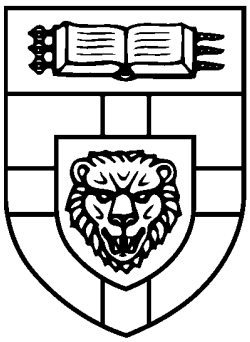
Simple and Adaptive Particle Swarms

Daniel Bratton



A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London

Department of Computing

Goldsmiths College

June 20, 2011

Declaration

I, Daniel Bratton, declare that the work presented in this thesis is entirely my own.

Signature _____

Date _____

Abstract

The substantial advances that have been made to both the theoretical and practical aspects of particle swarm optimization over the past 10 years have taken it far beyond its original intent as a biological swarm simulation. This thesis details and explains these advances in the context of what has been achieved to this point, as well as what has yet to be understood or solidified within the research community. Taking into account the state of the modern field, a standardized PSO algorithm is defined for benchmarking and comparative purposes both within the work, and for the community as a whole.

This standard is refined and simplified over several iterations into a form that does away with potentially undesirable properties of the standard algorithm while retaining equivalent or superior performance on the common set of benchmarks. This refinement, referred to as a discrete recombinant swarm (PSO-DRS) requires only a single user-defined parameter in the positional update equation, and uses minimal additive stochasticity, rather than the multiplicative stochasticity inherent in the standard PSO. After a mathematical analysis of the PSO-DRS algorithm, an adaptive framework is developed and rigorously tested, demonstrating the effects of the tunable particle- and swarm-level parameters. This adaptability shows practical benefit by broadening the range of problems which the PSO-DRS algorithm is well-suited to optimize.

Acknowledgements

My greatest appreciation and thanks go to my supervisor, Tim Blackwell, for his enduring patience and his enthusiasm for my work. Even at the most difficult times his optimism and encouragement reassured me that this thesis would – eventually – be complete. Further thanks go to others who directly helped me along the way: Jim Kennedy, Riccardo Poli, Nick Holden, Ollie Bown, Christophe Rhodes, Geraint Wiggins, and Tony Poppleton, and to the many more who provided years of discussion, encouragement, and camaraderie. Neither could this work have been completed without the unfailing support of Mariamne Rose. Gratitude can hardly cover what I owe, but she has all of mine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Improvement is a constant goal in many endeavors. Betterment of the self, of ideas, of tools, of nearly every feature of activity and existence drives both the natural biological process and most personal or group endeavors. Technological advances, business processes, education, and evolution itself are all based on the concept of improving knowledge and methods.

If improvement is the search for a "better" state in a given domain, *optimization* is the search for the "best" state. Global optimization (GO) is an attractive subject of study due to its applicability within both academic and professional spheres, and has over several decades been expanded to include a wide variety of techniques that can often bear little resemblance to one another. New approaches are constantly being conceived, introduced, and applied to find better or optimal solutions to problems both old and new.

This thesis will examine one such technique, Particle Swarm Optimization (PSO) from several distinct angles. It will bring parts of the algorithm that are relics of the past up to date, expand this modern definition of the algorithm to a simplified, improved form, and take steps toward the future of the field with a proposal for a new framework of adaptation using this improved form.

1.1 Motivation

Particle Swarm Optimization, first introduced in 1995, is a subset of the broader global optimization community[1, 2]. Its relative ease of implementation and excellent performance on optimizing difficult nonlinear problems has led to quick adoption in the associated academic communities. This research has taken the form of variations to the original algorithm of widely varying techniques and ability, but apart from several notable exceptions, there has been little active research focused on taking that original formulation and progressing it to an easily understood, more general form.

In spite of the mentioned ease of implementation, the PSO update equations are deceptively simple. Their straightforward form conceals complex second-order behaviour, with memory, multiple points of attraction, and potentially explosive multiplicative stochasticity. Even the simplest formulation contains at least two separate variables that must be balanced to produce stable behaviour and acceptable performance. The subtle effect that these various features can have on particle behaviour have limited analysis and restricted development to trial-and-error methods of testing variations to discover effective adjustments.

Simplified forms of PSO that remove unwanted and unnecessary behaviour while retaining the appealing level of performance, and vitally, the ease of comprehension and implementation, help to advance a better understanding of how the algorithm works, and why it behaves in its specific ways. The more complex any algorithm becomes, the more difficult and time-consuming it becomes to learn its use and application, limiting its appeal to new researchers and limiting new avenues of development. By shaping PSO into something that is demonstrably effective and can be picked up quickly, understood thoroughly, and expanded upon, the field can move forward to a focus on new concepts and practical uses.

One of these concepts that has seen only limited research is development of an adaptive form of the algorithm. Despite years of speculation that an adaptive particle swarm represents the future of the algorithm[3, 4], efforts to this point have been extremely complex[5], directed specifically to narrow problem types[6, 7, 8], or so extensively altered as to be unrecognizable as being derived from PSO[9]. Without a clear idea of how changing a parameter affects the behaviour of the swarm, all that can be done is continual random adjustment until an effective setting or combination of settings is found.

To move beyond these works, a general framework needs to be defined that allows for clearly-defined behaviour to be encoded into the swarm by means of adaptive rules to adjust these parameters. These rules need to be easily swapped in and out to demonstrate the effects on performance on a given problem. Being able to apply diverse adaptive rules to the same basic algorithm also leads to a better understanding of its functioning by observation of the effects on behaviour. By creating a simple and adaptive optimizer that is firmly tied to the principles of PSO, we can advance the algorithm in previously neglected directions in a way that improves understanding and functionality throughout the field.

1.2 Contributions

In this thesis PSO will be examined in terms of its relationship with established algorithms, including both swarm intelligence, and the larger field of evolutionary algorithms with which it is tentatively associated. Substantial advances have been made to both the theoretical and practical aspects of PSO since its inception, taking it far beyond its original intent as a simulator of biological swarms[10, 11, 12]. This progress will be detailed and explained in the context of what has been achieved to this point.

In spite of such progress, little has been done to establish a set standard version of the PSO algorithm that could be adopted and used by the entire community. Many variations exist, each with advantages and disadvantages that are often not obvious without in-depth research. Full understanding of the effects of the parameters of each individual variant is necessary to know how adjustments will affect behaviour of the particular algorithm, so the establishment of a fixed baseline for comparison is more feasible than incorporating every known version of PSO into such a standard. The establishing work for this standard is discussed and expanded upon, and used in later chapters in its intended role as a base of comparison for new techniques.

While this definition of a standard form of PSO is necessary to firmly establish its place in the library of GO algorithms, it should by no means be considered the final word on the subject. New techniques and variations are under constant development in nearly all fields, practical and theoretical; optimization

algorithms are no different. Such a development is undertaken here as a step toward the overall effort to develop a simple, adaptive form of the PSO algorithm.

This new adjustment to the standard algorithm constitutes a substantial reduction to the complexity of PSO, allowing for better understanding and an explanatory analysis of the behaviour of this class of algorithm. Importantly, performance is not impaired by these simplifications, remaining at least equal in nearly all cases and significantly improved in several areas. It is demonstrated that several components of the standard PSO are inconsequential to its operation – their removal in the simplified techniques is shown to have little effect on performance for the types of problems focused on in this work.

Finally, the future evolution of the PSO algorithm is discussed, and steps are taken down the path of an adaptive form. While the standard form of PSO, as well as the new techniques derived here, are shown to do well across a range of benchmark problems under clearly-defined general settings, each has several parameters that can be tuned to adjust the behaviour of the swarm and its associated performance. Creating a framework for adapting these parameters to the problem at hand permits relevance to a much wider array of possible optimization problems by removing the need for a well-defined, specially configured version of the algorithm to be set up for each problem. After a description of the methods and means of adaptation as applied to the suitability of PSO, it is shown to be an encouraging step for future development of the algorithm.

1.3 Objectives

The overall aim of this work is to present the concepts and implementations involved in simplifying the established PSO algorithm, allowing for the introduction of an adaptive aspect that is broadly applicable and unreliant on alterations to the core algorithm. These objectives can be detailed as the following:

1. to establish a standard form of the PSO algorithm that can be used in this work and others as a guide and a baseline for future developments,

2. to simplify this algorithm to allow for easier analysis while obtaining equivalent or better performance,

3. to perform a thorough mathematical analysis of the obtained simplified form in order to explain their behaviour under normal circumstances, and

4. to develop an adaptive system that clearly defines points and methods of adaptation allowing for broader application of the simplified algorithm.

1.4 Thesis Outline

Chapter 2 starts by providing a review of the field of optimization, including an explanation of the relevant terms and concepts. The sub-discipline of *evolutionary optimization* is examined in more specific detail, with the major techniques divided out and discussed in terms of their features and the methods used in their processes. Although PSO is not, strictly speaking, an evolutionary algorithm, these vari-

ous techniques are some of its closest neighbors, operating on the same types of problems in the same iterative manner.

Particle Swarm Optimization is briefly explained within the context of *swarm intelligence*, alongside other similar techniques. Its use in various forms over several different problem types is mentioned, followed by a review of the major variations to the algorithm that have been developed. Finally, a short summary of the applications of the algorithm to a variety of problem types is given.

Chapter 3 offers a thorough explanation of the PSO algorithm as originally proposed. The major techniques and modifications introduced over the next 10 years of development are itemized and discussed before being combined into a single standard form of the algorithm. This newly-defined form is rigorously tested and clearly explained, providing a solid, basic formulation of PSO for use both within this work and for future external use. As this baseline definition comprises a collection of previously proposed modifications to the original algorithm its status as an original contribution lies in the representation of an effort to bring the community together around a single, well-defined standard form from which all future research can be derived.

Chapter 4 builds on this standard algorithm by taking a reformulation and progressively altering and removing components until a much-simplified update equation is attained. This new equation, and the intermediate forms between it and the standard PSO are examined in terms of similarities and differences, and performance trials are used to demonstrate the equivalent-to-superior functioning of the simplified forms. This is used to argue that while the best representation of the technique remains the standard algorithm for reasons of the current level of adoption, that formulation very likely includes non-integral and possibly detrimental behaviours that can be removed without loss. The resulting form, which is less complex and hence more easily analyzed, is the focus of the following research.

Chapter 5 presents a mathematical analysis of the simplified algorithm using techniques which allow for description of both its behaviour and its convergence properties. These techniques are applied to the general form of the algorithm, covering all of the intermediate steps between it and the standard PSO described in chapter 3.

Chapter 6 develops a framework of adaptation that can be applied to the simplified PSO algorithm. The real-time adaptations to the main parameters describing the swarm improve its competitiveness on a broader problem benchmark over the original fixed-parameter version. More importantly, this framework can be used to develop new rules and adaptations to the algorithm in order to both broaden its applicability to new types of problems, and to improve its performance over current benchmarks.

Finally, chapter 7 concludes the thesis with a summary of findings. Potential future research that can be undertaken from the concluding point of this work is discussed.

The appendices present full results tables for the various empirical trials of the previous chapters and a list of publications that came about as part of the research detailed in the course of this degree.

Chapter 2

Literature Review

Despite the relatively recent inception of PSO when compared to established GO algorithms, it has been a popular topic of research within the community. Much of this research has been focused upon practical applications of the technique, resulting in a number of papers and articles demonstrating effectiveness on a variety of problems. These contributions have done much to spread knowledge in the academic and professional worlds, but little to advance understanding of the theory and mechanics behind the PSO process.

Conversely, few publications have been put forward that detail the inner workings of the algorithm and the causes behind its ability to optimize. Those contributions to PSO theory, though fewer in number, have played a larger role in its growing acceptance within the global optimization community, and more specifically that of heuristic evolutionary optimization.

This chapter provides a background for the development of particle swarms as optimizers through a review of the characteristics of optimization. The general field of global optimization is briefly examined, followed by a closer look at heuristic and evolutionary optimization. These broader super-fields contain the algorithms which are most often associated with PSO, most specifically evolutionary/genetic algorithms and differential evolution. The origins of PSO are then assessed, followed by descriptions of the most important of the numerous variations to the original algorithm.

2.1 Optimization

A complete definition of optimization is perhaps unattainable in anything but a complete review of the subject, but for the purposes of this study it can be summarized as the process by which a given measure of optimality is satisfied through an analytical search for an appropriate solution. Both the search and the solution are often constrained by specific requirements of the measure, e.g. limiting the searching area to some sort of feasible space, or prohibiting certain combinations of components. The discipline is divided into several distinct fields; the most relevant distinction for the research presented here is the division into *linear* and *non-linear* problems.

Linear optimization, widely referred to as *linear programming* is defined as the process of determining the maximum and/or minimum of a linear function, which has the common form in two variables x and y:

$$y = mx + b \tag{2.1}$$

where the constant *m* defines the gradient of the line and the constant *b* determines the offset of *x* from *y*. This is a simplified example, but clearly demonstrates that linear equations are those through which a first-order polynomial is equated to zero. Extremely capable algorithms have been developed for solving complex linear equations, most notably the simplex method [13] and a later algorithm utilizing the interior point method [14]. These techniques are common parlance in economics and engineering where linear equations arise on a regular basis.

Non-linear optimization, which is the area that most evolutionary techniques, including PSO, fall into, is concerned with determining the maximum and/or minimum of non-linear functions. These functions are composed of second or higher-order polynomials that often express complex behaviour, as well as functions that cannot be exactly expressed as finite-order polynomials. Non-linear equations are generally more difficult to solve than linear equations, requiring special techniques that are not always completely analytical.

