passes. In high dimensions, this becomes equivalent to a brute-force approach for finding a favourable region in the search space, since the number of points generated by the simplex's iterative walk is much smaller than the number of its initial vertices $(n + 1)$, which are chosen randomly. The distance between the initial particle positions are thus also quite large. This is supported by figure 7.5.

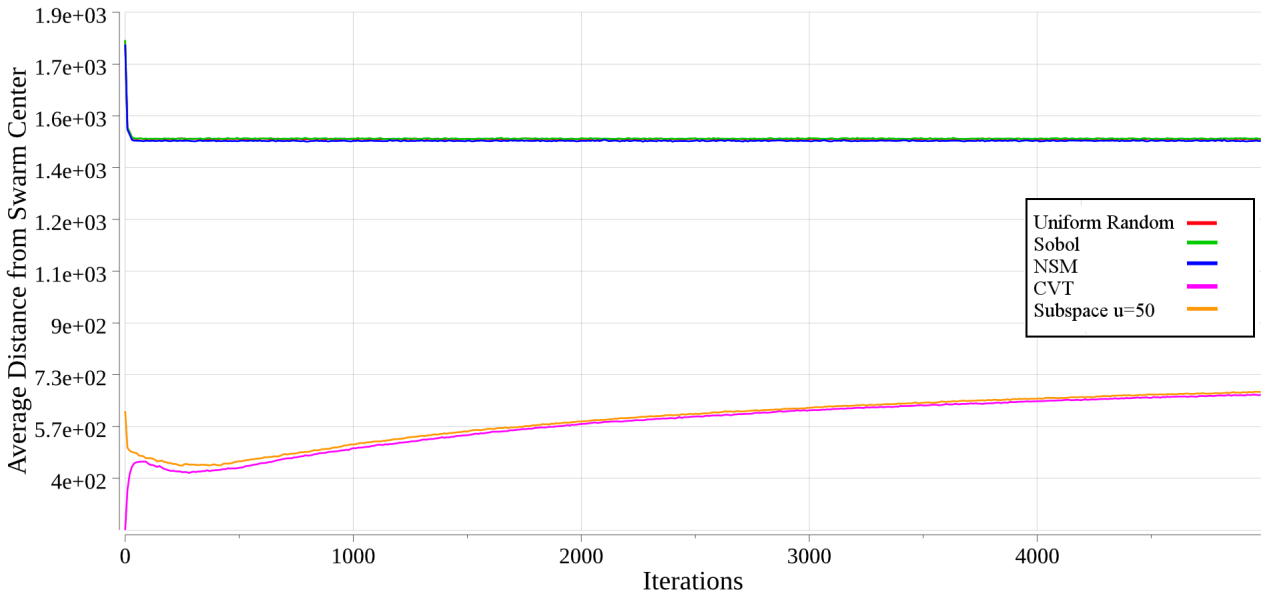


Figure 7.5: Swarm diversity per iteration on F17 with $n = 2000$

The momentum of the particles will thus be fairly large, due to the large initial distances between the particles. Even if a good solution is found, the particles will be unable to exploit the surrounding area effectively due to their large velocities, which will prevent fine-grained searching.

The argument above is supported by figures 7.3 to 7.5. Figure 7.3 shows that for the uniform random PSO, the Sobol sequence PSO and NSM PSO, the best score per iteration fails to improve after initialization. Additionally, figure 7.4 shows that the distance from the optimum also did not improve after initialization for uniform random, Sobol and NSM strategies. As illustrated in figure 7.5, after an initial drop, the average diversity of the swarm stays relatively high for the remainder of the search. This implies that the swarms converged to a local minimum that was found within the first few iterations of the search. However, the sustained high diversity indicates that the swarms were unable to exploit further within the discovered region.

Both CVT and the subspace-based method exhibited lower swarm diversity than the other methods, both initially and throughout the search. This may have been an advantage: for the functions on which the CVT method outperformed the subspace-based

method, the CVT swarm exhibited lower diversity than the subspace-based swarm. The inverse also applies: for functions on which the two methods performed similarly, the diversity profiles of CVT and subspace were very similar (see figures 7.6 to 7.9).

Low initial diversity implies that particles were initialized close together. The momentum components of the particles' velocities would thus be smaller (since any possible attractors are not far away and velocities are initialized to zero), enabling the particles to perform fine-grained exploration in a selected region.

The subspace method also causes the swarm to have a low initial diversity due to the way in which the scalar t is chosen. Although t (the scalar by which the direction vector $\mathbf{d}$ is scaled) is chosen randomly, $t\mathbf{d}$ must still be approximately within the bounds of the search space. If the value of $\mathbf{d}$ is relatively large for only one dimension, this will force t to be small, thereby confining the particle position to a smaller initial space in all the other dimensions. The rest of this section is dedicated to examining CVT and the reasons for its small initial diversity.

The aim of CVT is to distribute its centroids evenly throughout the search space. Based on the initial swarm diversity, the algorithm certainly does not produce an even spread of particles. For further investigation, two aspects of the CVT centroids were measured: the average distance between centroid pairs, PD, and the average component-wise distance between centroid pairs, PD_C, as defined by equations (7.10) and (7.11) below:

$$\begin{aligned} PD &= \frac{\sum_{\textit{each unique i,k pair}} \|\mathbf{C}^i - \mathbf{C}^k\|}{\textit{total number of unique pairs}} \\ &= \frac{1}{\binom{|C|}{2}} \sum_{i=1}^{|C|} \sum_{k=i+1}^{|C|} \|\mathbf{C}^i - \mathbf{C}^k\| \\ &= \frac{2}{|C|(|C|-1)} \sum_{i=1}^{|C|} \sum_{k=i+1}^{|C|} \|\mathbf{C}^i - \mathbf{C}^k\| \end{aligned} \tag{7.10}$$

where $\mathbf{C}^i$ denotes the i-th centroid, $|C|$ denotes the total number of centroids and $\|.\|$ denotes Euclidean distance. The total number of unique pairs is denoted by $\binom{|C|}{2}$, which is the total number of ways in which unique pairs can be chosen from a set of size $|C|$.

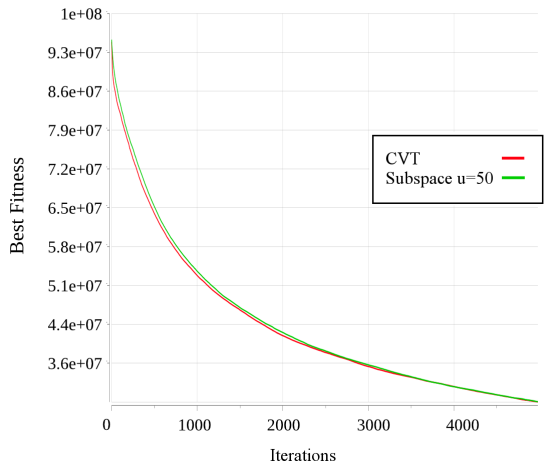


Figure 7.6: Similar fitness of CVT and subspace-based PSO on F17 with n=1000

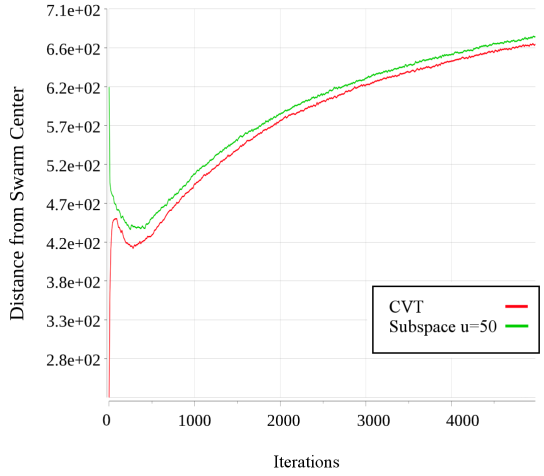


Figure 7.7: Similar diversities of CVT and subspace-based PSO on F17 with n=1000

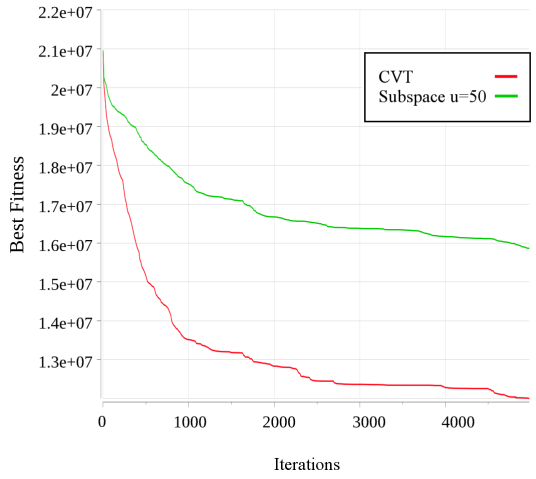


Figure 7.8: Dissimilar fitness of CVT and subspace-based PSO on F6 with n=1000

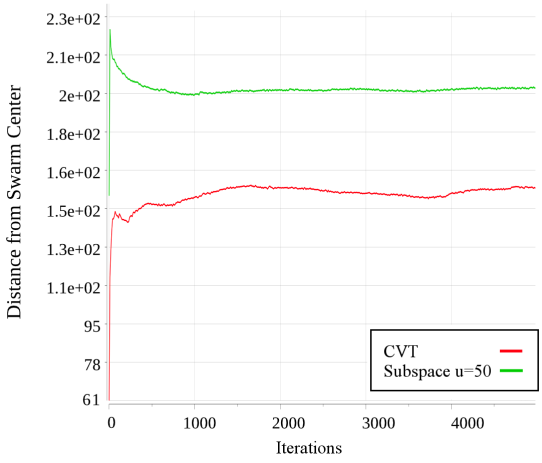


Figure 7.9: Dissimilar diversities of CVT and subspace-based PSO on F6 with n=1000

The average component-wise distance between centroid pairs is given by:

$$PD_C = \frac{\sum_{each\ unique\ i,k\ pair} \sum_{j=1}^{n} |C_j^i - \mathbf{C}_j^k|}{n \times total\ number\ of\ unique\ pairs}$$
$$= \frac{2}{n|C|(|C|-1)} \sum_{i=1}^{|C|} \sum_{j=i+1}^{|C|} \sum_{j}^{n} |\mathbf{C}_j^i - \mathbf{C}_j^k| \tag{7.11}$$

Consider figure 7.10 which shows the average Euclidean distance between every pair of centroids, for various dimensionalities. The higher dimensional centroids have larger initial distance, but the distance decreases sharply as the CVT iterations progress. Figure 7.11 plots the ratio between the final and initial inter-centroidal distances for every dimensionality. As dimensionality increases, the distance between centroids generally becomes smaller, relative to the initial distances.

Additionally, figure 7.12 shows that the average distance between any two dimensions is large for low-dimensional problems and small for high dimensional problems, also suggesting that the centroids are moving closer together.