2.1.1 Local Optimization

An optimization process is said to be *local* if only information from the immediate neighborhood in the search space of a candidate solution is used in revising the solution. Formally, a feasible point x^* is a local minimizer of f if:

$$f(x^*) <= f(x), \forall x \in B \tag{2.2}$$

where *B* is a subset of the entire search space *S*. Given that a search space can contain multiple unique subspaces, it follows that there can be multiple unique occurrences of these *local minima* in the broader space.

Local optimization as a whole can be broken down into the fields of gradient-based optimizers and direct search optimizers. Gradient search methods use gradient information from the first derivative of an objective function to update candidate solutions in a direction approaching the local minimum of the subspace. Examples of gradient-based methods include sequential quadratic programming [15], augmented Lagrangian methods [16], and the steepest descent algorithm [17]. Direct search methods do not use gradient information in their search procedure, relying entirely on objective function values for optimizing behaviour. Algorithms utilizing this approach include Nelder-Mead simplex method [18] (also known as the *Amoeba* algorithm), simulated annealing [19], and local versions of evolutionary algorithms such as genetic algorithms [20].

2.1.2 Global Optimization

Global optimization differs from local optimization in its overall goal. Where local optimization is concerned with finding the optimal solution in a subspace, global optimization is tasked with finding the optimal solution for the entire search space. This is usually a much more difficult task than local optimization due to the presence of multiple local optima within the search space. Figure 2.1 shows

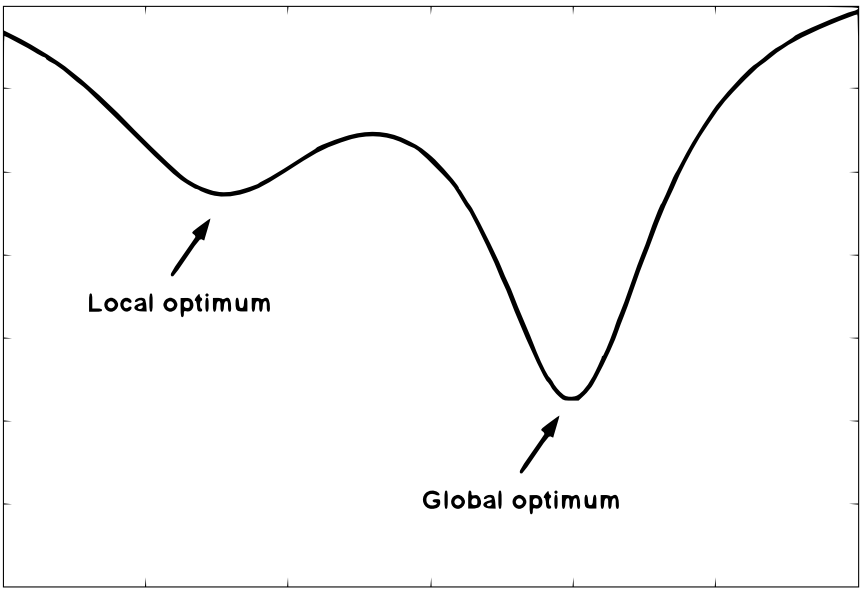


Figure 2.1: Local and global minima

local and global minima for a basic problem.

Determining which of these optima is in fact the global best solution is where gradient search methods fail on these problems. In following the slope of a function, a gradient optimizer is unable to "escape" from the basin of attraction of whichever minimum the initial candidate solution is placed in. In any problem containing more than one local minimum, gradient search methods are unable to assume or assure discovery of the global optimum.

Formally, a feasible point x^* is a global minimizer of f:

$$f(x^*) <= f(x), \forall x \in S \tag{2.3}$$

where *S* is the search space. In terms of the unconstrained problems that are the main focus of this work, $S = \mathbb{R}^n$ is common, where *n* is the dimension of *x*.

Global optimization techniques can be divided into four general categories based on their methods and reliability. According to Neumaier's definitive review of the field [21]:

- An *incomplete* method uses clever intuitive heuristics for searching but has no safeguards if the search gets stuck in a local minimum.
- An *asymptotically complete* method reaches a global minimum with certainty or at least with probability one if allowed to run indefinitely long, but has no means to know when a global minimizer has been found.
- A *complete* method reaches a global minimum with certainty, assuming exact computations and indefinitely long run time, and knows after a finite time that an approximate global minimizer has

been found (to within prescribed tolerances).

- A *rigorous* method reaches a global minimum with certainty and within given tolerances even in the presence of rounding errors, except in near-degenerate cases, where the tolerances may be exceeded.

Most evolutionary algorithms, including PSO, are classified as *incomplete* methods. While most are able to escape local minima at least some of the time, they have no means of guaranteeing an optimal global solution. They are, however, much faster than other methods, a result of only searching small sections of the search space and directing updates to their candidate solutions based on heuristic rules.

2.1.3 No Free Lunch Theorem

The "no free lunch" (NFL) theorem, put forward in a series on both search and optimization, theorized that for the set of all possible problem spaces, no individual algorithm could claim superior performance over any other algorithm[22][23]. In other words, both a highly specialized algorithm and a general purpose algorithm would show identical *average* performance when tested against all possible problems. This was demonstrated to be true on finite problem spaces, though it has yet to be proven for infinite spaces.

While on the surface these theorems appear to cast doubt on the effectiveness of developing new algorithms, it is notable that they only hold for the entire set of all possible problems. It was quickly claimed that NFL is not as restrictive as it first seems due to the fact that in practice, the operational space of most optimization algorithms is subject to constraints that are not taken into account by the theorem [24]. On subsets of these problems the theorems do not apply to the same extent. Because the area of interest for most optimization algorithms is a decidedly finite subset of the entire set of all problems, NFL does not erase hopes of improving on performance of standard algorithms.

This distinction led to a tightening of the definition of NFL to accommodate subsets of the space of all possible problems [25]. While this broadened the applicability of the NFL rules to smaller subsets, it is still the case that in many cases algorithms are specialized enough for a specific type of problem that significant improvements in performance over competing processes can be clearly demonstrated across the range of that subset.

2.2 Evolutionary Optimization

Evolutionary optimization is a process which can be applied both the local and global optimization paradigms. It is an application of the evolutionary computation (EC) technique, which was developed by several research teams simultaneously with slightly differing details, resulting in the fields of evolutionary algorithms, evolutionary programming, and evolution strategies. All of these techniques were developed with the same general goals in mind, and while originally each was considered a distinct process, in more recent times they have grown together and have been enveloped under the umbrella of evolutionary optimization.

Evolutionary computing (EC) is a field that has enjoyed a surge in growth and research in recent times. Its concepts are rooted in biology and computer science, which are combined to produce a disci-

pline that encompasses all methods for solving problems that are inspired by the traditional evolutionary method seen in nature. These processes are designed and modeled after biological evolution to produce the same developmental effects on a population over a period of time.

2.2.1 Evolutionary Algorithms

Evolutionary Algorithm (EA) is a very broad term that can potentially encompass all of the research subjects within the following sections. It is most often used to refer to the techniques developed for optimization and artificial intelligence, although it can also be used to describe any process which functions using biologically-influenced Darwinian rules[26].

2.2.1.1 Genetic Algorithms

Genetic Algorithms (GAs) compose a subset of EAs where the representation of a possible solution is encoded in a way inspired by a biological genome. They are a direct application of the concepts of biological evolution to computer models, and were originally designed to closely replicate the features observed in that process. The concept was originally proposed in 1957 by Alex Fraser[27], with several following publications expanding on the topic[28, 29]. Wider recognition and popularity came with the publication of the book *Adaptation in Natural and Artificial Systems* by John Holland[20].

A GA is designed to solve a specific problem, without any prior knowledge of an optimal solution. It operates by creating an initial number, or population, of *genomes*, each genome representing a single possible solution to the problem, usually randomly initialized. The encoding of the genome is an important factor, being the representation of the often abstract genome. This encoding can take a variety of forms, from simple bit strings or sets of floating-point numbers to more complex structures such as trees or key/value maps[30]. These genomes are then ranked according to their fitness - a measure of how well each individual genome meets the requirements defined by the problem and set by the designer for an optimal system. Those genomes with higher fitness ratings are considered to be superior, and are more likely to be selected to "reproduce" with other genomes.

This breeding is most often implemented through *crossover*, a process by which individual genes are selected from two "parent" genomes and placed into a newly created genome, considered to be the "child" of both of the parents. This results in a new genome that usually possesses characteristics of both of the parent genomes, without being an exact replica of either. *Mutation* can then be applied to the child genome on a probability basis where only a small portion of the offspring of a population will be mutated. How this is accomplished is totally dependent upon the form of the genome and the problem being solved.

In the widely-known GA formulation popularized by Holland[20], the processes of crossover and mutation are applied until there are an equal number of child genomes to parent genomes, at which point the population of parents is destroyed and replaced by the offspring. This entire process is known as a single generation. By selecting individual genomes with higher fitness values for reproduction, it is frequently the case that the new population of offspring after a generation will have a higher average fitness value than the population of parents. A GA can be set to run for as many generations as the designer sees necessary for the development of an optimal or near-optimal solution to the problem at

hand.

There has been a great deal of research and innovation in GAs since they were originally proposed in 1975, including extremely thorough examinations of the details and behaviour of the algorithm [31]. With a better understanding of the mechanics of GAs, new selection, crossover, and mutation methods have been introduced and refined, resulting in faster, more efficient and better performing techniques.

2.2.2 Evolutionary Programming

Evolutionary Programming (EP) was introduced as a method of simulating evolutionary processes with the aim to create learning behaviour for artificial intelligence systems [32]. Its original use was in the evolution of finite state machines for prediction.

In the originally defined form, EP is similar to genetic algorithms with the exception of a general lack of recombination. The main operator of the algorithm is mutation, applied randomly, using uniform probability distributions. Given the lack of recombination, each parent is set to produce a single offspring, mutated from the solution vector. Selection is then applied deterministically to eliminate the bottom half of the combined existing and newly-generated populations, ranked by fitness/performance.

Like all heuristic optimization algorithms, EP faces the problem of premature convergence to a suboptimal local minimizer, in this case caused by the tendency of strategy parameters used for adapting mutation sizes to reduce diversity early in the optimization process. Several techniques have been proposed to alleviate this problem, most notably the application of a dynamic lower bound to the parameter values to prevent total loss of diversity [33].

Modern implementations of EP have broadened the area to include a number of variations, bringing it more under the umbrella of EA representations. Real-valued variants of the algorithm have been proposed which operate by applying Gaussian mutations to solution vectors [34]. Improved performance was demonstrated through the replacement of the Gaussian mutation with a Cauchy-distributed mutation, with the fatter tails of the Cauchy distribution allowing more large mutations leading to escape from local minima [35]. This variation is known as *Fast Evolutionary Programming* (FEP).

2.2.3 Evolution Strategies

Evolution Strategies (ESs) are optimization techniques that when proposed were somewhat unique in the field of evolutionary optimization due to the inclusion of a method of adapting the strategy parameters during the problem optimization process[36][37]. It resembles GAs, and differs from EP, through its inclusion of recombination, in this case as a secondary update operator. Like EP, however, mutation still plays the primary role.

The two most common approaches to selection in ES are the (μ, λ) and $(\mu+\lambda)$ techniques, differing in the treatment of the selection process. (μ, λ) generates λ offspring from a population of μ parents. A population of size μ is then selected from the offspring to form the population for the next generation. $(\mu + \lambda)$ also generates λ offspring from μ parents, but the new population is selected from the combined parent and offspring population. This can be likened to a form of elitism in GAs, where the best of the original parent generation is retained after selection.

2.2.3.1 Differential Evolution

Differential Evolution (DE) is a global optimization method similar to GAs, but usually classed as an evolution strategy. It was proposed in the mid-1990s [38], around the same time as PSO, and has since accumulated research interest of approximately the same scale. Like PSO, it is a fast, relatively simple alternative to more complex evolutionary algorithms that shows very robust performance on nonlinear optimization problems.

DE operates by generating new position vectors by adding the weighted difference vector between two randomly-chosen population members to a third member, also chosen randomly. If the resulting vector is evaluated with a more optimal objective value than another randomly-chosen population member, the newly generated vector replaces that population member. If the objective value of the new vector is equally or less optimal than that of the chosen population member the existing member is retained and the new vector is discarded.

A comparison study of DE, PSO, and a few EAs demonstrated that DE showed superior performance to the PSO and the EAs on a large suite of benchmarks [39]. It should be noted however that the PSO algorithm used did not meet the stability criteria previously defined as necessary for stable performance (see Section 2.5.1.2), and was only tested using a global communication topology which had been previously demonstrated to be an inadequate choice for many optimization problems (see Section 2.5.2.1).

Limitations in the comparisons aside, DE has been shown to be an extremely capable optimizer, certainly on a par with PSO and any other heuristic optimization algorithm.

2.3 Swarm Intelligence

Swarm intelligence (SI) can be described as a system where a group of decentralized, relatively simple components work together to achieve a more complex overall goal. An algorithm or framework implementing the concept will normally adhere to five basic attributes defined in [40] and [41], namely:

1. *Proximity* - the swarm must be able to perform simple space and time computations,
2. *Quality* - the swarm should be able to respond to quality factors in the environment,
3. *Diverse response* - the swarm should not commit its activities along excessively narrow channels,
4. *Stability* - the swarm should not change its behaviour every time the environment changes, and
5. *Adaptability* - the swarm must be able to change its behaviour when the computational cost is not prohibitive.

The chief characteristic of the swarm is its self-organization – there are no external controls in place, and no rules that are applied to the conglomerated entity. The movement, behaviour, and optimizing potential all arise from the interactions of the individual components.

These individual components are described in various ways, depending on the application, algorithm, and the preferences of those who propose them. Although these components will usually have

different characteristics depending on the particular system being used, they will always be designed to interact with others of their type, giving rise to the emergence of the desired effect on the swarm level.

2.3.1 Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) is a SI algorithm that uses a population of agents interacting directly to search for optimal solutions to a given problem[42]. This search is carried out by means of a process where individual agents each test a single candidate hypotheses to determine whether it partially fits the definition of a "good" solution. Agents possessing such a hypothesis are then permitted to share their findings with the rest of the population, while those left without such a hypothesis become passive listeners. During the diffusion phase of the process, these listeners make direct communication with other randomly-chosen agents, and if one of these is in possession of a good partial solution, this is passed to the listener. Once a certain proportion of the swarm of agents is in possession of a single hypothesis, the process is halted and this hypothesis becomes the solution for the problem. As some hypotheses are more accurate than others, i.e. more of their partial evaluations will fit the good solution definition, the best found hypothesis at any point will propagate throughout the entire swarm, replaced by newer, more accurate hypotheses, until no better solution can be found.

SDS has been successfully used in a broad range of applications, including design of wireless networks[43], image processing and facial recognition[44], and robotics[45]. Its independence from solution encoding allows for use on almost any sort of separable space, eliminating the need for highly specific formulations designed for distinct types of problems.

SDS has been rigorously mathematically analyzed[46, 47, 48], describing its means of convergence to a global optimum and the behaviour associated with this process. In this respect it stands out from many evolutionary algorithms, a number of which are widely used but not necessarily well-understood.

2.3.2 Ant Colony Optimization

Ant colony optimization (ACO) was originally developed as a SI-based method for finding a path through a graph landscape that minimizes the distance traveled[49], a bottom-up approach to pathfinding algorithms that avoided the rigidity of established techniques. It was designed to mimic the behaviour of certain types of ants, which lay down pheromone trails between the nest and a food source. As ants are more likely to follow a previously-laid pheromone trail, and as the pheromones evaporate over time, shorter routes will be more frequently reinforced. This means that shorter, more optimal routes over time will be followed by more ants, in turn leading to a heavier pheromone trail, and eventually resulting in the entire ant population converging to a single route.

ACO uses simulated "ants", which interact indirectly in the same way as those in the natural world by laying down and strengthening links that direct the swarm toward improved solutions. This system was quickly adopted in the optimization community and explored and expanded upon in great depth.

While the original application of pathfinding provided an initial proof of the effectiveness of the concept, later applications have ranged from data mining[50] to job scheduling[51] to protein folding[52], amongst many others.

The exact functioning of the ACO algorithm is dependent on the particular variation and application in use. All variations follow the same method of a biased stochastic search of the problem landscape, using what are referred to in the literature as *pheromone models* to determine the biases. By changing the pheromone model, a researcher can alter the behaviour of the colony/swarm to fit the needs of the research topic being pursued.

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO), first proposed by Eberhart and Kennedy in 1995 [1][2], is a relatively new form of optimization algorithm that has become popular due in part to its speed and relative ease of implementation. Originally conceived as a simulator of bird flocking behaviour, it is most appropriately classed as a form of swarm intelligence[41], though given the similar objectives and concepts many researchers study it (as well as many other swarm intelligence algorithms) in the context of evolutionary algorithms despite its distinctly different origins.

Patterns of behaviour within swarms have been studied in a great deal of depth with many models developed, some of them representing quite accurate simulations of biological systems such as flocking and swarming [53][54]. The optimum-seeking behaviour of a flock seeking a food source was realized to be a form of optimization of a problem space, with the flock members representing candidate solutions and the food source representing the optimum of the search space. Biological groups of birds, bees, and the like demonstrate complex internal interaction patterns which allow the optimum of the physical search space to be found, a behaviour which can be translated to mathematical optimization of problems in real-valued spaces.

2.5 The Basic PSO Algorithm

In the implementations of PSO described and cited in this review, all of which derive from the seminal works of the field[1, 2], a particle moves through the search space using a combination of an attraction to the best solution that that particle individually has found, and an attraction to the best solution that any particle in its *neighborhood* has found. For these implementations, the neighborhood is defined for each individual particle as the subset of particles which it is able to communicate with. The very first PSO model used a Euclidian neighborhood for particle communication, measuring the actual distance between particles to determine which were “close enough” to be in communication[1]. This was done in imitation of the behaviour of bird flocks, taken from biological models where individual birds are only able to communicate with other individuals in the immediate vicinity.

While more biologically accurate (and nice to watch in graphical representations), the Euclidian neighborhood communication model was quickly abandoned in favor of less computationally intensive models when research focus was shifted from biological modeling to mathematical optimization[2]. Topological neighborhoods unrelated to the locality of the particle came into use, first what has come to be known as a global, or *gbest*, neighborhood model, where each particle is connected to and able to obtain information from every other particle in the swarm (see figure 2.2(a)), and the local *lbest* model, where each particle is able to communicate with only a limited number of neighbors. While strict usage of the term *lbest topology* allows for description of any swarm topology that is not global, it is often used interchangeably with the term *ring topology*. A *ring* topology exists when each particle is able to communicate only with its two neighbors when the swarm is arranged in an abstract "ring" layout (see figure 2.2(b)). This study will refer to such a topology as a *lbest ring* or simply *ring*. See Section 2.5.2.1 for more on swarm topologies.

By the original definitions[1, 2], an individual particle i is composed of three vectors: its position in the *D*-dimensional search space $\overrightarrow{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$, the best position that it has individually found $\overrightarrow{p}_i = (p_{i1}, p_{i2}, ..., p_{iD})$, and its velocity $\overrightarrow{v}_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Particles were originally initialized in a uniform random manner throughout the search space; velocity was also randomly initialized.

These particles then move throughout the search space using a set of update equations. The algorithm updates the entire swarm at each time step by updating the velocity and position in every dimension d of each particle i sequentially by the following rules:

$$v_{id} = v_{id} + c\epsilon_1 (p_{id} - x_{id}) + c\epsilon_2 (p_{gd} - x_{id}) \tag{2.4}$$

$$x_{id} = x_{id} + v_{id} \tag{2.5}$$

where in the original equations c is a constant with the value of 2.0, ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each individual dimension $d = 1$ to D, and $\overrightarrow{p}_g$ is the best position found by any neighbor of the particle.

The PSO algorithm

for each particle i in the swarm **do**
  initialize position $\overrightarrow{x}_i$
  initialize memory $\overrightarrow{p}_i$ and $\overrightarrow{p}_g$
for each time step t **do**
  for each particle i in the swarm **do**
    update position $\overrightarrow{x}_i^t$ using eqs 2.4 & 2.5
    calculate particle fitness $f(\overrightarrow{x}_i^t)$
    update particle and swarm memories $\overrightarrow{p}_i$ and $\overrightarrow{p}_g$

Examination of the velocity update equation reveals the mechanics of particle motion. The first term, v, is the velocity from the previous update process – this is similar to inertia in a physical setting,

and was originally used to simulate natural motion when particle swarms were intended as biological simulations. The second term, $c\epsilon_1 (p_{id} - x_{id})$ is the attractive force toward the best found position of the particle (*personal best*), while the third term, $c\epsilon_2 (p_{gd} - x_{id})$ is the attractive force toward the best found position of any of the neighbors of the particle (*neighborhood best*). While the constant c could be detached into two separate values c_1 and c_2 which could be tuned independently to weight the personal and social influence of two terms, the effect on particle velocity was not well understood at the time and poor choices led to swarm instability.

Particle velocities in the original PSO algorithm had to be clamped at a maximum value *vmax* where $v_d < |vmax|$ for every dimension d, effectively restricting its value to the range [-*vmax*, *vmax*]. Without this clamping in place the system was prone to entering a state of explosion, wherein the random nature of the ϵ_1 and ϵ_2 values eventually led to successive combinations that caused particle velocities to increase rapidly, approaching infinity and taking particles far outside the relevant search space.

2.5.1 Controlling Instability

The *vmax* solution introduced several undesirable elements to the PSO process. The value chosen for *vmax* was highly dependent on the other properties of the system, especially the landscape being optimized, and there was no established guideline for determining an acceptable value short of trial and error. Particle behaviour was altered as well; rather than allowing particles to converge naturally to an attractor, their movements were abruptly halted when velocity reached *vmax*, in turn disrupting subsequent velocity updates. The imposition of this artificial limit introduced an external factor into the algorithm, and complicated investigation and analysis of the behaviour observed in the optimization process.

2.5.1.1 Inertia Weight

To circumvent these problems, a new parameter was introduced in an effort to strike a better balance between global exploration and local exploitation while preventing the explosion in velocity seen in the unclamped PSO[55]. This new *inertia weight* parameter w was designed to replace *vmax* by adjusting the influence of the previous velocity on the optimization process. The parameter was added to the original velocity update equation, resulting in:

$$v_{id} = wv_{id} + c\epsilon_1 (p_{id} - x_{id}) + c\epsilon_2 (p_{gd} - x_{id}) \tag{2.6}$$

By adjusting the value of w up or down, the previous velocity term is weighted and the swarm has a respectively lesser or greater tendency to eventually constrict down to the area containing the best fitness and explore that area in detail. It was later demonstrated that w can be defined as a dynamic value over the optimization process, starting with a value greater than 1.0 to encourage early exploration of the search space, and decreasing eventually to a value less than 1.0 to focus the efforts of the swarm on a

specific locale containing the global optimum [56]. This control of the "inertia" of particles, similar to the role of friction or viscosity in a physical setting, removed the need for velocity limiting and showed improved performance over the original algorithm.

Several other studies have looked into the inertia weight parameter and explored strategies for its use. In one case a random value was applied to the previous velocity term rather than using either a fixed or dynamically decreasing value as originally proposed, with competitive results[57]. Elsewhere, an increasing value for w was applied, allowing the swarm to quickly find an optimum and then search around that area for improved regions[58].

2.5.1.2 Constriction Coefficients

Another method of solving the problem of explosion, referred to as *constriction*, was being explored simultaneously with the inertia weight method. Like inertia weight, it involved the addition of a single parameter to the velocity update equation and removed the need for velocity clamping. This method was introduced and disseminated throughout the research community informally for several years and was used (often uncited) in multiple publications[59] prior to publication. It was formally published in 2002, accompanied by a complete analysis of particle trajectories and swarm stability[10].

Several methods were proposed therein for introducing this means of swarm constriction and convergence. The simplest of these, which was used as a concluding aspect of the work, was referred to in the study as PSO Type $1''$, and introduced a new parameter χ to the velocity update equation. χ, the *constriction factor*, was derived analytically from the existing constants where:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{2.7}$$

where $\varphi = c_1 + c_2$[10].

It had been previously observed that if φ was defined within the range $[0...4]$, trajectories of deterministic particles with no random components demonstrated regular behaviour[3]. The analysis of particle trajectories included in the introduction of constriction showed that when $0 < \varphi < 4$, the swarm would slowly "spiral" toward and around the best found solution in the search space with no guarantee of convergence, while for $\varphi > 4$ convergence would be quick and guaranteed. While it is possible to weight the velocity update equation to favor the best position of the individual particle p_i or the best position of the entire swarm p_g by adjusting the values of c_1 and c_2, for the sake of simplicity most implementations of constricted particle swarms use equal values for both parameters. Using equation 2.7 with the constant $\varphi = 4.1$ to ensure convergence, the values $\chi \approx 0.72984$ and $c_1 = c_2 = 2.05$ are obtained. This constriction factor is applied to the entire velocity update equation:

$$v_{id} = \chi \left(v_{id} + c_1\epsilon_1 \left(p_{id} - x_{id}\right) + c_2\epsilon_2 \left(p_{gd} - x_{id}\right)\right) \tag{2.8}$$

Given that constriction and inertia weight are algebraically equivalent via the mappings $w \leftrightarrow \chi$ and $c_{1,2} \leftrightarrow \chi c_{1,2}$, the effects are identical. Swarms show convergent behaviour and are eventually limited to a small area of the feasible search space containing the best discovered solution. A comparative study of the two methods demonstrated that the PSO algorithm with constriction is in fact a special case of the algorithm with inertia weight in which the values for the parameters have been determined analytically[60]; algebraically the standard constriction terms of $\chi \approx 0.72984$ and $c_1 = c_2 = 2.05$ equate to an inertia weight system with $w \approx 0.72984$ and $c_1 = c_2 = 1.49618$. The parameter values noted above were preferred in most cases when using constriction for modern PSOs due to the proof of stability for those values. This proof of stability makes the constricted form of PSO possibly the more popular "standard" choice for the algorithm in subsequent publications, though inertia weight is still widely used with the equivalent parameters.