Recall that the CVT method makes use of sample points to determine the cell centroids and to ensure that they are distributed evenly throughout the search space. However, the number of sample points remain fixed regardless of the problem dimensionality. One explanation for the uneven spread is that for high dimensional problems, the chosen number of sampling points are not enough to ensure even distribution of the cell centroids, particularly if the number of iterations also remains fixed. However, a series of experiments that used increased sample size did not produce better distributed centroids.

Another possible explanation lies in the random initialization of the centroids. If centroids are initially clustered, then centroids far from the cluster will be assigned more sample points than the centroids in the cluster. The centroids in the cluster will thus not spread far enough in the limited number of iterations (since no samples far from the cluster are assigned to it). The hyper-volume of the search space grows exponentially, while the number of centroids (or particles) remains constant. The probability of the randomly initialized centroids not being well-spread thus increases as dimensionality increases, causing disproportional sample point assignment and small inter-centroidal distances. Disproportionate sample point assignment can be tested for by measuring the average number of sample points assigned to each centroid. Ideally, the average number of sample points per centroid should be approximately equal to $\frac{q}{s} = \frac{50000}{100} = 500$ for the set of experiments. (The swarm had 50 particles and each particle requires an initial position and an initial personal best position). Figures 7.13 and 7.14 show the number of sample points assigned to each centroid, or the "centroid count", for each of CVT's five iterations. Figure 7.13 shows the centroid count for $n = 10$ and figure 7.14 shows

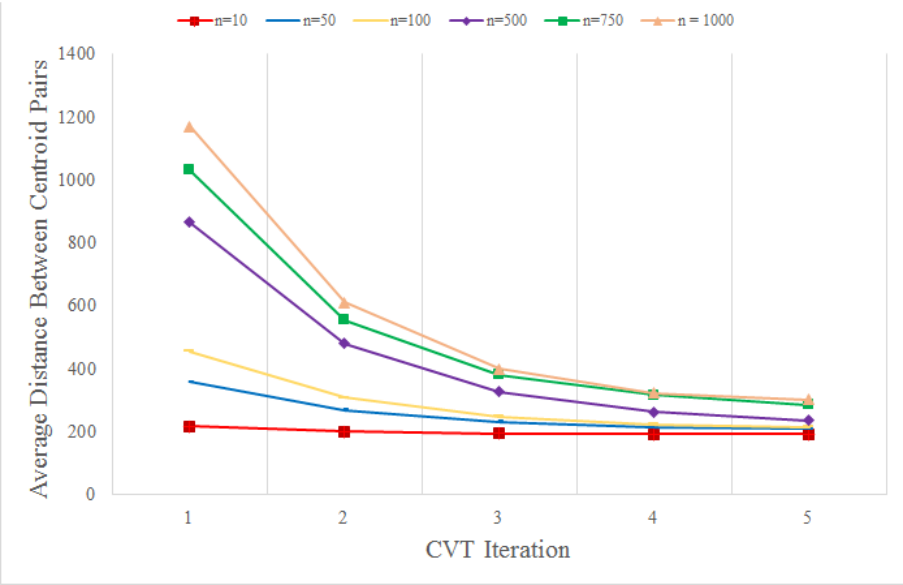


Figure 7.10: Average distance between centroid pairs per CVT iteration, $(L, U) = (-100, 100)$

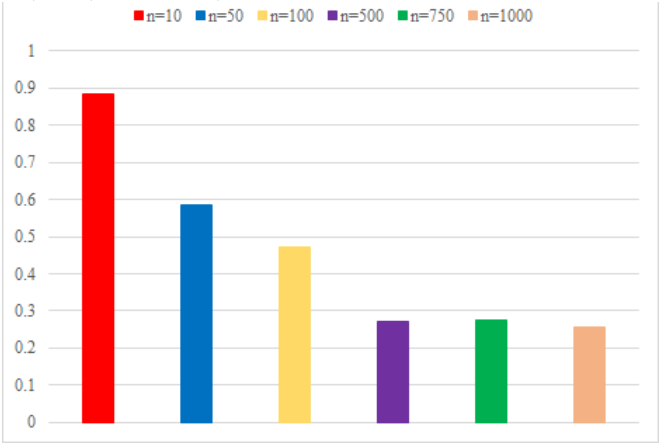


Figure 7.11: Ratio between final and initial distances between centroid pairs

the centroid count for $n = 1000$. As expected, when the dimensionality is large, there is initially a very large range for the centroid counts, due to the uneven distribution of the randomly generated centroids. As the centroids are adjusted with every iteration, the range becomes smaller and eventually centers around 500, as desired. For $n = 10$, the initial and final ranges are much smaller and the mean becomes 500 much faster. Thus,

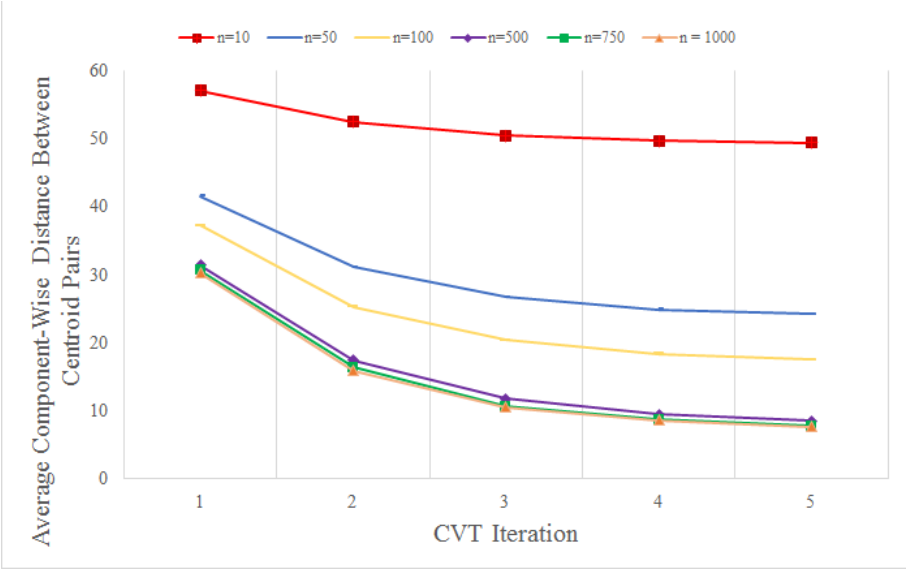


Figure 7.12: Average component-wise distance between centroid pairs per CVT iteration, $(L, U) = (-100, 100)$

disproportionate centroid counts does not explain the low initial swarm diversity.

Another explanation may lie in the norm used to assign samples to centroids. The sum of sufficiently many random samples of $\mathbf{x}$ and $\mathbf{y}$ will cause $\|\mathbf{x} - \mathbf{y}\|$ to approach normal as $n \to \infty$ for both the ℓ_1 and ℓ_2 norms ($\|.\|_1$ and $\|.\|_2$). This may cause the centroids to be normally distributed and biased towards the center of the search space. Future work may try to apply the maximum norm ($\|.\|_\infty$) when assigning samples to centroids.

Regardless of why the CVT strategy causes particles to be initialized close together, the technique performs well. Unfortunately, the computational cost of essentially performing k-means clustering with 50000 samples in high dimensional spaces is not negligible. For every iteration of CVT, the distance between every sample point and every centroid must be calculated in order for the sample point to be assigned to the closest centroid, which is particularly expensive in high dimensional spaces. Then, the arithmetic mean of all samples assigned to a given centroid must be calculated used to transform the centroid. Using big-O notation, the time complexity of the clustering algorithm is $O(qsin)$ [32] where q is the number of sample points, s is the number of centroids, i is

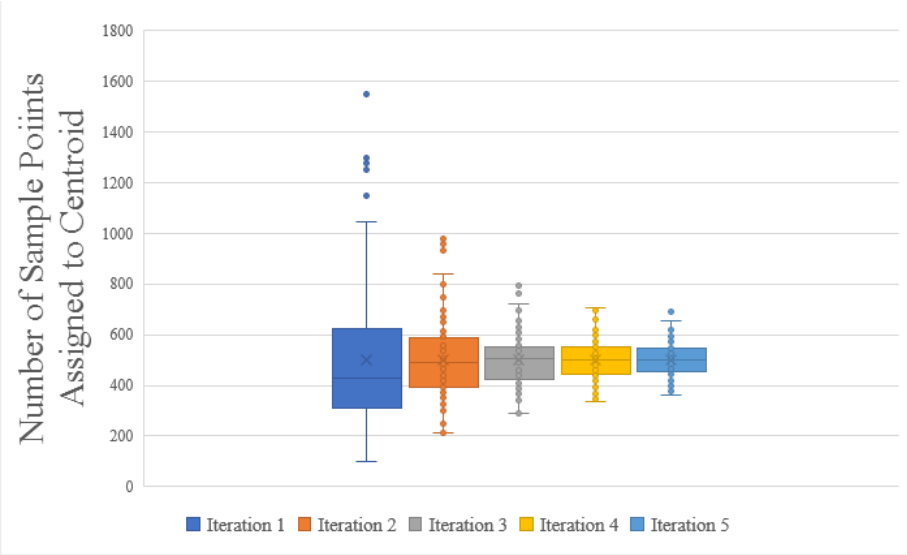


Figure 7.13: Number of sample points assigned to each centroid for each CVT iteration (n=10)

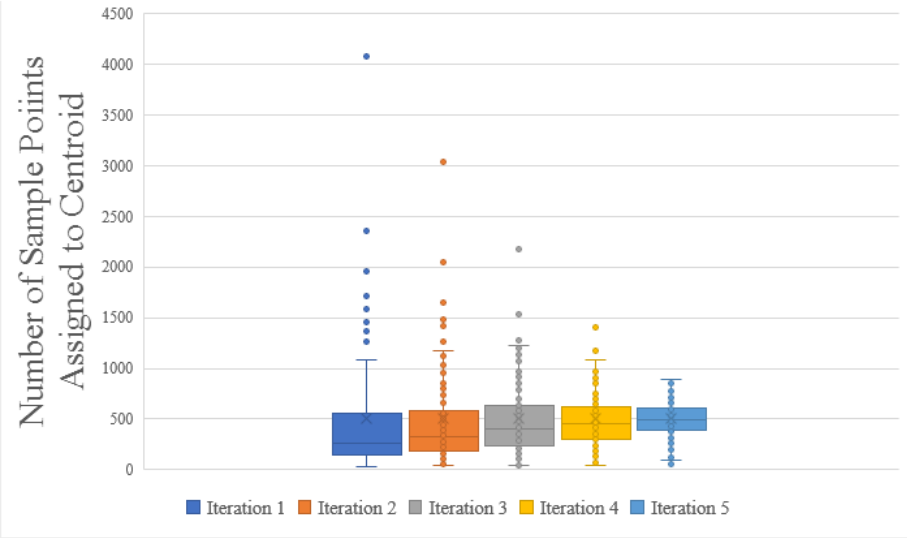


Figure 7.14: Number of sample points assigned to each centroid for each CVT iteration (n=1000)

the number of iterations for which CVT is run and n is problem dimensionality, as usual. By comparison, the subspace-based strategies performed the same as CVT on more than three-quarters of the benchmark suite and has much lower computational cost.

7.4.2 Different Seed Set Sizes

This section briefly examines the effect of different seed set sizes on the subspace-method's performance.

Figure 7.17 shows the performance of subspace-based PSOs for different values of u, across varying dimensionality. For the low dimensional problems, it is not possible to generate u linearly independent vectors if $u > n$. The combinations of n and u that do not exist are colored with horizontal lines.

Clearly, $u = 1$ performed the worst across all dimensionalities. Since all the subspace-based methods exhibited very similar diversity profiles (see figures 7.15 and 7.16 for typical examples), the reason for its behaviour cannot simply be the position confinement brought about by t. (As mentioned previously, the value t by which the direction vector $\mathbf{d}$ is scaled, must be chosen so that $t\mathbf{d}$ is within the bounds of the search space. If $\mathbf{d}$ is very large in a few dimensions, then t is forced to be small, thereby confining the particle in all other dimensions).

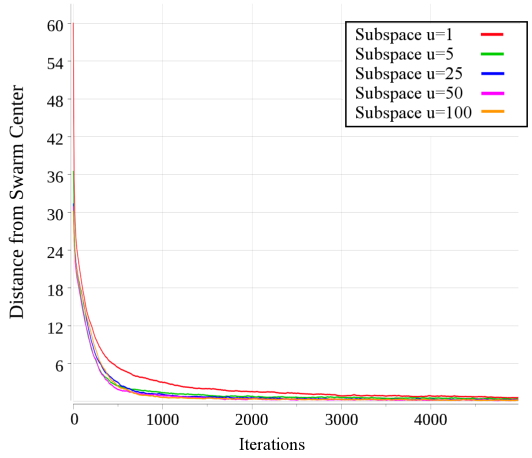


Figure 7.15: Typical swarm diversity profile (F10 with n=1000)

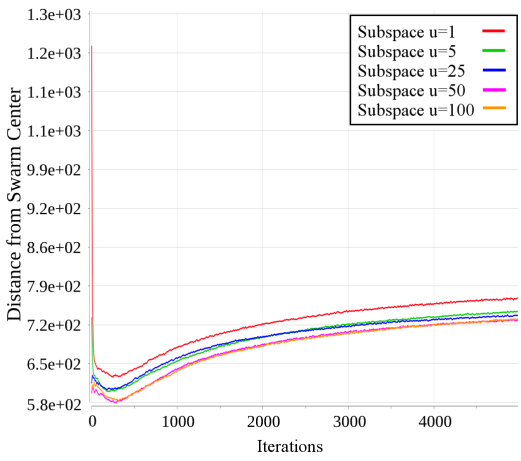


Figure 7.16: Typical swarm diversity profile (F11 with n=1000)

Thus, the difference in performance must be due to the degree of independence among the particles (brought about by the different u-values). Although the initial range within which particles are initialized becomes smaller as u increases, the initial subspace within

which the swarm is initialized becomes larger. Initializing it in a larger subspace gives the swarm more freedom of movement, as described in Chapter 6, decreasing the chances of premature convergence.

In general, larger seed sets performed better as problem dimensionality increased. Although $u = 100$ was never the best performing subspace-strategy, the trend indicates that it would have performed well on even higher dimensional functions. It is interesting that the optimal number of seed vectors for a given dimensionality is small relative to the dimensionality because it implies that a swarm with span much smaller than the search space can perform well in high dimensions. The ratio between optimal u and dimensionality becomes smaller as n increases (see figures 7.18 and 7.19).

The trend indicated by figure 7.19 implies that the optimal ratio between u and dimensionality will converge to a small, constant value.

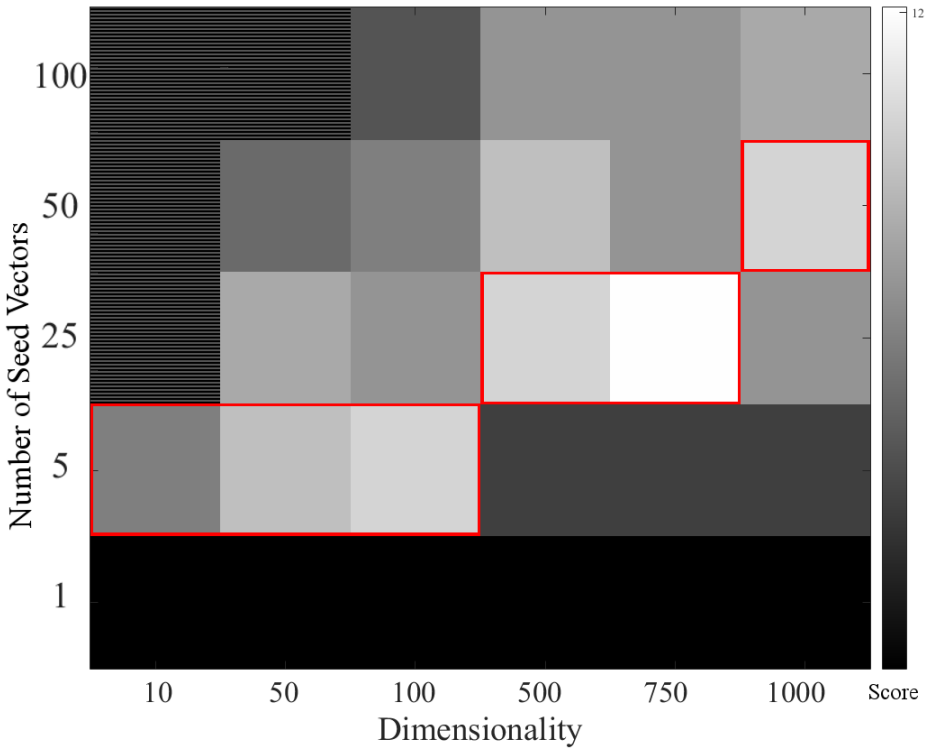


Figure 7.17: Performance of different seed set sizes as dimensionality varies (lighter is better)

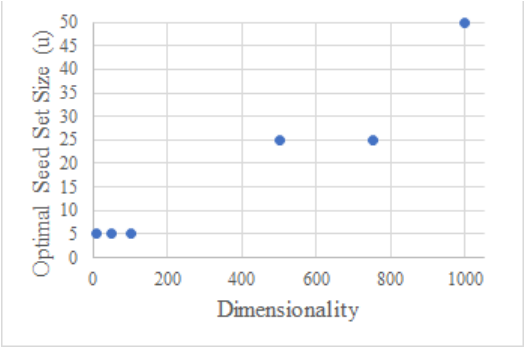


Figure 7.18: Optimal seed set size for each dimensionality

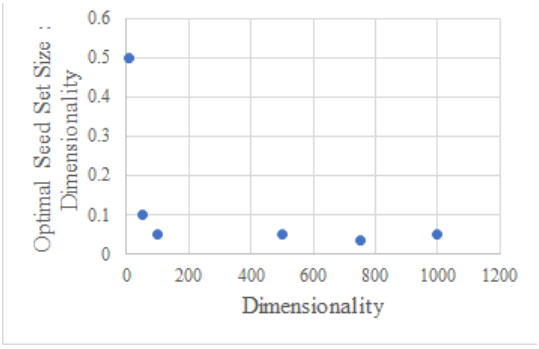


Figure 7.19: Ratio of optimal seed set size to problem dimensionality

7.4.3 Restricted Uniform Random Initialization

The previous section observed that the best-performing initialization strategies both initialize the particles close together, so that the swarm has a low initial diversity (and thus also low initial velocities). The question arises whether the success of the methods can be attributed solely to the low spread of initial positions, since the successful methods invariably had lower diversity than the other methods, both initially and throughout the search.This section determines whether simply initializing the particles in a very small area inside the search space is sufficient for good performance.

Table 7.7: Fraction of search space for particle initialization (τ)

Dimensions	τ
10	0.65
50	0.4
100	0.31
500	0.35
750	0.46
1000	0.55

For a given dimensionality, the maximum initial component-wise (MIC) distance when using the CVT strategy was calculated for each benchmark function (see table 7.8

Table 7.8: Maximum initial component-wise distance for CVT with $n = 10$

Function	Range	MIC	τ	Function	Range	MIC	τ
F1	200	130	0.65	F11	60	38.9	0.65
F2	10	6.50	0.65	F12	200	130	0.65
F3	60	39.13	0.65	F13	200	130	0.65
F4	200	131	0.65	F14	200	130	0.65
F5	10	6.49	0.65	F15	10	6.56	0.66
F6	60	38.9	0.65	F16	60	39.2	0.65
F7	200	129	0.64	F17	200	129	0.64
F8	200	130	0.65	F18	200	129	0.64
F9	200	131	0.65	F19	200	130	0.65
F10	10	6.51	0.65	F20	200	129	0.64

which provides the MIC distance for $n = 10$ as an example). The MIC distance is then expressed as a fraction of $[L, U]$ and averaged across all the benchmark functions. This produced a measure of the component-wise range exhibited by the CVT strategy for a given dimensionality. Particles are then initialized uniform randomly within a region at the center of the search space, where the size of the region is determined by the calculated component-wise range. The idea is expressed more formally below.

Let the MIC distance for function k be denoted by MIC_{F_k}, where $k \in 1, ..., K = 20$ for the benchmark suite. Then the fraction of the search space within which a particle will be initialized (denoted by τ) is given by

$$\tau = \frac{1}{K} \sum_{k=1}^{K} \frac{MIC_{F_k}}{U_j - L_j} \tag{7.12}$$

Particle i will be initialized in each dimension j by

$$x_{i,j}^0 \sim U\left(\frac{U_j + L_j}{2} - \tau(U_j - L_j), \frac{U_j + L_j}{2} + \tau(U_j - L_j)\right) \tag{7.13}$$

$$= U(C_j - \tau(U_j - L_j), C_j + \tau(U_j - L_j)) \tag{7.14}$$

$$= U(\tilde{L}_j, \tilde{U}_j) \tag{7.15}$$

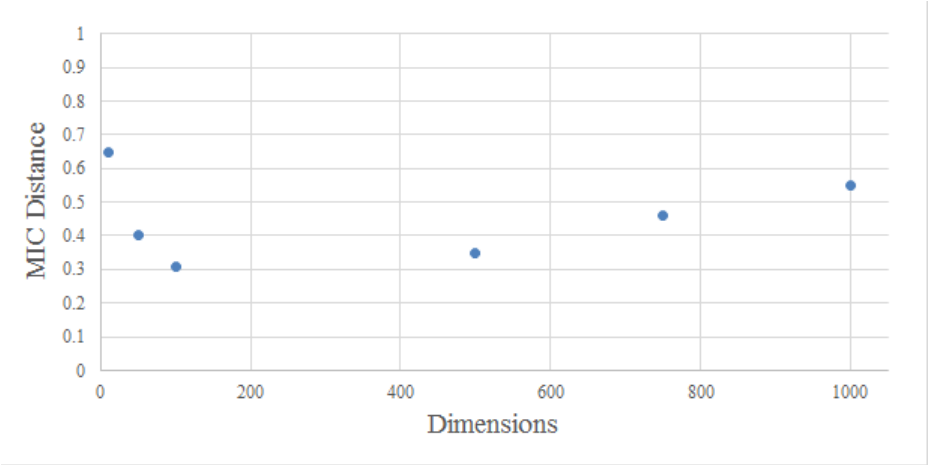


Figure 7.20: Maximal initial component-wise distance between centroids for CVT

where C_j denotes the center of the search space in dimension j. This initialization scheme will be referred to as restricted uniform random (RUS). The values of τ used for each dimensionality are given in table 7.7.