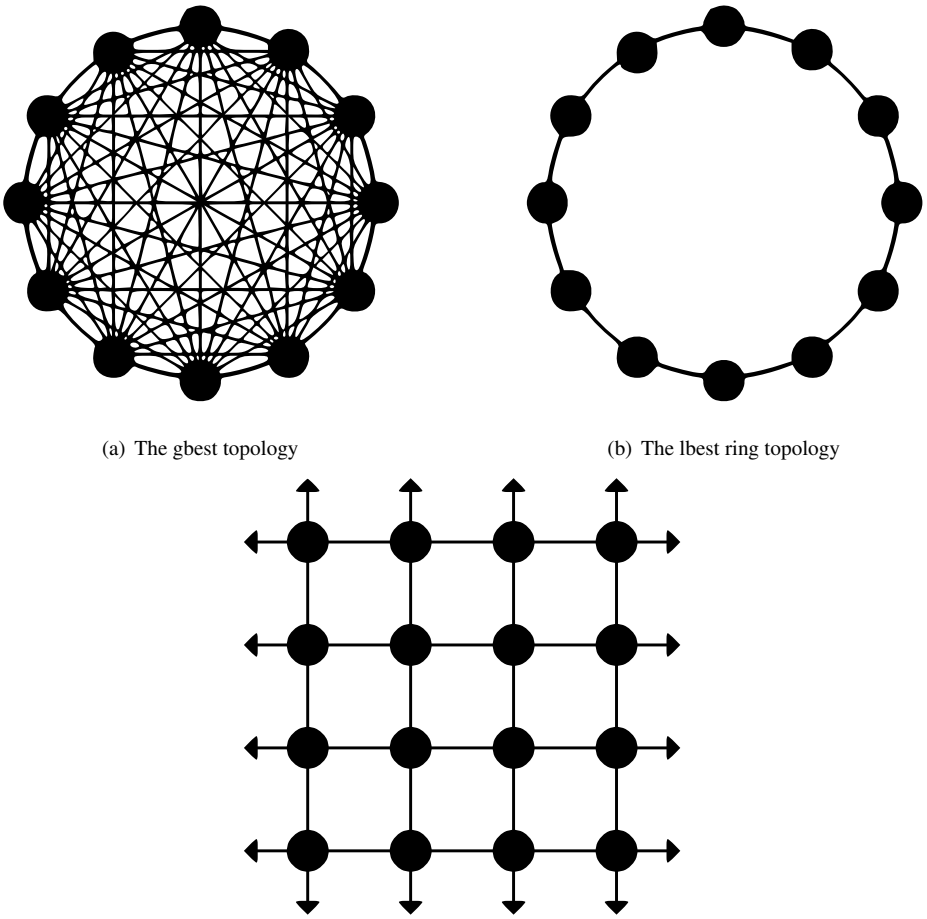
2.5.2 Communication Topologies

2.5.2.1 Static Topologies

In its original formulation as a simulation of biological swarms, communication within particle swarms was organized by way of Euclidian neighborhoods, i.e. each particle communicated only with its current nearest spatial neighbor(s) at any given point of the optimization process[1]. This was found even in that initial study to be extremely computationally expensive, especially in high dimensional spaces, and more abstract topologies were adopted in it and the following expansion where each particle was permanently connected to and able to communicate with other predetermined neighbors[1, 2].

The first of these was the *gbest*, or global topology (figure 2.2(a)). As the name suggests, this model allows communication to take place between all particles. This has the net effect of allowing every particle in the swarm to immediately access the best position found by the swarm as a whole for use as the neighborhood best term in its velocity update equation. While this is implausible for a biological swarm, it is far simpler for optimization purposes. The attraction toward the best found position in the entire swarm results in every particle being attracted toward that point, allowing for thorough exploitation of the associated optimum. This is beneficial in cases where that point is very likely to be the global optimum, such as unimodal problems – those containing a single, global optimum. Such behaviour is not always desirable however, especially in cases where many local optima exist that the swarm could be quickly drawn into.

The other topology used in the original proposal of PSO[1] was the *lbest*, or local topology (figure 2.2(b)). While global topologies are equivalent to fully-connected graphs with $K = \frac{n \times (n-1)}{2}$ edges where n is the number of vertices, local topologies are just one edge over being minimally connected, with n edges. This ring design allows information to be spread throughout the swarm, but it is done slowly as each particle is influenced only by two others. This is equivalent to a graph with $K = 2$ con-



(a) The gbest topology

(b) The lbest ring topology

(c) The von Neumann topology

Figure 2.2: Particle swarm topologies

nectivity. The advantage of the ring topology over global lies in its slower convergence rate. Rather than having every particle drawn simultaneously toward a single point of the search space, multiple points can be explored by subpopulations before the swarm is eventually drawn to the area with optimal fitness. Figure 2.3 shows a swarm consisting of twelve particles optimizing a multimodal problem, where each particle is attracted to one of three separate peaks, indicated by shading. This indicates that the swarm has split into multiple sub-swarms, simultaneously exploiting several areas of the problem landscape. Such behaviour results in a swarm that is less likely to become trapped within local optima on multimodal problems, making it a better global optimizer. At the same time, swarms with local topologies will usually optimize a peak considerably slower than those operating with global communication due to the slower information dissemination[1, 2, 61].

Based on the major differences in swarm behaviour between these two original topologies, further research was performed to look into the influences of the social dynamics of a swarm. Initial investigations into many different types of communication topologies showed that while performance varied

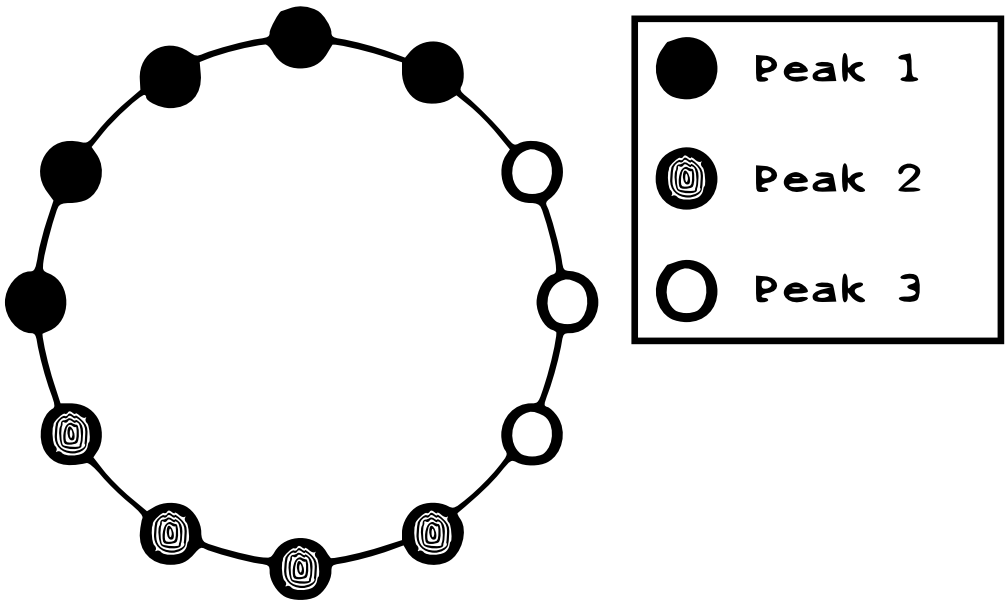


Figure 2.3: A ring-topology swarm split into sub-swarms across multiple peaks. Shading indicates which of three peaks the specific particle is currently attracted to via the cognitive term p_{id}.

greatly based on whichever was used, this was highly dependent upon the problem being optimized[61]. In an effect similar to that of the No Free Lunch theorem[22][23] (see section 2.1.3), no single topology could be demonstrated as optimal across all potential problems[61].

A general study of swarm communication topologies tested a great many models, most of which were randomly generated, with the intent of studying how the various properties of topologies influenced swarm behaviour using standard constricted update equations[62]. The properties were derived from work on the small worlds theories[63, 64, 65] and included two elements. The first was graph *K-connectivity*, or how many edges/connections emanated from each vertex/particle in the communication topology. The second was *clustering*, the average number of neighbors to a vertex/particle which were also neighbors to one another, over an entire swarm. The effect of self-inclusion of a particle into its own neighborhood was also examined, and was determined to be relatively unimportant next to overall topology design[62].

This examination, combined with a later, more in-depth study[12], revealed that swarm topology influences behaviour and performance to a great degree, often making the difference between success and failure on the tested optimization problems. Surprisingly, both lbest and gbest swarms showed poor performance when compared to swarms with moderate values for K-connectivity[12]. Overall, topologies with $K = 5$ performed best after 20,000 function evaluations, while those with $K = 10$ were those which were fastest to reach the optimum in trials where it was found[12]. Study of the clustering property showed that it had less effect than the connectivity of the topology, though a topology with

clearly defined clusters of particles was one of the better performers[12].

A major development to come from this work was what was referred to as the toroidal "vonNeumann" topology, one of those with $K = 4$ (see figure 2.2(c))[12]. This is a very straightforward non-random topology which showed excellent performance on the five tested problem landscapes. It constitutes a tradeoff between the respective low and high connectivity of lbest and gbest swarms, and a similar tradeoff between the slower exploratory searching and faster exploitative searching of the two extremes.

2.5.2.2 Dynamic and Adaptive Topologies

Given the demonstrated differences in behaviour among the various types of swarm communication topologies, a dynamic and/or adaptive topology would seem to be a viable option. As was shown with the von Neumann topology, a tradeoff between exploration and exploitation is beneficial to performance. This promotes the idea that sparsely-connected swarms may be most useful early in the optimization process for locating the area of the global optimum of a problem, while more densely-connected swarms would allow fast, accurate discovery of the single best point within that basin of attraction. Rather than using a fixed topology throughout, a topology could be adapted to take advantage of the current state of the system for optimal performance.

Basic research into this was performed where swarm connectivity was increased throughout the optimization process, though results were not entirely conclusive[66]. This method is dynamic, but lacks an adaptive nature – the topology is altered based entirely on a predetermined function of time, or more accurately the number of performed function evaluations. More in-depth study was carried out in an informal capacity and to date remains unreviewed and unpublished[67]. A study of the C-language code for this system shows that it uses an adaptive topology that is generated so that each particle is informed by three other particles, randomly chosen, i.e. $K = 3$. The best found fitness value for the entire swarm is checked after every iteration, and if it has failed to improve since the last check, the entire topology is destroyed and recreated from scratch using the same rules. Reported results have been promising, but to date remain unpublished.

2.6 Statistical Analyses of PSO

Until recently, mathematical analyses of particle swarm optimization have been rather limited in specific ways. This is for the most part due to the multiplicative stochasticity of the update equations, and the difficulty in quantifying the relationships and influences in swarms with numerous particles. In spite of these hurdles, there have been several illuminating studies into the theoretical aspects of swarms and particles that have revealed how a PSO algorithm actually functions.

While inroads were made into the study of the nature of swarm optimization fairly soon after its proposal by Ozcan and Mohan[68], this examination confined itself to a single particle operating deter-

ministically in a single dimension during stagnation with a single point of influence – the personal best and global best were set to be equivalent, i.e. $p_i == p_g$. Further, these early results only supported the original form of the algorithm, which had already been superseded in performance and adoption by the inertia weight formulation[55]. Despite these limitations, this first formalization of particle behaviour helped to define some of the concepts behind the algorithm, including particle trajectories. This study was later expanded and generalized to take into account multiple particles in a multi-dimensional search space[69].

Intense study into particle trajectories was occurring simultaneously with the previous work and being presented widely throughout the community by Clerc and Kennedy. Despite its delayed publication, this study was influential to Ozcan and Mohan's research as well as others in its approach to a statistical analysis of PSO[10]. This model included a proof of convergence for the algorithm given appropriate parameter settings alongside the new update formulation with constriction. It used assumptions similar to those of Ozcan with a single particle operating deterministically under stagnation.

Blackwell pursued a better understanding of the behaviour of the swarm as a whole via an investigation into the variation of its spatial properties over time[70, 71]. This study extended that of Clerc and Kennedy by modeling an entire swarm of constricted particles including interaction and inter-swarm influences. As in the Clerc/Kennedy work, stochasticity was not taken into account, however stagnation was not assumed by way of changeable personal best positions.

In his doctoral thesis, van den Bergh investigated a non-stochastic, stagnated analysis of particle trajectories, and demonstrated that the particle under stagnation is constantly attracted to a fixed point[72]. This fixed point was based entirely on the circumstances of the particle, and was not necessarily optimal in the search space, even for the immediately surrounding area.

Trelea analyzed the parameter space of a particle operating under the same common assumptions made in the previous Clerc/Kennedy study[73]. This demonstrated the influence of various parameter settings on the behaviour of particles and the resulting capability of the swarm to carry out an effective optimization.

Clerc returned to the analysis of PSO in an unpublished technical report on particle velocity under stagnation[74]. Unlike all previous work to this point, this study took stochasticity into account, which provided a model approach for later investigations.

Now taking stochasticity into account as per Clerc, Kadirkamanathan pursued an explanation of the stability of the single best particle in the swarm, i.e. the particle for which $p_i == p_g$[75]. Using a Lyapunov analysis, this work used an interdisciplinary approach to dynamical systems known as *control theory* to demonstrate that under appropriate selections for parameter settings, PSO can be guaranteed to converge, albeit not necessarily to an optimal point.

In 2007 Poli et al published an early investigation into using a generalized method to analyze a sim-

plified form of the standard PSO algorithm[76]. The simplified algorithm was not particularly effective in comparison to the full algorithm, but the methods used to derive its stability and sampling distribution were the first published appearance of a method of analysis that itself made no simplifying assumptions apart from stagnation.

Finally, later in 2007 Poli introduced a refined, generalized method of determining all of the characteristics of a swarm algorithm's sampling distribution while under stagnation[77, 78], and applied it to the full standard PSO algorithm. The only assumption made with this method is that of particle stagnation, meaning that it is able to be used for analysis of many types of stochastic update equations.

2.7 PSO Variants

2.7.1 Fully Informed Particle Swarms

The *Fully Informed Particle Swarm* (FIPS) constitutes a fairly major variation to the social interaction of particles from the system of more standard swarms [62, 81]. Rather than the traditional approach whereby a particle's position update was influenced by just two other positions in the search space, its personal best position and the best position found by any particle in its neighborhood, under FIPS the update takes into account the best found positions of every neighbor of the particle.