Since the new region of initialization excludes regions near the boundaries, this method will bias the particles towards solutions near the center of the search space. However, the benchmark suite shifts the optimum randomly in each dimension, so the strategy is not biased towards the benchmark suite. Future work may include randomly choosing the center of the region within which particles are initialized.

It is interesting to observe the MIC distance as a function of dimensionality (see figure 7.20). The MIC distance is large for low dimensional problems, decreases as n approaches 100 and then increases again for the remainder of the n range tested. The maximum distance in any dimension between any pair of particles is less than half of the search space, which indicates that the centroids are not well distributed (providing additional support for the observations of the previous section, regarding the spread of CVT's centroids).

In small dimensions, the number of samples and iterations work well for the generated centroids to be spread. However, as dimensionality increases, the component-wise distance between centroids becomes smaller (as seen in figure 7.12), causing the drop in MIC distance observed for $n = 50$ and $n = 100$. Once the dimensionality becomes

sufficiently large, the MIC distance increases again because the chance of obtaining two centroids that are far away from each other in at least one dimension increases as the problem dimensionality increases. Thus, the maximum component-wise distance eventually grows as dimensionality increases.

Figure 7.21 shows the result of comparing the restricted uniform random initialization with uniform initialization (across the entire search space), CVT and the subspace-based method. Although the restricted uniform random method performed the best for $n = 100$ (for which its range was the smallest), it did not generally perform better than the other strategies as dimensionality increased. It did perform better than uniform initialization for the higher dimensional problems ($n \geq 100$).

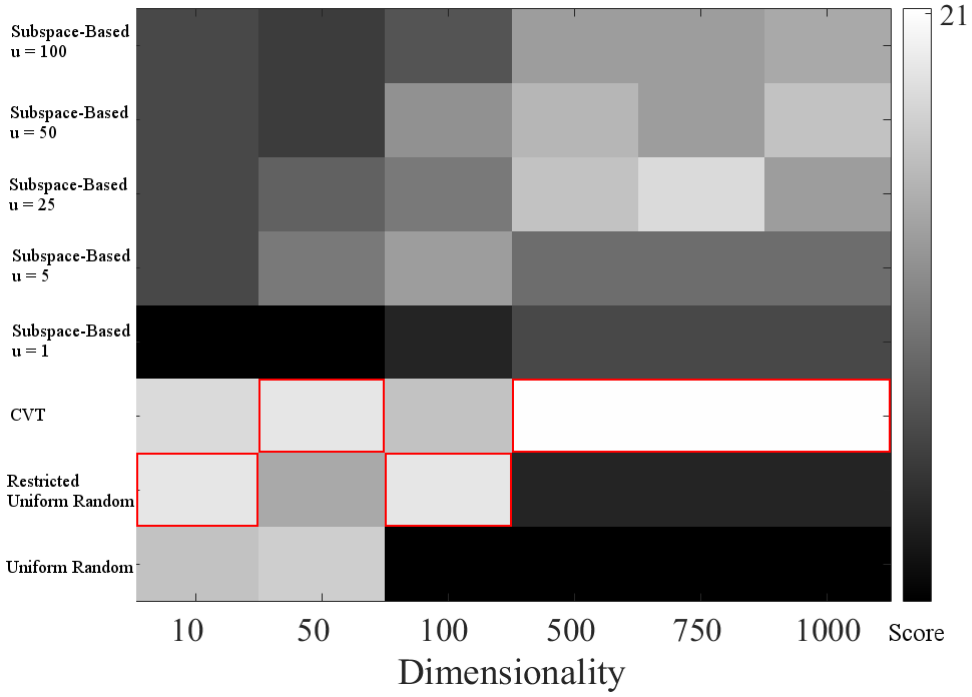


Figure 7.21: Performance of restricted uniform random as dimensionality varies (lighter is better)

Figures 7.22 to 7.25 show that the diversity of RUS was still higher than CVT and the subspace-based method, especially as problem dimensionality increased. It may be that using the maximum initial component distance is not restrictive enough. As explained

before, increasing dimensionality increases the chances of obtaining two centroids that are far away from each other in one dimension, even though most of the centroids are not far apart. Thus using the average initial component (AIC) distance may be more effective in limiting the swarm's initial diversity. An alternative strategy is to use the τ for $n = 100$, but applying it to problems with $n = \{10, 50, 100, 500, 750, 1000\}$. Since $n = 100$ produced the smallest τ, it may strike a good balance between emulating the restrictive nature of CVT without being affected by the probability of obtaining large MIC as n increases.

7.4.4 Roaming Behaviour

This section considers the question of whether the initialization strategies were successful in preventing particle roaming. Although the subspace-based methods and the CVT method were the best performing initialization strategies, figures 7.26 and 7.27 paint a dire picture of particle wandering.

Figures 7.26 and 7.27 show the fraction of the swarm that was out of bounds per iteration of the search. All the initialization strategies except CVT, subspace and RUS were out of bounds for almost the entire duration of the search on all the functions. The RUS strategy consistently exhibited less roaming than uniform random, NSM and Sobol, but more than CVT and the subspace-based method.

CVT and the subspace method showed particle roaming similar to figure 7.26, for 13 of the 20 benchmark functions, where at least 75% of the swarm was always out of bounds (in some cases, such as F13, more than 90% of the swarm was out of bounds after the first 100 iterations). On the more hopeful end of the spectrum, CVT and subspace showed behaviour similar to figure 7.27 for six of the functions, where at least 75% of the swarm eventually returned to the search space. Function F5 (figure 7.28) was the exception, with the subspace-based method following the profile of figure 7.26 and CVT following figure 7.27's profile.

Thus, the initialization strategies were not effective in preventing particle roaming for more than half of the cases.

Figures 7.29 and 7.30 plot the two typical profiles for the average number of dimensions out of bounds for every iteration. The CVT and subspace-based PSOs have fewer

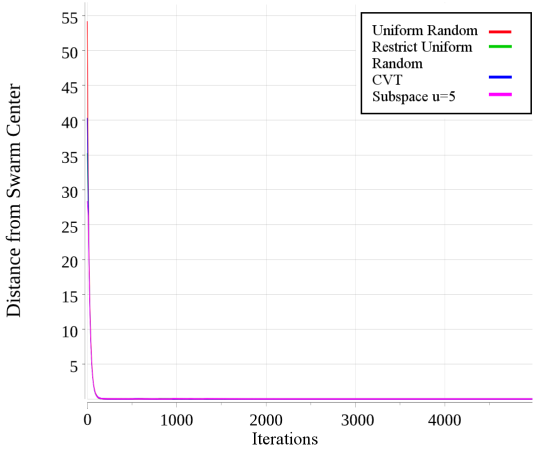


Figure 7.22: Swarm diversity of restricted uniform random on F16 with $n = 10$

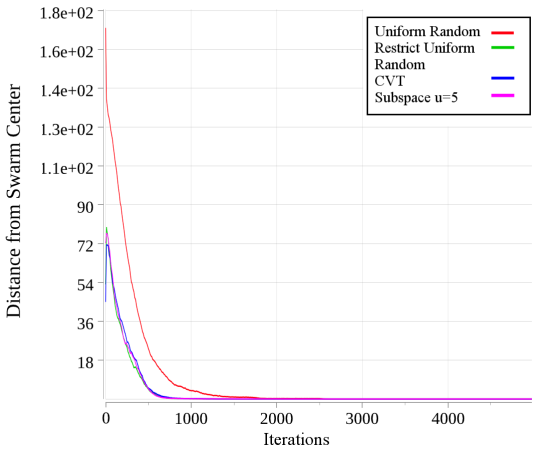


Figure 7.23: Swarm diversity of restricted uniform random on F16 with $n = 100$

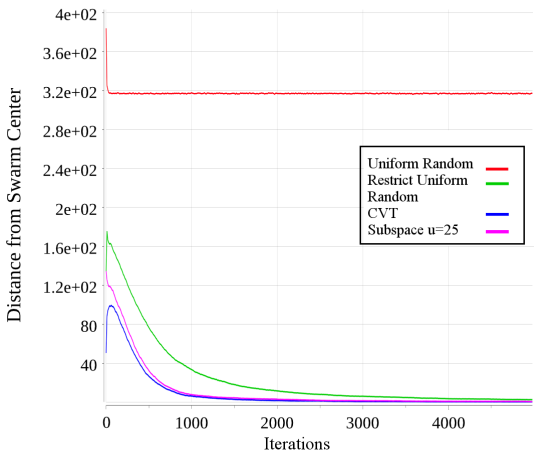


Figure 7.24: Swarm diversity of restricted uniform random on F16 with $n = 500$

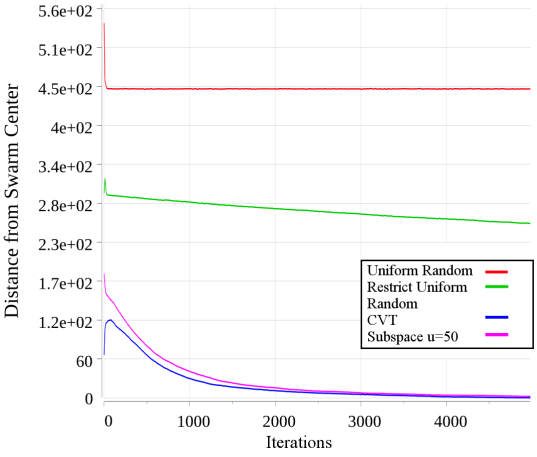


Figure 7.25: Swarm diversity of restricted uniform random on F16 with $n = 1000$

dimensions out of bounds than the other initialization strategies. Disconcertingly, the number of violated boundaries increased as the search progressed for a number of the functions (see figure 7.29). Upon further examination, the functions for which the particles failed to return to the search space were exactly those for which the number of violated boundaries increased as the search progressed. This is indicative of an under-

lying problem: although roaming is mitigated to some extent, the swarm continues to leave the search space instead of returning, just at a slower rate. A good choice in initialization strategy may thus mitigate particle roaming to some extent, but not reliably. It thus fails to address the underlying force which drives particles out of the search space.

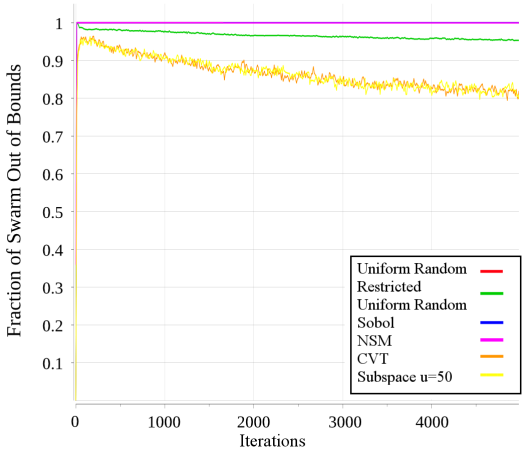


Figure 7.26: Fraction of swarm out of bounds on F12 with $n = 1000$ (profile 1)

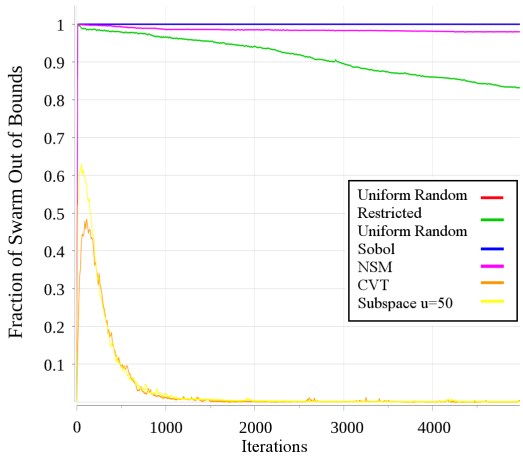


Figure 7.27: Fraction of swarm out of bounds on F3 with $n = 1000$ (profile 2)

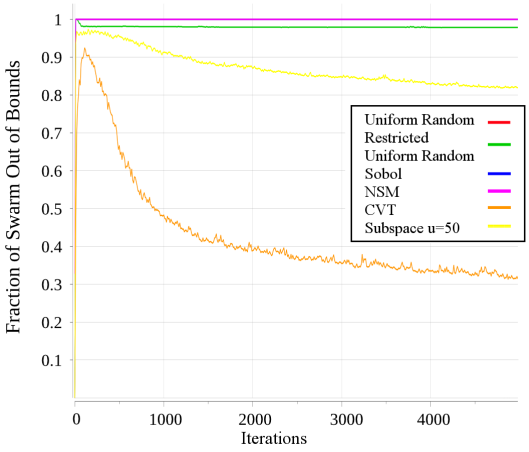


Figure 7.28: Fraction of swarm out of bounds on F15 with $n = 1000$ (atypical profile)

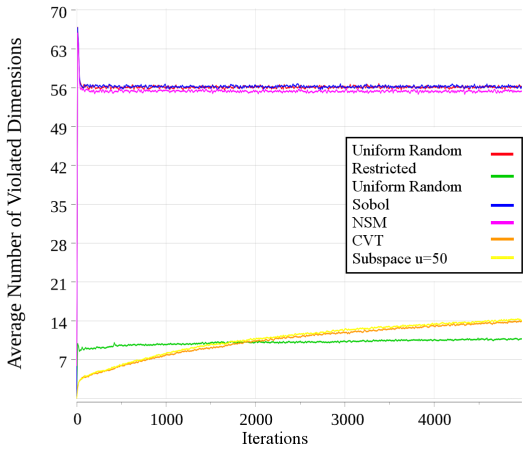


Figure 7.29: Average number of dimensions out of bounds on F12 with n=1000 (profile 1)

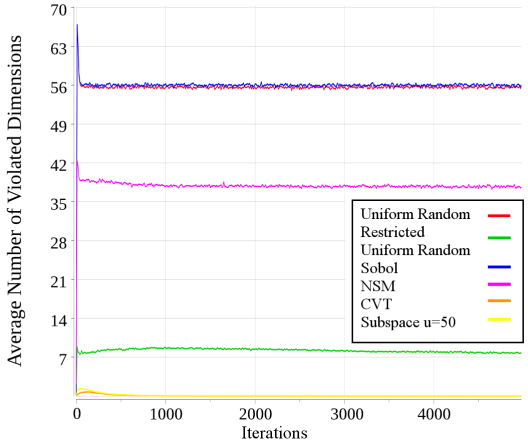


Figure 7.30: Average number of dimensions out of bounds on F3 with n=1000 (profile 2)

7.5 Conclusion

This chapter examined the effect of the swarm's initialization strategy on its performance in high dimensions. It was found that the choice in initialization strategy influences the roaming behaviour and general performance exhibited by the swarm.

The chapter also examined a novel initialization strategy which restricts the swarm's initial positions to a small subspace of the search space. The proposed initialization strategy thereby forces the swarm to focus on exploration within a small subspace of the search space, rather than attempting to explore the entire search space. The optimal number of basis vectors (i.e. the seed set size) used to generate the initial subspace was found to depend on the problem dimensionality, with higher dimensional problems requiring more basis vectors for good performance. The ratio between the optimal number of basis vectors and problem dimensionality was found to be asymptotically decreasing.

The subspace-based strategy outperformed uniform random initialization, Sobol sequence initialization, and NSM on the high dimensional spaces. The best performing initialization strategy was CVT, which differs from previous findings where the subspace-based initialization strategy performed better. However, using CVT in high dimensional spaces becomes prohibitively expensive in high dimensional spaces. Since the subspace-

based strategy was statistically equivalent to CVT for more than three quarters of the benchmark suite and is much less computationally complex than CVT, the subspace strategy remains useful for high dimensional problems.

The chapter proposes that low initial swarm diversity may be beneficial to the swarm's searching ability in high dimensional problems because the particles will have lower momentum, enabling the swarm to perform more fine-grained exploration within the initialized region.

The chapter also examined the possibility that the reason for CVT and the subspace-based strategy's success is due entirely to the low initial particle spread. The theory was tested by measuring the maximum initial component distance between the particle positions generated by CVT and using the resulting range to initialize particles uniform randomly within a restricted region of the search space. The restricted initialization method did not perform as well as CVT and the subspace-based method, implying that there may be some other advantage to the way in which these initialization strategies work than simply initializing particles close together.

Chapter 8

Conclusions

This chapter concludes the main part of the thesis. Section 8.1 summarizes the findings of the thesis and section 8.2 discusses directions of future work.

8.1 Summary

This work considered the behaviour of standard particle swarm optimization (PSO) algorithms on high dimensional problems. Chapter 3 showed that PSO exhibits unwanted, continuous roaming as a result of uncontained velocity explosion in the first few iterations. The roaming behaviour was shown to be independent of swarm size and roaming behaviour was exhibited on all of the benchmark functions.

The remainder of the thesis showed that strategies which emphasized locally exploitative or granular searching performed the best in high dimensional spaces. Rather than encouraging exploration and attempting to find the optimal solution in a search space which grows exponentially in hyper-volume, a more feasible approach is to focus on finding a good solution in some smaller region of the search space. Particles were restricted in a number of different ways such as clamping velocity to force small step sizes, selecting values for the particles' inertia weight and acceleration coefficients that result in smooth movement trajectories and high momentum, introducing coupling between problem dimensions to reduce the swarm's degrees of freedom and initializing the swarm within a small region of the search space.

Chapter 4 showed that velocity clamping can force particles to search in a granular manner by reducing the step sizes of particles. As problem dimensionality increased, the optimal maximum velocity decreased, indicating that smaller step sizes perform better in high dimensional spaces. It was shown that the optimal maximum velocity converges to a very small positive constant. The roaming behaviour of swarms with small maximum velocities was reduced, but not entirely. Although the swarms with small maximum velocities mostly succeeded in confining the particles to the search space, the swarms were also vulnerable to premature convergence.

The influence of the inertia weight and acceleration coefficients were considered in Chapter 5. It was shown that restricting the variance of particle positions reduces the roaming behaviour of the particles. It was also observed that the best performing combinations had high inertia weights and low acceleration coefficients, which reduces the influence of updates to particles' local and global attractors. The particles trajectories are thus less likely to rapidly change direction, which reduces the magnitude of the velocity explosion. Chapter 5 also considered the influence of the inertia weight and acceleration coefficients on the movement patterns of the swarm. It was found that coefficients that brought about smooth trajectories with highly correlated particle positions performed the best in high dimensional spaces. Such trajectories generally had small base frequencies and relatively low variances.

The importance of the stochastic scaling components, $\mathbf{r}_1, \mathbf{r}_2$ was considered in Chapter 6. The chapter provided theoretical analysis that illustrated the importance of applying component-wise stochastic scaling in the velocity update equation, especially when the problem dimensionality exceeds the swarm size. The chapter also introduced grouping strategies, which divide the problem variables into a number of groups and apply the same stochastic scalar to all components within a given group. This reduces the swarm's degrees of freedom by introducing coupling between variable dimensions. The most effective approach was to restrict the swarm's movement initially, and then to gradually allow more freedom to encourage continued improvement. This result is opposite to the typical behavioural profile, which first performs exploration and then behaves exploitatively.

Chapter 7 examined the influence of the swarm's initialization strategy on its performance in high dimensional spaces. It was found that the initialization strategy strongly

affects the swarm's behaviour. In low dimensional spaces, strategies that maximized the swarm's initial spread performed well. However, as the dimensionality increased, strategies that restrict the swarm's initial positions to a small region within the subspace performed better.

In high dimensional spaces, initialization strategies that generate positions within a small region of the search space perform better than strategies that maximize the swarm's initial spread. Such strategies help to prevent the initial velocity explosion because all the particles are near one another initially.

Chapter 7 also introduced a novel initialization strategy that restricts the swarm's initial positions to a small subspace of the search space. The proposed initialization strategy forces the swarm to focus on exploration within a small subspace of the search space, rather than attempting to explore the entire search space. The optimal number of basis vectors (i.e. the seed set size) used to generate the initial subspace was found to depend on the problem dimensionality, where higher dimensional problems could be better solved by increasing the size of the seed set. The proposed initialization strategy performed significantly better on high dimensional problems than the strategies that maximized the swarm's initial spread. Although the initialization strategy employing centroidal Voronoi tessellations performed better than the proposed strategy on four of the benchmark problems, the two strategies were tied in performance for the remaining sixteen problems in the benchmark suite and the proposed strategy is considerably less computationally expensive.

Chapter 7 also investigated the feasibility of simply initializing the particles within a smaller, bounded region at the center of the search space based on the initial maximum spread produced by the centroidal Voronoi tessellation strategy.