The new positional update equation for FIPS was in the form:

$$\varphi_k = U\left(0, \frac{\varphi_{max}}{|N|}\right), \forall k \in N$$

$$P_m = \frac{\sum_{k \in N} W(k)\varphi_k \times P_k}{\sum_{k \in N} W(k)\varphi_k} \tag{2.9}$$

where φ replaces the random numbers $\epsilon_{1,2}$ from the original PSO equations, N is the set of neighbors for the particle being updated and P_k is the best position found by the individual particle k. W is a function describing a relevant property of the particle, which was explored in the original investigation using a number of variations such as best fitness or distance between particles. This general form was later refined to a traditional combination of the velocity and positional updates:

$$v_{t+1} = \chi\left(v_t + \sum_{n=1}^{N} \frac{c\epsilon(p_{nbr(n)t} - x_t)}{N}\right) \tag{2.10}$$

$$x_{t+1} = x_t + v_{t+1}$$

where χ is a constricting factor, N is the number of particles in the neighborhood of the currently updating particle, and $nbr(n)$ is the particle's nth neighbor[87].

FIPS was demonstrated to return significantly better performance than the original PSO system when properly tuned[62, 81]. It was also noted that the swarm communication topology played a large

Variant	Type	Description	Reference
Bare-bones Gaussian	Static	Replaces update equations with a Gaussian probability distribution	[79]
Bare-bones Lévy	Static	Replaces update equations with a Lévy probability distribution	[80]
FIPS	Static	Replaces update equations with a method which informs particles using a combination of the best positions of all neighbors	[81]
Simpler PSO	Static	Removes multiplicative stochasticity from update equations	[76]
PSODR	Static	Replaces personal or neighborhood best position in update equations with a recombinant position informed by all neighbors	[82]
MOPSO	Multiobjective	Guides particles to unexplored space to find non-dominated solutions	[83]
DIW PSO	Dynamic	Randomizes inertia weight to prevent full convergence to a single moving peak	[57]
Charged Swarms	Dynamic	Introduces repulsive force between particles to prevent full convergence to a single moving peak	[84]
Multi-swarms	Dynamic Multiobjective	Uses multiple independent, interacting swarms using repulsion, exclusion, and anti-convergence methods to track multiple moving peaks	[85]
Tribes	Adaptive	Replaces update equations with a pivot method, uses social interaction of particles to dynamically regenerate topology	[11]
EPSO	Adaptive	Randomly perturbs algorithm parameters according to adaptive rules informed by selection-based process	[86]
APSO	Adaptive	Uses a fuzzy classification system for estimating the state of the swarm to inform adaptations	[5]

Table 2.1: Notable variants of the PSO algorithm

role in this new method, a theme that would become common in many variants of PSO.

2.7.2 Bare-Bones PSO

In an effort to better understand the mechanics underlying particle behaviour during the optimization process, PSO was reduced to a simpler form[79]. Examination of the distribution of positions around a fixed point in the search space over multiple iterations revealed a similarity to a Gaussian distribution. This reduced "bare-bones" particle swarm took advantage of this similarity by replacing the method of updating the positions of particles (previously done via equations 2.4 and 2.5) with a normal probability distribution with mean in each dimension of $\frac{p+g}{2}$ and a standard deviation of $|p - g|$.

As velocity is no longer required for the update, the entire update equation for this form of PSO is wholly positional:

$$x_t = G(\frac{p_t + g_t}{2}, |p_t - g_t|) \tag{2.11}$$

where G($mean$,std) is a sample from a Gaussian distribution.

This original Gaussian-based bare-bones PSO performed comparably to the more standard algorithm over five standard test functions, suggesting that the velocity vector could be unnecessary, an "arbitrary piece of baggage from the initial forms of PSO." Removing the velocity vector from the algorithm reduces the effect of particle trajectories and refocuses attention on the connections between particles and their influence on swarm behaviour.

> "The essence of the paradigm is not to be found in its flying trajectories, but in the influence that individuals have on one another, in the persistence of individual identities over time, in the stability of sociometric interaction patterns. The trajectories are well-described by the bell-curve of test points centered between the individual's previous best point and some other point derived from social experience."[79]

However, further investigation of this system revealed that while bare-bones PSO possessed many similarities to the standard algorithm, performance was degraded in some instances and behaviour was not identical[88]. Close examination of the supposed Gaussian-like distribution of particle positions revealed a much higher kurtosis than for a Gaussian, i.e. there were more extreme deviation from the mean influencing the variance of the velocity-based system than in a Gaussian distribution, where the variance is influenced by a larger number of moderate deviations. These extreme deviations were referred to as "bursts of outliers", and were conjectured to account for the slightly improved performance of the velocity-based PSO due to what was believed to be a higher swarm diversity[88].

This implication led to the development of a bare-bones system using a Lévy distribution[80]. The Lévy distribution is based on the concept of Lévy flights, a type of random movement in which the

increments are distributed with infinite variance [89]. This gives a distribution with outlying deviations similar to those observed in a PSO system, the frequency of which can be adjusted according to a parameter α. For values of $\alpha = 1.4$ it was demonstrated that the performance and behaviour of this "Lévy Swarm" was extremely close, if not identical, to that of a standard velocity-based PSO[80].

2.7.3 Adaptive PSOs

An adaptive PSO has been defined as a system with no user-defined parameters that adjusts its behaviour to an optimal form related to its current situation without the need for external input[90]. As of yet, few attempts have been made to develop an adaptive particle swarm algorithm despite the obvious appeal of being able to disregard the tedium and guesswork of parameter tuning and hybridization of the method for better performance on specific problems.

While this form of adaptation has been shown to be very effective on dynamic landscapes, few such methods exist for the more common static landscapes. A concept from evolution strategies was applied to PSO where the parameters were randomly perturbed by small amounts dependent on a selection-based process that was used to adapt the perturbations[86].

There have been a few efforts at introducing adaptive features into PSO using the standard constriction coefficient or inertia weight formulations[91][6][92]. Perhaps the most notable of these is *Tribes*, an optimization system influenced by the social interaction of particles within the algorithm, but with distinctive particle behaviour and complex adaptive rules which are used to develop a swarm intended to be well-suited to the landscape under optimization[11]. Tribes has demonstrated very good performance on a variety of benchmarks, though its major deviations from the norms of a canonical PSO system, e.g. foregoing the weighted neighborhood influence for a pivot method of positional updates, make its classification within the same field somewhat questionable.

More recently there have been efforts at creating an algorithm using the original particle swarm optimizer where the swarm bases the adaptation of three different update equation parameters on the state of a feedback variable[5]. This showed promising results, with the calculation of this "evolutionary factor" being used to determine the state of the optimization, which is then passed through one of four different strategies of adaptation, finally applying a sigmoid mapping to the previous velocity term to control its influence. While results were shown to be improved over several other forms of PSO[5], the level of complexity in the implementation limits the possibility for extension by any but the original authors.

The most obviously adaptable components of a PSO system are the number of particles in the swarm, the K-connectivity of the topology, and the various parameters in the velocity update equation. It has been shown that adaptation of these parameters can lead to impressive gains in performance[90], and it has been theorized that a fully adaptive particle swarm is the future of the field[4].

2.7.4 PSO on Other Problem Types

2.7.4.1 Constrained Optimization

While most research into PSO has focused on unconstrained optimization, which has few to no requirements on the values of the parameters, a few examinations of its capability on *constrained nonlinear optimization problems* (CNOP) have been published. CNOPs have been very well-researched topics of research in the general optimization community, with significant amounts of research devoted to algorithms designed for the purpose of solving them, including several evolutionary algorithms [93].

General CNOPs are defined as [94] the effort to optimize a nonlinear objective function subject to a set of equality and inequality constraints. Formally, the goal is to find the solution vector

$$x^*, x^* = (x_1, x_2, ..., x_n) \in \mathbb{R}^n \tag{2.12}$$

where for the objective function $f(x) \in \mathbb{F} \subseteq S$ where $\mathbb{F}$ defines the feasible region of the problem. For unconstrained optimization, $\mathbb{F}$ is most often defined simply as the region between the lower and upper bounds in each dimension, where

$$l(i) \geq x(i) \leq u(i), i = 1, ..., n. \tag{2.13}$$

Constrained optimization adds additional limits to the feasible region, specifically inequality and equality conditions:

$$g_i(x) \leq 0, i = 1, ..., p \tag{2.14}$$

$$h_j(x) = 0, j = q + 1, ..., m. \tag{2.15}$$

The basic PSO algorithm was applied to a number of standard CNOP benchmarks and consistently solved the majority of the problems therein[95]. Drawbacks were mostly confined to those that are present for most stochastic algorithms on CNOPs, namely that the equation constraints can be difficult to deal with due to the random nature of the update values, and the difficulty of generating an initial population contained within the feasible portion of the search space. PSO was shown to be substantially faster than the compared more established algorithms. A local topology was shown to return more accurate results[95].

Another study incorporated a penalty function into the particle fitness evaluation, allowing for a softer approach to the constraints and improved performance[96]. Penalty functions have been defined as

$$f'(x) = f(x) + h(k)H(x), x \in S \tag{2.16}$$

where k is the current iteration of the algorithm, $h(k)$ is a dynamic penalty function, and $H(x)$ is a penalty factor [97]. Without special parameter tuning, PSO produced results that were competitive with or superior to those reported for several other algorithms[96].

2.7.4.2 Multiobjective Optimization

Multiobjective optimization (MO) is a subfield of optimization wherein

> "...a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term optimize means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.[98]"

Formally, multiobjective optimization is a process by which optimal solutions for a vector of multiple objective functions is determined. For an *n*-dimensional search space *S* where $f_i(x), i = 1, ..., k$, k being the number of objective functions defined over *S*,

$$\vec{F(x)} = [f_1(x), f_2(x), ..., f_k(x)] \tag{2.17}$$

is the vector of objective functions and

$$x^* = (x_1^*, x_1^*, ..., x_1^*), x^* \in S \tag{2.18}$$

is the optimal solution that satisfies all necessary constraints and optimizes the objective function vector. Because it is not likely that all of the objective functions being optimized will contain the same global optimum at the same point in the search space, trade-offs must be made between the functions. The solution to a MO problem is a set of *Pareto optimal solutions* which are each unique, and which each solve the set of objective functions to the same degree of optimality. This set is called the *Pareto optimal set* and consists of vectors which are *non-dominated*, i.e. there exist no better solutions for the problem. The *Pareto front* is the plot of the objective functions for which the non-dominated vectors are in the Pareto optimal set.

While EAs have been applied to MO for some time, e.g. [99], PSO was first used for optimizing a benchmark of common problems more recently. Testing was done using a technique known as the *weighted aggregation* approach by the values of all objective functions at a candidate solution are summed to a weighted result $F = \sum_{i=1}^{k} w_i f_i(x), w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$ where the weights w_i can be either fixed or variable. PSO was shown to be effective on solving a number of two-function MO problems[100].

A PSO variant was developed separately but simultaneously for solving MO problems[83]. This proposed system was based on the existence of a global repository in which every particle recorded its experiences after each cycle. This repository was then used for each particle in the swarm to identify a leader that would guide its search. Results for this algorithm indicated a high degree of competitiveness with other current MO techniques[83].

2.7.4.3 Dynamic Optimization

Further specific research has been done into PSOs that operate on dynamic landscapes. These variants are adaptive to an extent; they are designed to expect changes in the optimization problem landscape and change their behaviour in order to account for this. The objective functions used in such dynamic problems usually operate as a function of time, altering the locations of their optima at regular or irregular intervals through the optimization process.

As long as the changes to the objective function are slow and regular enough, it was shown that a normal PSO with no changes was able to find and follow the optimum through the search space [101], while a minor adjustment to randomize the inertia weight $U(0.5, 1)$ also showed promising results for simple dynamic problems [57].

More complex dynamic problems are not so easily solved however, due to two factors [85]:

1. *Outdated memory* - Every time the objective function is updated, the best found positions of every particle becomes inaccurate, and hence liable to guide other particles toward suboptimal search areas.

2. *Diversity loss* - While convergence is usually desirable in static problems, when it takes place on dynamic ones it can eventually restrict the swarm to such a limited space that it is unable to re-diversify itself in order to search for new optima.

Many approaches have been taken to working with dynamic landscapes [84][6][96][102]. Most deal with the issue of outdated memory by assuming that changes to the problem landscape can be easily detected by means of stationary "sentry" particles that are constantly evaluated for changes to their objective value. When a change is detected, the personal best position for each particle in the swarm is reset to the current position. An alternative approach is to re-evaluate all of the best found positions and either reset them or leave them as they are, dependent on which option is associated with the better objective value [91].

Circumventing the second issue, that of diversity loss, is somewhat more difficult. Re-initializing some proportion of the particles in a swarm has been proposed [6], though this requires hard-coded input tuned to the specific problem as to when this should take place. Diversity control was built into the swarm in another approach, so that a cloud of non-converging "charged" particles surrounded a

nucleus of normal constricting particles [103][84]. This had the effect of maintaining diversity while allowing a core of particles to exploit knowledge of the discovered optimum. The most recent and most effective method involves the use of multiple swarms to discover and track all of the peaks in a dynamic environment using the previous charged particle method [104]. This technique was soon expanded to include the properties of *exclusion*, whereby swarms repel each other to avoid settling on the same peak, and *anti-convergence*: when all swarms have settled on a peak, the worst of them is completely re-initialized in the search space[85]. This ensures that there is constantly at least one swarm searching for newly-created peaks.