8.2 Future Work

The obvious avenue to pursue in future work is the application of boundary handling techniques, which was beyond the scope of this thesis. Boundary handling techniques address particle roaming by preventing particles from leaving the search space or repairing particle positions that are outside the search space.

Chapter 4 noted that although velocity clamping may be used to mitigate particle roaming to some extent, swarms with small maximum velocities were vulnerable to premature convergence. Future work may examine other methods of velocity clamping such as decreasing the maximum velocity throughout the search. This may address the problems exhibited by the velocity clamping with premature convergence, since particle step sizes are initially large which facilitates some exploration. The maximum velocity becomes smaller as the search progresses, forcing particles to exploit locally towards the end of the search.

Chapter 5 examined the relationship between the inertia weight and acceleration coefficients, and swarm movement patterns. Based on these observations, future work may develop a self-adaptive PSO that calculates the appropriate inertia weight and acceleration coefficients to bring about a desired movement pattern. The inertia weight and acceleration coefficients can be recalculated whenever the global and personal best positions are updated, to maintain the desired variance in particle positions.

The grouping strategies proposed in Chapter 6 were either static, where the dimensions assigned to a group were never changed, or dynamic, where the dimensions assigned to a group were changed with every iteration. In future, an intermediate approach may be examined, where the dimensions assigned to a particular group are only shuffled once every hundred iterations or where shuffles are triggered by some other criteria such as the number of personal best updates or particle step sizes.

The grouping strategies tested in Chapter 6 included using a fixed number of groups, linearly increasing the number of groups and linearly decreasing the number of groups. Future work may include developing different grouping strategies, such as logarithmically increasing the number of groups. Hybrid grouping strategies, where the number of dimension groups increases for some members of the swarm and decreases for other members of the swarm may have applications in dynamic optimization.

Chapter 7 proposed an initialization strategy that initialized the swarm within a small subspace in the search space. Future work may combine the proposed initialization strategy with the dimension grouping strategies in Chapter 6 to solve niching problems. Swarms that are initialized in disjoint subspaces and restricted, at least partially, to the initial subspace, may be effective in identifying different optima within the search space.

Chapter 7 also examined the idea of initializing the swarm within a small region of the search space. The calculated bounds for the small region were based on the maximum spread exhibited by the best-performing initialization strategy. Future work may extend the idea and utilize even smaller bounds for initialization (for instance, by using the average spread of the best-performing strategy).

In general, the research presented in this thesis indicated that the initial velocity explosion and unwanted roaming behaviour must be mitigated for PSO to be useful in the field of large scale optimization (LSO). Future work that attempts to apply PSO on high dimensional problems must thus address these two problems.

Bibliography

[1] A. Auger, N. Hansen, N. Mauny, and M. Schoenauer R. Ros. Bio-inspired continuous optimization: The coming of age. Invited Talk at the IEEE Congress on Evolutionary Computation. 2007.

[2] M. Bataineh. On chebyshev array design using particle swarm optimization. *Journal of Electromagnetic Analysis and Applications*, 3(6):213–219, 2011.

[3] R. Bellman. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[4] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, pages 217–235. Springer-Verlag, 1999.

[5] M. R. Bonyadi and Z. Michalewicz. Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE Transactions on Evolutionary Computation*, 20(5):814–819, Oct 2016.

[6] M. R. Bonyadi and Z. Michalewicz. Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 21(3):378–390, June 2017.

[7] M. R. Bonyadi and Z. Michalewicz. Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 21(3):378–390, June 2017.

[8] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 120–127. IEEE Computer Society, 2007.

[9] S. Chen, J. Montgomery, and A. Bolufé-Röhler. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence*, 42(3):514–526, April 2015.

[10] S. Chen, J. Montgomery, and A. Bolufé-Röhler. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence*, 42(3):514–526, April 2015.

[11] R. Cheng and Y. Jin. A competitive swarm optimizer for large scale optimization. *IEEE Transactions on Cybernetics*, 45(2):191–204, Feb 2015.

[12] C. W. Cleghorn and A. P. Engelbrecht. Particle swarm convergence: An empirical investigation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2524–2530, July 2014.

[13] C. W. Cleghorn and A. P. Engelbrecht. Particle swarm convergence: Standardized analysis and topological influence. In *Proceedings of the 9th International Conference on Swarm Intelligence*, pages 134–145. Springer International Publishing, Sept 2014.

[14] C. W. Cleghorn and A. P. Engelbrecht. Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. *Swarm Intelligence*, Sep 2017.

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

[16] V. N. Costa, M. R. Monteiro, and A. C. Zambroni de Souza. Particle swarm optimization applied to reactive power compensation. In *Proceedings of the 17th International Conference on Harmonics and Quality of Power*, pages 780–785, Oct 2016.

[17] I. L. Dalal, D. Stefan, and J. Harwayne-Gidansky. Low discrepancy sequences for monte carlo simulations on reconfigurable platforms. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*, pages 108–113, July 2008.

[18] Y. Delice, E. Kızılkaya Aydoğan, U. Özcan, and M. S. İlkay. A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, 28(1):23–36, 2017.

[19] P. Domingos. A few useful things to know about machine learning. *Communications ACM*, 55(10):78–87, October 2012.

[20] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Oct 1995.

[21] R. C. Eberhart and X. Hu. Human tremor analysis using particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, volume 3, pages 1927 – 1930, 1999.

[22] R.C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 84–88 vol.1, 2000.

[23] A. P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, Chichester, England, December 2002.

[24] A. P. Engelbrecht. Roaming behavior of unconstrained particles. In *Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, BRICS-CCI-CBIC '13, pages 104–111. IEEE Computer Society, 2013.

[25] A. P. Engelbrecht. Fitness function evaluations: A fair stopping condition? In *Proceedings of the IEEE Symposium on Swarm Intelligence*, pages 1–8, Dec 2014.

[26] A.P. Engelbrecht. Particle swarm optimization: Velocity initialization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, June 2012.

[27] A.P. Engelbrecht. Particle swarm optimization: Global best or local best? In *Proceedings of the BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC)*, pages 124–135, Sept 2013.

[28] H. Fan. A modification to particle swarm optimization algorithm. *Engineering Computations*, 19(8):970–989, 2002.

[29] P. Faria, J. Soares, Z. Vale, H. Morais, and T. Sousa. Modified particle swarm optimization applied to integrated demand response and dg resources scheduling. *IEEE Transactions on Smart Grid*, 4(1):606–616, March 2013.

[30] E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, Technical Report 4, January 1951.

[31] 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 (CEC)*, pages 431–439, July 2016.

[32] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[33] S. Helwig and R. Wanka. Particle swarm optimization in high-dimensional bounded search spaces. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 198–205. IEEE Computer Society, 2007.

[34] S. Helwig and R. Wanka. Theoretical analysis of initial particle swarm behavior. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature - Volume 5199*, pages 889–898. Springer-Verlag New York, Inc., 2008.

[35] K. Ishaque and Z. Salam. A deterministic particle swarm optimization maximum power point tracker for photovoltaic system under partial shading condition. *IEEE Transactions on Industrial Electronics*, 60(8):3195–3206, Aug 2013.

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

[37] S. Joe and F. Y. Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *Association for Computing Machinery Transactions on Mathematical Software*, 29(1):49–57, March 2003.

[38] L. Ju, Q. Du, and M. Gunzburger. Probabilistic methods for centroidal voronoi tessellations and their parallel implementations. *Parallel Computing*, 28(10):1477 – 1500, 2002.

[39] S.M. Kamalapur and S. Sane. Scalars impact on particle swarm optimization performance. In *Proceedings of the International Conference on Recent Trends in Information, Telecommunication and Computing*, pages 318–320, March 2010.

[40] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, Nov 1995.

[41] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 1671–1676, 2002.

[42] T. Krink, J. S. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. In *Proceedings of the Congres on Evolutionary Computation CEC - Volume 02*, CEC '02, pages 1474–1479. IEEE Computer Society, 2002.

[43] Y. Li, L. Jiao, R. Shang, and R. Stolkin. Dynamic-context cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation. *Information Sciences*, 294:408 – 422, 2015. Innovative Applications of Artificial Neural Networks in Engineering.

[44] Q. Liu. Order-2 stability analysis of particle swarm optimization. *Evolutionary Computation*, 23(2):187–216, June 2015.

[45] Z. Liu. Empirical study of the random number parameter setting for particle swarm optimization algorithm. In *Proceedings of the IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications*, pages 246–252, Sept 2010.

[46] A.P. Engelbrecht M. Omran, A. Slaman. Image classification using particle swarm optimization. In *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, pages 270–374, 2002.

[47] K. Malan and A. P. Engelbrecht. Algorithm comparisons and the significance of population size. *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 914–920, 2008.

[48] J. Matoušek. On the l2-discrepancy for anchored boxes. *Journal of Complexity*, 14(4):527–556, December 1998.

[49] R. Morgan and M. Gallagher. Sampling techniques and distance metrics in high dimensional continuous landscape analysis: Limitations and improvements. *IEEE Transactions on Evolutionary Computation*, 18(3):456–461, June 2014.

[50] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[51] O. Olorunda and A. P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1128–1134, June 2008.

[52] C. C. Paige, M. Rozložník, and Z. Strakos. Modified gram-schmidt (mgs), least squares, and backward stability of mgs-gmres. *Society for Industrial and Applied Mathematics Journal Matrix Analysis and Applications*, 28(1):264–284, May 2006.

[53] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In

Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, pages 400–407, April 2010.

[54] U. Paquet and A. P. Engelbrecht. Particle swarms for linearly constrained optimisation. *Fundamental Informatics*, 76(1-2):147–170, Feb 2007.

[55] K. E. Parsopoulos and M. N. Vrahatis. Initializing the particle swarm optimizer using the nonlinear simplex method. In *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 216–221. WSEAS Press, 2002.

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

[57] D. Poole. *Linear Algebra: A Modern Introduction, Third Edition*. Cengage Learning, Canada, 2011.

[58] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, December 2010.

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

[60] M. Richards and D. Ventura. Choosing a starting configuration for particle swarm optimization. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 2309–2312, July 2004.

[61] J. Riget and J.S. Vesterstrm. A diversity-guided particle swarm optimizer - the arpso. Technical report, 2002.

[62] D. P. Rini, S. M. Shamsuddin, and S. S. Yuhaniz. Article: Particle swarm optimization: Technique, system and challenges. *International Journal of Computer Applications*, 14(1):19–27, January 2011. Full text available.

[63] J. Robinson and Y. Rahmat-Samii. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2):397–407, Feb 2004.

[64] 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*, 39(3):263 – 278, 1996.

[65] I. Schoeman and A. P. Engelbrecht. Effect of particle initialization on the performance of particle swarm niching algorithms. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 560–561. Springer Berlin Heidelberg, 2010.

[66] J.F. Schutte and A.A. Groenwold. Sizing design of truss structures using particle swarms. *Structural and Multidisciplinary Optimization*, 25(4):261–269, Oct 2003.

[67] F. Shahzad, A. R. Baig, S. Masood, M. Kamran, and N. Naveed. *Opposition-Based Particle Swarm Optimization with Velocity Clamping (OVCPSO)*, pages 339–348. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[68] S. Shan and G. G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, Mar 2010.

[69] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, May 1998.

[70] Y. Shi and R. C. Eberhart. *Parameter selection in particle swarm optimization*, pages 591–600. Springer Berlin Heidelberg, Mar 1998.

[71] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, page 1950 Vol. 3, 1999.

[72] G. D. Smith and S. Ebrahim. Data dredging, bias, or confounding. *BMJ (Clinical research ed.)*, 325(7378):14371438, December 2002.

[73] P. N. Suganthan. Particle swarm optimiser with neighbourhood operator. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, page 1962 Vol. 3, 1999.

[74] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the cec2010 special session and competition on large-scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, 2009.

[75] T. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317 – 325, 2003.

[76] B. Tudu, S. Majumder, K. K. Mandal, and N. Chakraborty. Comparative performance study of genetic algorithm and particle swarm optimization applied on off-grid renewable hybrid energy system. In *Proceedings of the Second International Conference on Swarm, Evolutionary, and Memetic Computing - Volume Part I*, SEMCCO'11, pages 151–158. Springer-Verlag, 2011.

[77] F. Van Den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Pretoria, South Africa, South Africa, 2002. AAI0804353.

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

[79] F. van den Bergh and A. P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Science*, 176(8):937–971, April 2006.

[80] E. van Zyl and A. Engelbrecht. Comparison of self-adaptive particle swarm optimizers. In *Proceedings of the IEEE Symposium on Swarm Intelligence*, pages 1–9, Dec 2014.

[81] E. van Zyl and A.P. Engelbrecht. A subspace-based method for pso initialization. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 226–233, Dec 2015.

[82] E.T. van Zyl and A.P. Engelbrecht. Group-based stochastic scaling for pso velocities. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1862–1868, 07 2016.

[83] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Proceedings of the 8th International Work-Conference on Artificial Neural: Networks Computational Intelligence and Bioinspired Systems*, pages 758–770. Springer Berlin Heidelberg, June 2005.

[84] H. Zhang, X. Li, H. Li, and F. Huang. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, 14(3):393 – 404, 2005. International Conference for Construction Information Technology 2004.

[85] A. Zimek, E. Schubert, and H. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, October 2012.

Appendix A

Benchmark Functions

The benchmark suite used throughout the thesis was specifically developed for testing large-scale global optimization algorithms by Tang et. al. [74]. This appendix provides a detailed explanation of the benchmark suite's construction. The first section explains the concept of separability and its relationship to problem difficulty. The section also explains how problems of varying separability may be constructed. Section A.2 defines the basic functions which are used to build the benchmark functions. Section A.3 defines the benchmark suite. Lastly, section A.4 concludes the appendix with a brief summary of the benchmark functions' properties such as modality, domains and separability.

A.1 Separability and Basic Functions

The benchmark suite consists of minimization problems of varying degrees of separability. Separability of a function as defined in [1] is provided below:

Definition A.1. A function $f(\mathbf{x})$ is *separable* if and only if there exist functions $f_1, ..., f_n$, each a function of one variable such that

$$\arg\min_{x_1,...,x_n} \{f(x_1, x_2, ..., x_n)\} = \left(\arg\min_{x_1}\{f_1(x_1),\}, ..., \arg\min_{x_n}\{f_n(x_n),\} \right) \tag{A.1}$$

for the entire domain of f.

Thus, if f is a function of n variables, then it is separable with regards to addition if and only if it can be rewritten as the sum of n, single-variable functions. If $f(\mathbf{x})$ is

a separable function, then its parameters x_i are *independent*. A function that is not separable is called *nonseparable*.

A function may exhibit varying degrees of separability. This is formalized below.

Definition A.2. A nonseparable function is called *m-nonseparable* if at most m of its parameters are not independent.
A nonseparable function is called *fully-nonseparable* if every pair of its parameters are not independent.

Problem separability is often used as a measure of the problem's difficulty. Separable problems considered easy and fully-nonseparable problems are considered difficult. Partially separable problems generally fall between separable and fully-nonseparable problems on the difficulty scale. The partially separable problems contained in the benchmark suite fall into one of three classes. The first class of functions contains a number of dependent variables with all the remaining variables independent. The second class consists of multiple independent subcomponents, with each subcomponent being m-nonseparable. The third class is a combination of these two and consists of a number of independent subcomponents, some separable and some m-nonseparable. The benchmark suite comprises of separable problems, fully-nonseparable problems and partially separable problems from all three classes.

Functions of varying degrees of separability are constructed by dividing the input variables into several groups, each of which can be kept independent or made dependent by means of coordinate rotation [64]. Each group of variables is then evaluated by one of six basic functions and these values are added together to produce the benchmark function's value. The six basic functions are listed below:

1. Sphere Function
2. Rotated Elliptic Function
3. Schwefel's Problem 1.2
4. Rosenbrock's Function
5. Rotated Rastrigin's Function

6. Rotated Ackley's Function

Apart from the Sphere function, the basic functions are nonseparable. The number of variable groups is determined by specifying m, the number of variables in each group. The degree of problem separability is thus determined by the m parameter. The authors proposed a value of $m = 50$, which is used in this thesis. In order to accommodate comparisons between performance on high and low dimensional problems, the value of $m = 10$ is also used on occasion (when comparing among problem dimensionalities as small as $n = 10$).

The random variable grouping may be achieved as follows. Let P be a random permutation of $\{1, 2, ..., n-1, n\}$ and let $\mathbf{x}$ be an n-dimensional variable. Then $\mathbf{x}(P_1 : P_m) = (x_{P_1}, x_{P_2}, ..., x_{P_{m-1}}, x_{P_m})^T$ is a random group of size m chosen from the components of $\mathbf{x}$. The random permutation is used in this manner to index the objective variables and produce random groups of the desired number and size.

The next section A.2 defines the basic functions used to construct the benchmark functions.

A.2 Basic Functions

This section provides the definitions of the basic functions mentioned in section A.1. The notation may be interpreted as follows: n denotes the problem dimensionality, m denotes the group size and $\mathbf{x}$ denotes an input variable or candidate solution.

A.2.1 Sphere Function

The Sphere function is a separable function defined by

$$F_{sphere}(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 \tag{A.2}$$

A.2.2 Rotated Elliptic Function

The Elliptic function is separable and is defined as:

$$F_{elliptic}(\mathbf{x}) = \sum_{j=1}^{n} (10^6)^{\frac{j-1}{n-1}} x_j^2 \tag{A.3}$$

The Elliptic function is multiplied by an orthogonal matrix to form the nonseparable, Rotated Elliptic basic function. This is given by

$$F_{rot\_elliptic}(\mathbf{x}) = F_{elliptic}(\mathbf{Mx}) \tag{A.4}$$

where $\mathbf{M}$ is a $n \times n$ orthogonal matrix.

A.2.3 Schwefel's Problem 1.2

Schwefel's Problem 1.2 is nonseparable and given by

$$F_{schwefel}(\mathbf{x}) = \sum_{j=1}^{n} \left(\sum_{k=1}^{j} x_j \right)^2 \tag{A.5}$$

A.2.4 Rosenbrock's Function

Rosenbrock's Function is nonseparable and given by

$$F_{rosenbrock}(\mathbf{x}) = \sum_{j=1}^{n-1} \left(100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2 \right) \tag{A.6}$$

A.2.5 Rotated Rastrigin's Function

Rastrigin's function is separable and is defined by

$$F_{rastrigin}(\mathbf{x}) = \sum_{j=1}^{n} \left(x_j^2 - 10\cos(2\pi x_j) + 10 \right) \tag{A.7}$$

The nonseparable basic function, Rotated Rastrigin's function, is obtained from Rastrigin's function by multiplying with an orthogonal matrix for co-ordinate rotation.

$$F_{rot\_rastrigin}(\mathbf{x}) = F_{rastrigin}(\mathbf{Mx}) \tag{A.8}$$

where $\mathbf{M}$ is a $n \times n$ orthogonal matrix.

A.2.6 Rotated Ackley's Function

Ackley's function is separable and is defined by

$$F_{ackley}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}\right) - \exp\left(\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)\right) + 20 + exp \quad \text{(A.9)}$$

Ackley's function is multiplied with an orthogonal matrix M to form the nonseparable basic function Rotated Ackley's function.

$$F_{rot\_ackley}(\mathbf{x}) = F_{ackley}(\mathbf{Mx}) \quad \text{(A.10)}$$

where $\mathbf{M}$ is a $n \times n$ orthogonal matrix.

A.3 Benchmark Functions

The definitions of all the benchmark functions are provided in subsections A.3.1 to A.3.5. The function definitions are grouped according to separability. As before, n denotes the problem dimensionality, m denotes the group size and $\mathbf{x}$ denotes an input variable or candidate solution. The global optimum of a function is denoted by $\mathbf{o}$ and $\mathbf{z}$ denotes the shifted candidate solution $\mathbf{z} = \mathbf{x} - \mathbf{o}$. P denotes a random permutation of $\{1, 2, ..., n-1, n\}$ as described previously.

A.3.1 Separable Functions

The benchmark suite contains three separable functions, which are defined in table A.1. The corresponding expressions for the relevant basic functions can be found in section A.2. As mentioned, $\mathbf{z} = \mathbf{x} - \mathbf{o}$ where $\mathbf{o}$ denotes the function's shifted global optimum.

A.3.2 Single-Group m-Nonseparable Functions

A partially separable benchmark function is called single-group m-nonseparable if it contains a number of dependent variables with all the rest of the variables independent.

Table A.1: Separable functions

Function	Name	Expression
F_1	Shifted Elliptic Function	$F_{elliptic}(\mathbf{z})$
F_2	Shifted Rastrigin's Function	$F_{rastrigin}(\mathbf{z})$
F_3	Shifted Ackley's Function	$F_{ackley}(\mathbf{z})$

These functions fall into the first class of partially separable problems. The general form of these problems is given in Equation (A.11):

$$F(\mathbf{x}) = F_\gamma\big(\mathbf{z}(P_1 : P_m)\big) \times 10^6 + F_\alpha\big(\mathbf{z}(P_{m+1} : P_n)\big) \tag{A.11}$$

where F_γ is a nonseparable basic function, F_α is a separable basic function and $\mathbf{z}$ is obtained from $\mathbf{x}$ and $\mathbf{o}$ as described previously. There are five single-group m-nonseparable functions in the benchmark suite, provided in table A.2 below by specifying F_γ and F_α.