2.8 Applications of PSO

Despite its relative newness next to other optimization algorithms, PSO has proved very popular for use in practical applications. Detailing all of the diverse uses of the algorithm would require an entire report in itself; fortunately one such report does exist [105]. The most common and some of the most interesting uses are described in this section.

Neural network training is frequently used as a test for optimization algorithms; this was the case for PSO as well. One of the publications that originally proposed the algorithm used it for training the weights in several neural networks, and reported it to be as effective as the most common training method of backpropagation [2]. PSO was later shown to be superior to a gradient search technique for training a neural network for the XOR problem, a common benchmark [106]. A more practical approach with real-world results was undertaken after the development of inertia weight which used the algorithm to train a neural network to recognize patterns in the symptoms of patients suffering from essential tremor, a.k.a. Parkinson's disease [107].

On the topic of medical uses of PSO, the algorithm has been used on several occasions for cancer classification applications [108][109][110]. These applications used specialized versions of several optimization algorithms and reported that a properly tuned particle swarm algorithm showed very competitive performance for this purpose.

A previous review of the field [111] included various reported uses of the algorithm. One of these was for evolving asynchronous neural networks (both weights and structure) used for end milling [112]. Another involved use of a PSO in conjunction with backpropagation to train an asynchronous neural network as a state-of-charge estimator for battery packs used in electric vehicles [113].

Another popular use of PSO is in the design of electricity network and load dispatch. An example of this use can be seen in [114], where PSO is used to solve mixed-integer nonlinear problems, showing results competitive with established methods of the field. A very thorough examination of most of these applications to date can be found in [115].

Exhaustive listings of PSO publications on theory and applications can be found in [105] and [111].

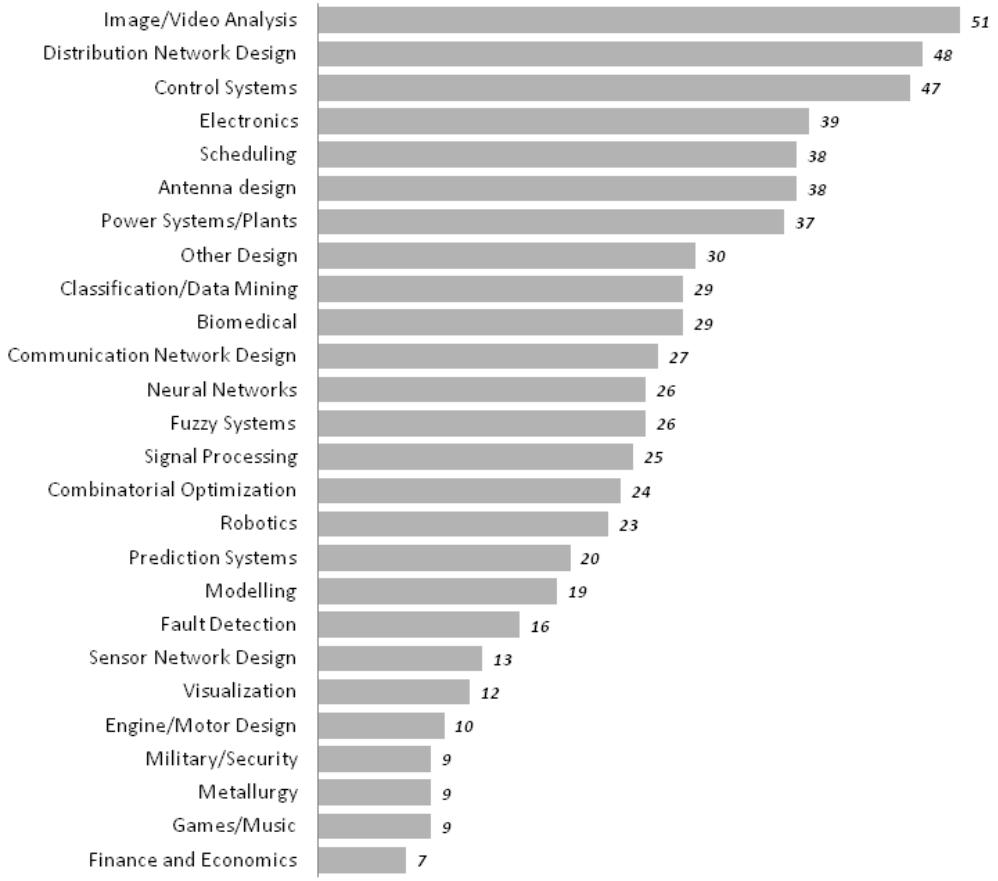


Figure 2.4: Proportion and Quantity of Publications in Fields of Application for PSO

Findings in [105, 111], as well as more specific lists of applications [116, 117, 118] are consolidated to table 2.4 to summarize the number of papers and proportionality of fields of application in PSO.

2.9 Summary

As early as the initial proposals[1, 2] it was realized that PSO was more than an attractive simulation of biological flocking and swarming. These behavioural characteristics could be exploited for optimization within the broader context of mathematical optimization. This field is well-defined and extensive, subsuming both local optimization, the search for a "good" solution, and global optimization, the search for the best solution. A number of methods for performing these optimizations have been introduced, many of which are also biologically inspired. Such methods are categorized as evolutionary optimizers when they use principles of evolution for their functioning, or swarm intelligence optimizers when they are based on the principles of communication and cooperation amongst a set of agents.

PSO is a method based on swarm intelligence, along with other such notable algorithms as stochastic diffusion search (SDS) and ant colony optimization (ACO). The large number of variants and applications described above are testament to its popularity. What is noticeably lacking from the lit-

erature, however, is a standard formulation beyond the original, first proposals[1, 2]. A great many improvements have been made to this extremely basic original algorithm, including methods to control instability[55, 10], a convergence proof[10], and research into the connective topologies between particles[62, 12]. With no well-defined standard the field has fragmented to some degree, with no common baseline of comparison and no common source from which to start research into variants.

At the same time, this lack of a common starting point has led to the development of very diverse and effective algorithms using PSO-style mechanics. These include FIPS[62, 81], the bare-bones PSOs[79, 80], Tribes[11], and numerous variants specialized for use on specific types of problems, seen in section 2.7.4. However, as each of these was developed independently without a means of comparison, determining the characteristics of performance in a comparative/competitive sense can only be accomplished by reimplementation by an interested party. Again, comparing against a known standard, working on a clear benchmark of problems, would provide a means of placing an algorithm amongst the other variants.

With the exception of the bare-bones algorithms[79, 80], the PSO variants discussed in this chapter were all developed by adding to or altering the functionality of PSO. This work proposes that a more effective approach can be taken by removing components of the update equations – ensuring that the original functionality is retained – and resulting in a stripped-down form that contains no unnecessary or extraneous calculations.

Finally, such a simplified form could then be used to advance the research of adaptive forms of PSO. In the same way that the original, static algorithm has been extended through the addition of new functionality and components of varying complexity, to date all adaptive forms of PSO have involved substantial alterations to the core algorithm. Currently there exists no simple, straightforward approach to adapting the variable parameters of the swarm that is easily reproducible and extendable. The goal of this work is to fill this gap in the field through development of such a system that is usable by any interested party for research into both improved performance on standard problem types, and expansion into areas for which the static algorithm has to be extensively specialized to achieve suitability.

Chapter 3

Defining a Standard for Particle Swarm Optimization

3.1 Introduction

In the years since the introduction of particle swarm optimization (PSO) as a new method for global optimization[1, 2], many researchers have expanded on the original idea with alterations ranging from minor parameter adjustments to complete reworkings of the algorithm. Others have used PSO for comparison testing of other global optimization algorithms, including genetic algorithms and differential evolution[39, 119]. The PSO field has expanded dramatically since its inception, but until recently there has been little to no consensus as to what constitutes the "standard" or "canonical" PSO algorithm. Despite regular usage of the term, the actual implementation of this undefined standard varies widely between publications. Also troublesome is the fact that many of the variations on the particle swarm algorithm that are used for comparison testing do not take into account some of the major developments that have taken place since the original algorithm was proposed.

This lack of cohesion is understandable to an extent - despite its simplicity, the large number of minor variations on the PSO algorithm have led to such a wide assortment of choices that it is often difficult to determine which version can be expected to give the best or most relevant performance for a given research question. Some variations show improved performance on a specific class of problems, while others may have been created or adapted for use on an entirely different type of search space. Further, many times a study focuses on the problem being solved, or on the details of comparative performance, and neglects to give a detailed explanation of the optimization algorithm or algorithms under investigation to the extent that both algorithm(s) and results can be reproduced.

Due to this often overwhelming amount of choice, many researchers have settled for using minor variations on the form of the PSO algorithm as originally proposed over 15 years ago. Unfortunately, this original algorithm is no longer in any way representative of the state-of-the-art in PSO and has

not been for some time. This chapter examines the major improvements that have taken place in the field since that original definition, and defines a standard for PSO that takes these into account. It is important to note that this definition should by no means be viewed as the optimal configuration for PSO across all problem sets, but rather as an evolution of the original algorithm designed to take advantage of subsequent generally applicable improvements that have come from a number of independent sources. Having a well-known, strictly-defined standard algorithm provides a valuable point of comparison which can be used both in this work as well as throughout the field of research to better test new concepts and alterations.

As a first step, it is necessary to define the exact properties and functioning of the PSO algorithm. A clear definition not only clarifies the research of this work as a whole, it also provides a standard, fixed version of the algorithm that can be used by any interested party as a representative baseline for comparative purposes.

This chapter defines such a standard form of PSO that will be used throughout the work. It takes into account the developments introduced to control instability in particle motion discussed in section 2.5.1 and the research into topology from section 2.5.2. Of the available options, a single approach to each of these aspects of the algorithm is adopted into the standard. These selections are based on a combination of results obtained through empirical research performed here, and simplicity.

There are several further aspects of the PSO optimization process that have, prior to this work, not been studied in as much detail, consistently applied, or proposed as standard settings in the same way as the previous developments. These are the initialization conditions for particles, the behaviour of particles when encountering a boundary, the number of particles in a swarm, and the process of examination of performance results. Each of these is examined in turn and a single approach or value is selected for use in the standard swarm.

Along with a straightforward laying out of the algorithm, information demonstrating what sort of performance and behaviour can be expected from this standard swarm is a necessary part of the definition. Empirical results are obtained and used as evidence of the optimization ability of this baseline, as well as aspects of its method for performing the optimization.

Finally, the full, exact definition of the algorithm is presented in a form facilitating implementation.

3.2 Definable Properties

3.2.1 Swarm Communication Topology

The *lbest* swarm model constitutes perhaps the most significant variation to the original PSO algorithm, and was in fact proposed in one of the very first PSO publications[1]. This topology did not commonly appear in PSO literature for quite some time, due to this original investigation that showed inferior performance when compared to the global *gbest* model. More recent research has revealed that when

given sufficient time to optimize, lbest swarms return improved results across many standard problem sets, particularly when used in conjunction with other improvements to the algorithm[87].

As mentioned previously, while lbest, or *local* can be used to describe any non-global topological model, both early research[1] and further major investigations into PSO topologies[62, 12, 87] have used the term in reference to a single $K = 2$ ring model – this practice is adopted in this work. While a number of different limited communication topologies have been tested with varying results as described above, the lbest ring model used here is the simplest, most reproducible form. The lbest ring model connects each particle to only two other particles in the swarm, in contrast to the gbest model where every particle is able to obtain information from the very best particle in the entire swarm population. Having the swarm converge when it has found the global optimum is obviously beneficial and necessary, but when the converged-upon location is suboptimal, the convergence is referred to as "premature" and is undesirable as it prevents the algorithm from escaping from an inferior local optimum.

Interestingly, it is the slower rate of convergence of the lbest model that was originally responsible for the disregard of it as an acceptable alternative topology[1]. The much faster convergence of the gbest model seems to indicate that it produces superior performance, but this is misleading. While results for the gbest model are indeed superior for many problems relatively early in the optimization process, the best found fitness for the lbest model quite often surpasses that of the gbest after some sufficient number of function evaluations have been performed, particularly on multimodal problems.

Despite the advantages of a local topology, it is important to note that it should not always be considered to be the best choice in all situations. The faster convergence rate of a global topology will usually result in better performance on simple unimodal problems than that of any non-global topology due to the lack of any danger of convergence to a suboptimal local minimum. Even on some very complex multimodal problems a gbest model swarm can deliver performance competitive with the lbest model given proper circumstances. For thorough, rigorous results tests should be run using both topologies, but in the interest of defining a single standard PSO algorithm, the superior performance of the lbest model over the majority of benchmarks qualifies it as the better choice for cases where a straightforward means of comparison is desired. In any case, modern research performed using only swarms with a global topology is incomplete at best.

The inclusion of the local ring topology as part of a standard algorithm for particle swarm optimization comes with a caveat, however. Given the slower convergence of the lbest model, more function evaluations are required for the improved performance to be seen. This is especially important on unimodal functions, where the fast convergence of the gbest model combined with a single minimum in the feasible search space results in quicker performance than that of the lbest swarm with its limited communication.

3.2.2 Initialization and Boundary Conditions

It has been suggested that many optimization algorithms have what is described as a *bias* toward some area of the space in which the population is initialized. Research has shown that some algorithms can return superior performance when the global optimum of the problem being solved is located in or near the center of the area in which the swarm is initialized[120].

This is especially problematic in algorithm comparison: as a basic example, it is simple to construct an algorithm that instantly finds the point at the center of the feasible bounds through a single mathematical equation. If that point is also the global optimum, the simple algorithm would appear to show superior optimization performance to any other heuristic algorithm, when in actuality the algorithm's ability to find that optimum would be totally dependent upon it being located at the easily-calculated center of the search space. Benchmark problems with the global optimum at the center of the feasible bounds are common in optimization literature, so adjustment is necessary for accurate performance testing.

The common method for negating any centrist bias in an optimization algorithm is to *shift* the function when the optimum is defined at the center of the feasible search space. This is also known as the *center offset* method. Moving the optimum away from the point which the algorithm is supposedly biased toward reduces or eliminates any inappropriate advantage that may be gained.

Another proposed method of alleviating this potential bias is population initialization within a subspace of the entire feasible search space that does not contain the global optimum[121], often referred to as *region scaling*. Ensuring that the global optimum does not lie within this subspace forces the swarm to expand its search beyond its initial limits, and eliminates its ability to immediately converge to a single point without exploring the entire space to find the global optimum of the problem.

Both these methods are most applicable as research standards for performance testing and algorithm comparison when both the problem and its optimum are well-known and understood; for practical optimization applications they are obviously unnecessary and self-defeating.

It can be shown that a bias toward the center can arise in algorithms similar to PSO when the trajectory of a particle is artificially limited at the boundary of the search space[120]. To avoid any effect on the performance of the algorithm, the simplest and most straightforward method for handling particles which cross the boundary of the feasible search space is to leave their velocity and infeasible position unaltered. The fitness evaluation step is then skipped, thereby preventing the infeasible position from being set as a personal and/or global best. Using this method, particles outside the feasible search space will eventually be drawn back within the space by the influence of their personal and neighborhood bests. As it is possible under certain very rare circumstances for particles to develop extremely high velocities when they are allowed to continue past the edge of the defined search space, the use of a very generous *vmax* value has been recommended to keep the particle from going too far beyond the feasible region

[60]. Like the original necessary *vmax* parameter, this optional value has never been explicitly defined, but for the purposes of this study it was set to be $10\times$ the size of the search space in each dimension, i.e. $10 \times (xmax - xmin)$. While this is good practice to account for extraordinary cases, in all of the simulations performed in this work the limit was never reached – constriction always prevented particle velocity from growing beyond acceptable levels.

These boundary conditions, referred to colloquially as "letting the particles fly"[122], prevent this from being a contributing factor in any potential chance of a bias toward the center of the search space; region-scaled swarm initialization and shifting in the case of a centrally-located optimum will substantially reduce or eliminate the risk of such a centrist bias altogether.

3.2.3 Number of Particles

Empirical results from research into PSO have shown that the number of particles composing the swarm can influence the resulting performance by a varying amount, depending on the problem being optimized[123]. Some test functions show improved performance as the size of the swarm is increased, while others tend to be better optimized by smaller swarms. There seems to be no definitive value for the swarm size that is optimal across all problems, even in a very limited benchmark, so to avoid tuning the algorithm to each specific problem, a compromise must be reached.

This compromise has to take into account a number of different effects that the swarm size has on performance and behaviour. In the case of unimodal problems, swarms need only enough particles to converge to the single optimal point – once the minimum number of particles necessary for this has been determined, every particle over that amount is only adding an extraneous function evaluation at every iteration. This isn't as much the case for multimodal landscapes however, as larger swarms are able to explore more of the landscape, and take longer to settle down to a single basin of attraction. This parameter further interacts with the topology being used: a swarm using an lbest ring topology will disseminate information to the entire swarm more and more slowly with each additional particle, while speed of dissemination for a swarm using a gbest global topology is unaffected by the number of particles. Even further, the dimensionality of the landscape plays a role in the effect of this compromise. The size of a landscape increases exponentially as the number of dimensions is increased, necessitating larger and larger swarm sizes for effective optimization.

The fact that optimal swarm size is highly dependent on the problem to which the swarm is being applied means that a single value for this property in a standard definition is correspondingly dependent on the benchmark of problems on which the standard is tested. The benchmark used here is defined, and the experimental process given, in section 3.3.1, and a single value for the standard is selected as the one for which the best overall performance is obtained via this empirical process.

3.2.4 Statistical Analysis of Results

Having gathered some empirical data, differences in performance between several versions of algorithms can become notable. One version may be more likely than another to reach a criterion, or the final function result after some number of function evaluations or iterations may be better, averaged over some number of trials. The real issue, though is whether or not it is *significantly* better – in other words, is there a real difference between the two versions, or did chance play the deciding role.

It is not necessary to undertake any statistical analysis that is especially complicated, difficult, or time-consuming when comparing results taken from the application of algorithms such as PSO. The practice of performing t-tests on pairs of groups of data gives a *p-value* which is compared to a constant α to determine whether a difference is significant or not.

There is a problem, however, in conducting multiple significance tests. Because they are probabilistic, it is possible that some results are due simply to chance – even random data generated from the same distribution will differ "significantly" sometimes. Statisticians have addressed this problem in various ways. Corrections known in the literature include Duncan, Bonferroni, Sheffé, Fisher, and Tukey adjustments, amongst others. These approaches typically manipulate α or the p-value in some way that corrects for the likelihood of forming spurious conclusions due to chance.

Unfortunately, many of these corrections are too conservative – good results are rejected simply because the adjustment was too severe. A good, widely accepted alternative is a modified Bonferroni procedure known as Holm-Bonferroni[124], which manipulates the α value in a way that protects against capitalization on chance, without penalizing the obtained results[125].

A number of t-tests are conducted, and p-values recorded. These are ranked from smallest to largest, and the ranks recorded. These ranks are then inverted, so that the highest p-value gets an inverse-rank of 1, and the test with the lowest p-value gets an inverse-rank of N, the number of tests performed.

Next, α is divided by the inverse rank for each observation. Assuming the typical desired $\alpha = 0.05$, then the first observation receives a new value of $\alpha' = 0.05/N$, and so on, with the worst result receiving a new value of $\alpha' = 0.05/1 = 0.05$.

These new values of α' are then compared down the list of inverse-ranked t-tests, checking each p-value against the new α' until a nonsignificant result is found. If $p < \alpha'$, then a test is reported as significant. When the first nonsignificant result is encountered, the subsequent observations can be taken as nonsignificant and reported as such.

This procedure is used to validate t-tests, which compare pairs of groups. In comparing more than two groups over a set of functions, it may be necessary to perform a great number of t-tests. In this case, it may be wise to reject certain tests at the beginning, for instance if one group is always worse than others, or if some means are identical. This cuts down on the number of tests and keeps α' from becoming too small.

3.3 Results

3.3.1 Performance

Three PSO algorithms were compared in the interest of demonstrating the performance gains granted by improvements to the technique: the original 1995 algorithm, a constricted swarm using a global topology, and a constricted swarm using a local topology.

Each algorithm was run on an array of common benchmarks, shown in table 3.1 on page 51, for 600000 evaluations per function. These benchmarks were chosen for their variety - functions $f_1 - f_3$ are unimodal functions with a single minimum, $f_4 - f_9$ are complex high-dimensional multimodal problems, each containing many local minima and a single global optimum, and $f_{10} - f_{14}$ are lower-dimensional multimodal problems with few local minima and a single global optimum apart from f_{10}, which is symmetric about the origin with two global optima, as well as their previous definition in well-known and respected publications[35, 126]. All swarms were randomly initialized in an area equal to one quarter of the feasible search space in every dimension that was guaranteed not to contain the optimal solution. Initialization ranges for all functions can be found in table 3.2 on page 54.

Performance was measured as the minimum error $|f(x) - f(x^\star)|$ found over the trial, where $f(x^\star)$ is the optimum fitness for the problem.

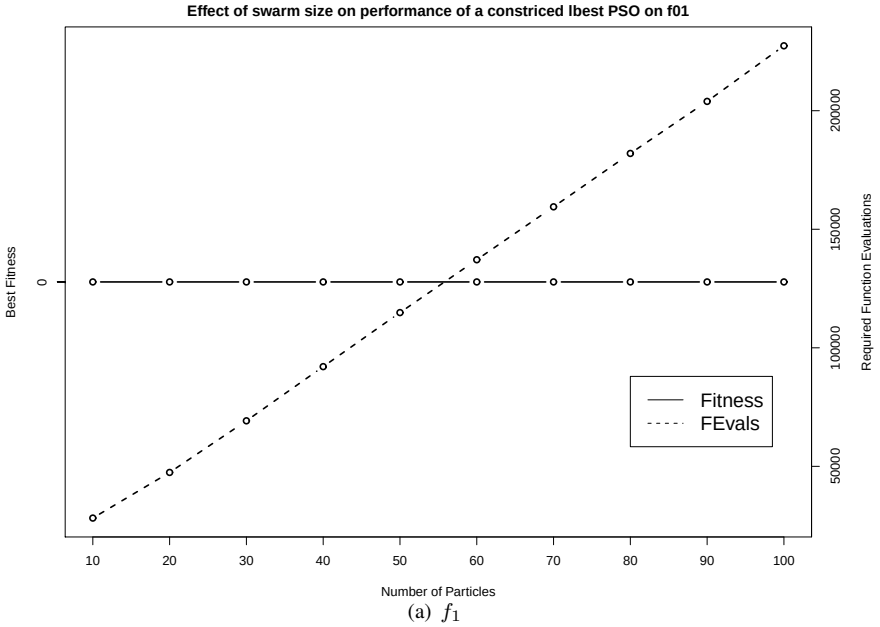
Plots of relative performance for various swarm sizes under an lbest topology are shown in figures 3.1 and 3.2 for problems on which the swarm was able to find the global optimum on at least 10% of trials for each size, and hence return a value for the number of function evaluations required. These results demonstrate the complexity of choosing a single best value. The unimodal f_1 problem was solved by every swarm configuration, but the number of necessary function evaluations increased linearly as the swarm size was increased. Swarms with more particles were better able to solve the multimodals f_7 and f_9, but the number of necessary function evaluations rose on either side of specific "best" configurations. The three shekel problems, f_{12}–f_{14}, seemed to show little correlation between swarm size and speed of successful optimizations. These types of complex results were seen for the entire benchmark.

The complications involved in recommending a single value for swarm size make this a difficult aspect of defining a standard. Fortunately, this standard is not intended to be the single best available configuration of the particle swarm algorithm, but merely an indicator of what can be expected as a reasonable configuration, providing acceptable performance. In support of this, swarms of almost all reasonable sizes have been shown to perform *acceptably* – i.e. significantly better than a random search – on most standard benchmarks. 50 particles were used in the final results presented here, as swarms of this size performed best by a very slight margin when averaged across the entire range of test problems.

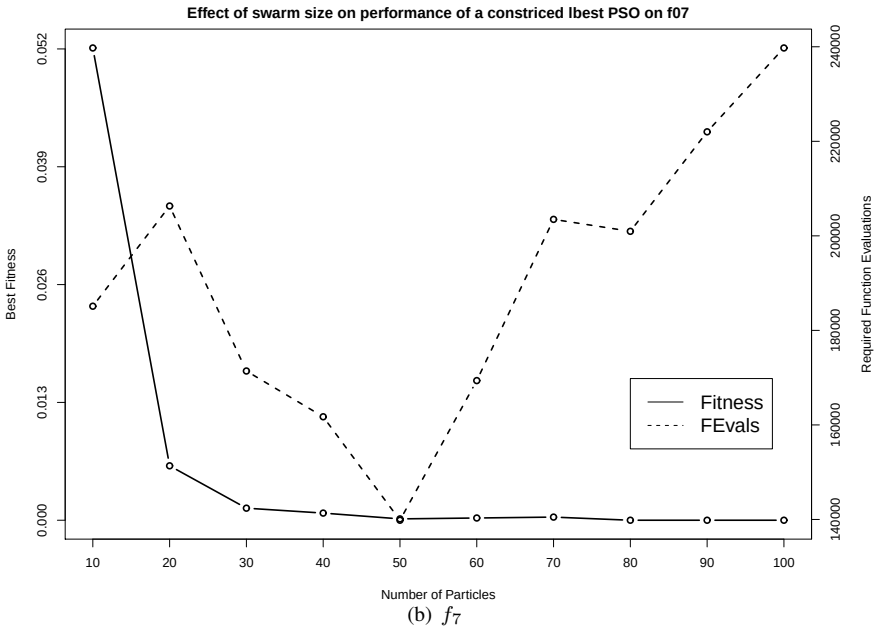
It should be taken into account, however, that this improved performance is tied directly to the benchmark that was used in this study, and even then is still an average - no one value was clearly