Table A.2: Single-Group m-nonseparable Functions

Function	F_γ	F_α
F_4	$F_{rot\_elliptic}$	$F_{elliptic}$
F_5	$F_{rot\_rastrigin}$	$F_{rastrigin}$
F_6	F_{rot_ackley}	F_{ackley}
F_7	$F_{schwefel}$	F_{sphere}
F_8	$F_{rosenbrock}$	F_{sphere}

A.3.3 $\frac{n}{2m}$-Group and m-Nonseparable Functions

A partially separable benchmark function is called $\frac{n}{2m}$-group and m-nonseparable functions if it consists of $\frac{n}{2m} + 1$ independent components, where the first $\frac{n}{2m}$ components are m-nonseparable and the last component is separable. These functions fall into the third class of partially separable problems. The general form of these problems is given in Equation (A.12):

$$F(\mathbf{x}) = \sum_{k=1}^{\frac{n}{2m}} F_\gamma\big(\mathbf{z}(P_{(k-1)m} : P_{km})\big) + F_\alpha\big(\mathbf{z}(P_{\frac{n}{2}+1} : P_n)\big) \tag{A.12}$$

where F_γ is a nonseparable basic function and F_α is a separable basic function. There are five $\frac{n}{2m}$-group and m-nonseparable functions in the benchmark suite, provided in table A.3 below by specifying F_γ.

Table A.3: $\frac{n}{2m}$-Group and m-nonseparable functions

Function	F_γ	F_α
F_9	$F_{rot\_elliptic}$	$F_{elliptic}$
F_{10}	$F_{rot\_rastrigin}$	$F_{rastrigin}$
F_{11}	F_{rot_ackley}	F_{ackley}
F_{12}	$F_{schwefel}$	F_{sphere}
F_{13}	$F_{rosenbrock}$	F_{sphere}

A.3.4 $\frac{n}{m}$-Group and m-Nonseparable Functions

A partially separable benchmark function is called $\frac{n}{m}$-group and m-nonseparable functions if it consists of $\frac{n}{m}$ independent subcomponents, all nonseparable. These functions fall into the second class of partially separable problems. The general form of these problems is given in Equation (A.13):

$$\sum_{k=1}^{\frac{n}{m}} F_\gamma\big(\mathbf{z}(P_{(k-1)m} : P_{km})\big) \tag{A.13}$$

where F_γ is a nonseparable basic function. There are five $\frac{n}{m}$-group and m-nonseparable functions in the benchmark suite, provided in table A.4 below by specifying F_γ.

A.3.5 Nonseparable Functions

The benchmark suite contains two nonseparable functions, which are defined in table A.5. The corresponding expressions for the relevant basic functions can be found in

Table A.4: $\frac{n}{m}$-Group and m-nonseparable functions

Function	F_γ
F_{14}	$F_{rot\_elliptic}$
F_{15}	$F_{rot\_rastrigin}$
F_{16}	F_{rot_ackley}
F_{17}	$F_{schwefel}$
F_{18}	$F_{rosenbrock}$

section A.2. As mentioned, $\mathbf{z} = \mathbf{x} - \mathbf{o}$ where $\mathbf{o}$ denotes the function's shifted global optimum.

Table A.5: Nonseparable functions

Function	Name	Expression
$F19$	Shifted Schwefel's Problem 1.2	$F_{schwefel}(\mathbf{z})$
F_{20}	Shifted Rosenbrock's Function	$F_{rosenbrock}(\mathbf{z})$

A.4 Benchmark Function Summary

The benchmark suite consists of 20 minimization problems, of varying degrees of separability. The optimal objective function value is zero for all the functions.

Table A.6 summarizes each benchmark function's degree of separability, modality and domain. The class of partially separable functions are specified. The modality column denotes unimodal functions with "U" and multimodal functions with "M". The optimal objective function value for all of the benchmark functions is 0.

Table A.6: Benchmark functions

Function	Separability	Modality	Domain
$F1$	separable	U	$[-100, 100]^n$
$F2$	separable	M	$[-5, 5]^n$
$F3$	separable	M	$[-32, 32]^n$
$F4$	partial class 1	U	$[-100, 100]^n$
$F5$	partial class 1	M	$[-5, 5]^n$
$F6$	partial class 1	M	$[-32, 32]^n$
$F7$	partial class 1	U	$[-100, 100]^n$
$F8$	partial class 1	M	$[-100, 100]^n$
$F9$	partial class 3	U	$[-100, 100]^n$
$F10$	partial class 3	M	$[-5, 5]^n$
$F11$	partial class 3	M	$[-32, 32]^n$
$F12$	partial class 3	U	$[-100, 100]^n$
$F13$	partial class 3	M	$[-100, 100]^n$
$F14$	partial class 2	U	$[-100, 100]^n$
$F15$	partial class 2	M	$[-5, 5]^n$
$F16$	partial class 2	M	$[-32, 32]^n$
$F17$	partial class 2	U	$[-100, 100]^n$
$F18$	partial class 2	M	$[-100, 100]^n$
$F19$	fully nonseparable	U	$[-100, 100]^n$
$F20$	fully nonseparable	M	$[-100, 100]^n$

Appendix B

Acronyms

This appendix lists all the acronyms that were used throughout the thesis. Acronyms are typeset in bold, with the corresponding meaning alongside. The list of acronyms is ordered alphabetically.

AIC average initial component.

CVT centroidal Voronoi tesselations.

gBest global best.

lBest local best.

LSO large scale optimization.

MGS modified Gram-Schmidt.

MIC maximum initial component-wise.

NSM non-linear simplex method.

PSO Particle Swarm Optimization.

RUS restricted uniform random.

Appendix C

Symbols

This appendix lists the important symbols used throughout the thesis, as well as their corresponding meanings. The symbols are divided according to the chapter in which the symbols were first introduced. Each chapter's list of symbols is ordered alphabetically.

C.1 Chapter 2: Background

c_1	Cognitive acceleration coefficient.
c_2	Social acceleration coefficient.
i	Index of particle within the swarm.
j	Index for problem dimension.
L	Lower bound of the search space when all dimensions have the same boundary.
L_j	The search space's lower bound in the j-th dimension.
n	The number of problem dimensions.
n_s	The size of the swarm.
$\mathbb{R}^n$	The n-dimensional real number space.
$r_{1,j}$	First uniform random stochastic scalar in j-th dimension.
$r_{2,j}$	Second uniform random stochastic scalar in j-th dimension.
t	The current iteration of the PSO algorithm.
t_{max}	The maximum number of iterations.

U	Upper bound of the search space when all dimensions have the same boundary.
U_j	The search space's upper bound in the j-th dimension.
$\mathbf{v}_i^t$	The velocity of the i-th particle at iteration t.
$v_{max,j}$	Maximum velocity in dimension j.
$v_{min,j}$	Minimum velocity in dimension j.
w	Inertia weight.
x_i^t	The position of the i-th particle at iteration t.
y_i^t	The i-th particle's personal best position at iteration t.
$\hat{y}_i^t$	The i-th particle's neighbourhood best position at iteration t.

C.2 Chapter 3: Symptoms of High Dimensional Problems

$\mathcal{D}$	Swarm diversity.
m	Degree of separability parameter (see Appendix A).
$\mathcal{V}$	Average particle velocity.
$\hat{\mathbf{x}}$	Swarm centre, the arithmetic mean of all particle positions.

C.3 Chapter 4: Velocity Clamping

δ	Fraction of search space to which velocity is clamped.
δ^*	Optimal clamping fraction for a given algorithm configuration.
F	Number of functions in benchmark suite.
f	A benchmark function.
(g, h)	A pair of algorithm configurations.
J	Number of algorithm configurations.
M_g	Total score of configuration g. [Eq. (4.8), pg. 42]
$R_{h,f}$	Median of the best score achieved by configuration g on function f.
$s_{g,h,f}$	Win of configuration g over h on function f. [Eq. (4.6), pg. 41]

$z_{g,h}$ Score of configuration g's wins over h. [Eq. (4.7), pg. 41]

C.4 Chapter 5: Variance of Particle Positions

γ Fraction to which the variance is restricted.

σ_j Standard deviation of particle position in the j-th dimension.

σ^2 Variance of a particle's position.

c Social and cognitive acceleration coefficients ($c_1 = c_2 = c$).

F Base frequency of particle positions.

V_c Coefficient of variance. [Eq. (5.6), pg. 57]

C.5 Chapter 6: Grouping Stochastic Scalars

$\mathcal{B}$ A set of vectors that form a basis for a specified vector space.

$\mathcal{G}$ A grouping or partition of $\mathcal{P}$.

G_k The k-th group in a grouping $\mathcal{G}$.

g The number of groups specified by a grouping strategy (may depend on t).

$\mathcal{I}$ The set containing the swarm's initial positions, personal best positions and velocities.

$\mathcal{P}$ A random permutation of $\{1, 2, ..., n\}$.

$\mathcal{S}$ The search space.

S_I A particle's step size within the initial subspace.

S_O A particle's step size outside the initial subspace.

C.6 Chapter 7: Swarm Initialization

α The reflection coefficient for NSM.

β The contraction coefficient for NSM.

γ The expansion coefficient for NSM.

τ	Fraction of search space within which swarm is initialized using CVT.
$\mathbf{a}_i$	Arithmetic average of the sample points assigned to generator i.
$\mathbf{b}_i$	An orthogonalized unit vector generated from the seed set.
$\mathbf{C}_i$	The i-th centroid for a given CVT iteration.
$\mathbf{c}$	Centre of the search space.
$\mathcal{D}(J, s)$	The discrepancy of s-many points w.r.t sub-interval J. [Eq. (7.2), pg. 129]
$\mathcal{D}^*(s)$	The worst-case discrepancy of a set of points s-many points. [Eq. (7.3), pg. 129]
$\mathbf{d}$	Direction vector, a random linear combination of orthogonal unit vectors.
$\mathbf{g}$	Generator, initial centroid position.
$\mathbb{I}^n$	The half open, n-dimensional unit cube.
L	Line passing through centre of search space with direction $\mathbf{d}$.
PD	Average distance between centroid pairs. [Eq. (7.10), pg. 144]
PD_C	Average component-wise distance between centroid pairs. [Eq. (7.11), pg. 145]
p_j	The j-th vertex of a simplex.
Q	The set of sample points generated for a given CVT iteration.
q	The number of sample points generated for a given CVT iteration.
s	The number of required initial positions.
t	Scalar that determines the generated initial point's position on the line L.
t_{max}	Maximum bound for the scalar t.
t_{min}	Minimum bound for the scalar t.
u	The number of vectors in the seed set for the subspace-based method.
$\mathbf{v}_i$	A vector in the seed set.

Appendix D

Derived Publications

The following publications were derived from this dissertation.

- E.T. van Zyl and A.P. Engelbrecht. A Subspace-Based Method for PSO Initialization. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 226–233, Dec 2015.
- E.T. van Zyl and A.P. Engelbrecht. Group-based Stochastic Scaling for PSO Velocities. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1862–1868, Jul 2016.
- E.T. Oldewage, A.P. Engelbrecht and C. Cleghorn. The Merits of Velocity Clamping Particle Swarm Optimisation in High Dimensional Spaces In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, Dec 2017.