Table 3.1: Benchmark functions

Equation		Name	D	Feasible Bounds
f_1 =	$\sum_{i=1}^{D} x_i^2$	Sphere/Parabola	30	$(-100, 100)^D$
f_2 =	$\sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	Schwefel 1.2	30	$(-100, 100)^D$
f_3 =	$\sum_{i=1}^{D-1} \left\{ \{100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right\}$	Generalized Rosenbrock	30	$(-30, 30)^D$
f_4=	$-\sum_{i=1}^{D} x_i \sin\left(\sqrt{x_i}\right)$	Generalized Schwefel 2.6	30	$(-500, 500)^D$
f_5=	$\sum_{i=1}^{D} \left\{ x_i^2 - 10\cos\left(2\pi x_i\right) + 10 \right\}$	Generalized Rastrigin	30	$(-5.12, 5.12)^D$
f_6=	$-20\exp\left\{ -0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right\} - \exp\left\{ \frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi x_i\right) \right\} + 20 + e$	Ackley	30	$(-32, 32)^D$
f_7=	$\frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank	30	$(-600, 600)^D$
f_8=	$\frac{\pi}{D}\left\{ 10\sin^2\left(\pi y_1\right) + \sum_{i=1}^{D-1} \left(y_i - 1\right)^2 \left\{ 1 + 10\sin^2\left(\pi y_{i+1}\right) \right\} + \left(y_D - 1\right)^2 \right\}$ $+ \sum_{i=1}^{D} \mu\left(x_i, 10, 100, 4\right)$ $y_i = 1 + \frac{1}{4}\left(x_i + 1\right)$ $\mu\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k\left(-x_i - a\right)^m & x_i < -a \end{cases}$	Penalized Function P8	30	$(-50, 50)^D$
f_9=	$0.1\left\{ \sin^2\left(3\pi x_1\right) + \sum_{i=1}^{D-1} \left(x_i - 1\right)^2 \left\{ 1 + \sin^2\left(3\pi x_{i+1}\right) \right\} + \left(x_D - 1\right)^2 \times \left\{ 1 + \sin^2\left(2\pi x_D\right) \right\} \right\} + \sum_{i=1}^{D} \mu\left(x_i, 5, 100, 4\right)$	Penalized Function P16	30	$(-50, 50)^D$
f_{10}=	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump Camel-back	2	$(-5, 5)^D$
f_{11}=	$\left\{ 1 + \left(x_1 + x_2 + 1\right)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right) \right\} \times$ $\left\{ 30 + \left(2x_1 - 3x_2\right)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right) \right\}$	Goldstein-Price	2	$(-2, 2)^D$
f_{12}=	$-\sum_{i=1}^{5} \left\{ \sum_{j=1}^{4} \left(x_j - a_{ij}\right)^2 + c_i \right\}^{-1}$	Shekel 5	4	$(0, 10)^D$
f_{13}=	$-\sum_{i=1}^{7} \left\{ \sum_{j=1}^{4} \left(x_j - a_{ij}\right)^2 + c_i \right\}^{-1}$	Shekel 7	4	$(0, 10)^D$
f_{14}=	$-\sum_{i=1}^{10} \left\{ \sum_{j=1}^{4} \left(x_j - a_{ij}\right)^2 + c_i \right\}^{-1}$	Shekel 10	4	$(0, 10)^D$

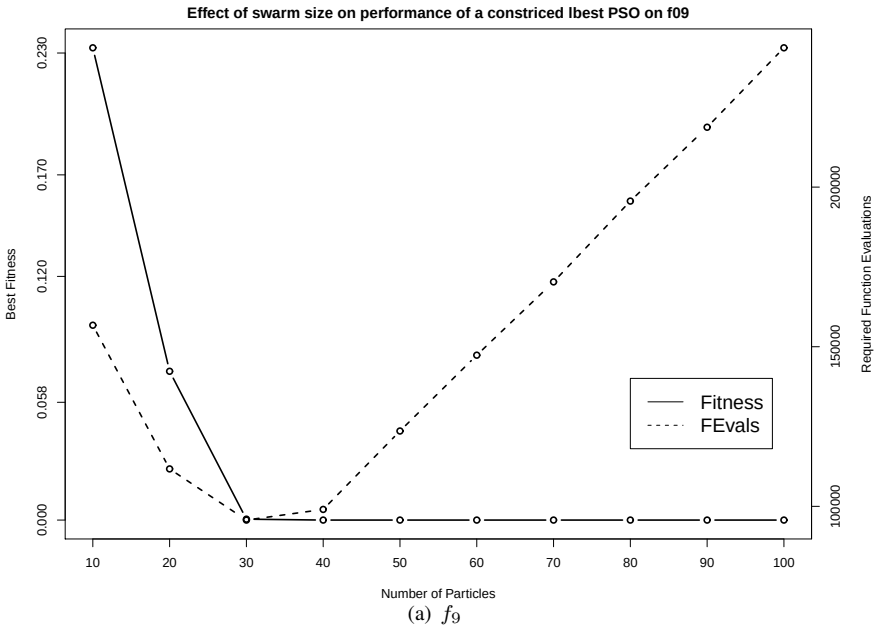


(a) f_1

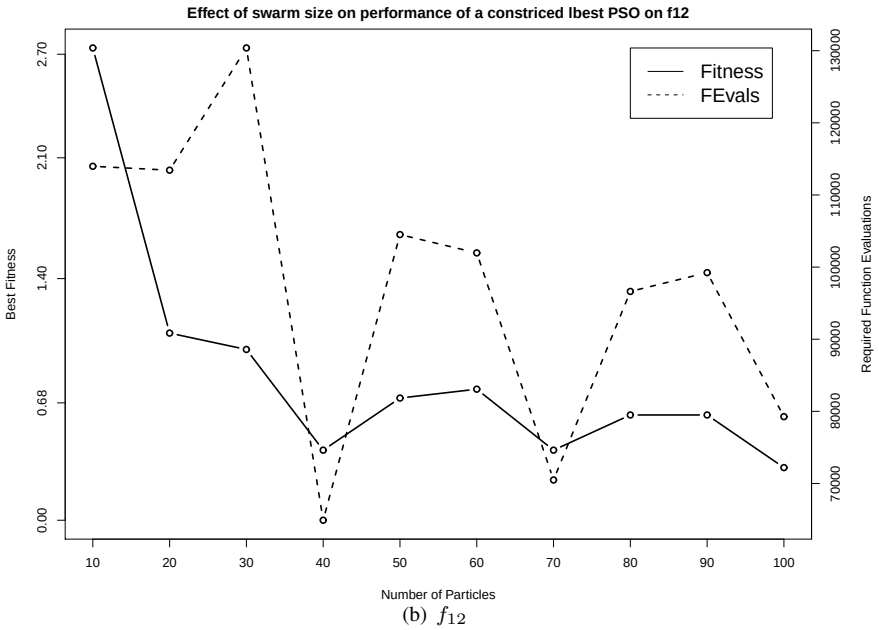


(b) f_7

Figure 3.1: Effect of swarm size on performance on problems f_1 and f_7. Mean performance over 50 trials, and number of function evaluations required in trials where the optimum was attained are shown.



(a) f_9



(b) f_{12}

Figure 3.2: Effect of swarm size on performance on problems f_9 and f_{12}. Mean performance over 50 trials, and number of function evaluations required in trials where the optimum was attained are shown.

Function	Feasible Bounds	Optimum	Initialization
f_1	$(-100, 100)^D$	0.0^D	$(50, 100)^D$
f_2	$(-100, 100)^D$	0.0^D	$(50, 100)^D$
f_3	$(-30, 30)^D$	1.0^D	$(15, 30)^D$
f_4	$(-500, 500)^D$	420.9687^D	$(-500, -250)^D$
f_5	$(-5.12, 5.12)^D$	0.0^D	$(2.56, 5.12)^D$
f_6	$(-32, 32)^D$	0.0^D	$(16, 32)^D$
f_7	$(-600, 600)^D$	0.0^D	$(300, 600)^D$
f_8	$(-50, 50)^D$	-1.0^D	$(25, 50)^D$
f_9	$(-50, 50)^D$	1.0^D	$(25, 50)^D$
f_{10}	$(-5, 5)^D$	$(-0.0898, 0.7126)$, $(0.0898, -0.7126)$	$(2.5, 5)^D$
f_{11}	$(-2, 2)^D$	$(0, -1)$	$(1, 2)^D$
f_{12}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$
f_{13}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$
f_{14}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$

Table 3.2: Optima and initialization ranges for all functions

		Random Search	**Original PSO**	**Constricted** *LBest*	**Constricted** *GBest*
f_1	Success	0%	0%	**100%**	**100%**
	Best	2.11E4	1.91	**0.0**	**0.0**
	Mean	3.27E4±460	2.55±0.04	**0.0±0.0**	**0.0±0.0**
	Worst	3.82E4	3.01	**0.0**	**0.0**
	FEvals	-	-	109253±360	**39262±312**
f_2	Success	0%	0%	0%	**100%**
	Best	2.57E4	3.31	8.66E-8	**0.0**
	Mean	3.47E4±485	4.64±0.08	2.39E-6±4.86E-7	**0.0±0.0**
	Worst	4.13E4	5.98	2.27E-5	**0.0**
	FEvals	-	-	-	**314435±1751**
f_3	Success	**0%**	**0%**	**0%**	**0%**
	Best	3.25E7	42.83	1.32E-4	**5.47E-8**
	Mean	6.13E7±1.28E6	151.2±24.0	**2.81±0.55**	3.29±1.45
	Worst	7.78E7	596.1	**14.36**	72.3
	FEvals	-	-	-	-

Table 3.3: Results for original PSO vs constricted form on unimodal problems

optimal on all problems. In the results obtained, no swarm size between 20 – 100 particles produced results that were clearly superior or inferior to any other value for a majority of the tested problems.

Results for each problem were averaged over 50 independent trials, and are displayed in tables 3.3 – 3.5. Convergence plots for selected functions are shown in figures 3.3–3.4. In cases where a value was $< 10^{-15}$ it was rounded to 0.0 in order to accommodate reproduction using programming languages that may not guarantee floating point precision at smaller values. Results that attained this level of error were counted as "successful", though this should be taken only as a means of brief high-level comparison – it is possible when comparing two algorithms for one to attain a better mean error but a lower success rate, as in the case here of f_6, where the gbest swarm was successful in fewer cases, but performed better on average than the lbest swarm. Which measure is preferable depends on the information that is

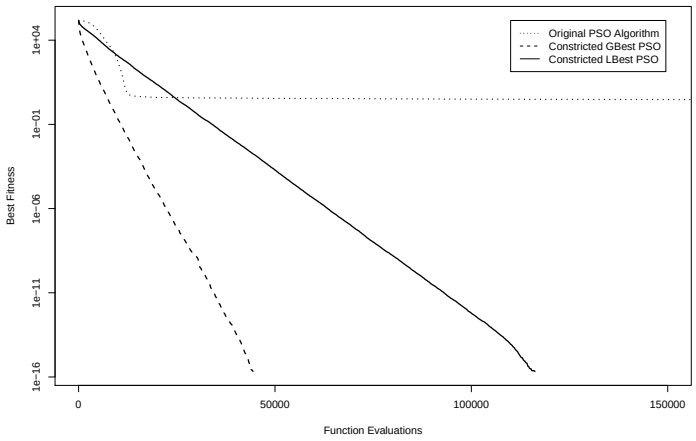
		Random Search	**Original PSO**	**Constricted** *LBest*	**Constricted** *GBest*
f_4	Success	**0%**	**0%**	**0%**	**0%**
	Best	6816	3561	2961	**2803**
	Mean	7559±34	4126±43	**3264±21**	3536±39
	Worst	7910	4769	**3614**	4185
	FEvals	-	-	-	-
f_5	Success	**0%**	**0%**	**0%**	**0%**
	Best	278.1	376.6	98.50	**66.66**
	Mean	304.7±1.68	416.6±2.99	149.0±3.48	**129.4±3.83**
	Worst	325.6	462.7	198.0	**193.0**
	FEvals	-	-	-	-
f_6	Success	0%	0%	**20%**	18%
	Best	18.05	20.13	**0.0**	**0.0**
	Mean	19.24±0.04	20.23±0.006	14.68±1.16	**13.6±1.23**
	Worst	**19.56**	20.34	19.75	19.79
	FEvals	-	-	239923±39688	**84809±1725**
f_7	Success	0%	0%	**98%**	32%
	Best	191.3	0.965	**0.0**	**0.0**
	Mean	295.7±4.14	1.0±0.002	**1.48E-4±1.48E-4**	1.83E-2±3.45E-3
	Worst	345.1	1.03	**7.4E-3**	0.10
	FEvals	-	-	124726±4922	**39818±351**
f_8	Success	0%	0%	**100%**	64%
	Best	2.63E7	7.31	**0.0**	**0.0**
	Mean	8.5E7±2.87E6	14.06±0.7	**0.0±0.0**	1.79E-1±5.26E-2
	Worst	1.28E8	30.44	**0.0**	1.97
	FEvals	-	-	128167±1107	**46294±1366**
f_9	Success	0%	0%	**100%**	74%
	Best	9.2E7	0.082	**0.0**	**0.0**
	Mean	2.29E8±6.2E6	0.112±0.002	**0.0±0.0**	4.61E-3±2.04E-3
	Worst	3.13E8	0.134	**0.0**	9.89E-2
	FEvals	-	-	118098±440	**43565±1066**

Table 3.4: Results for original PSO vs constricted form on complex multimodal problems.

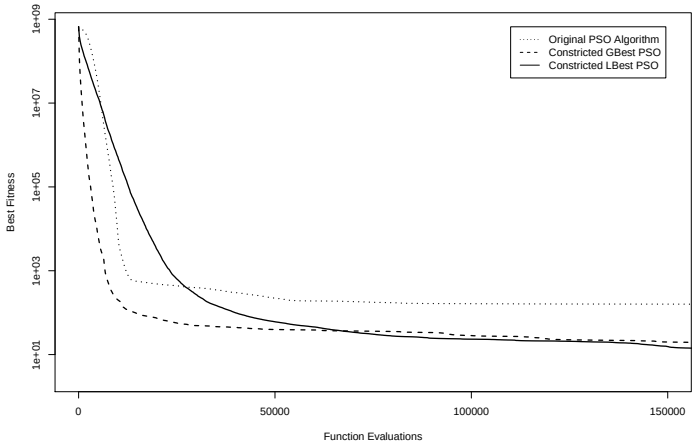
desired. In this case, significance aside, the gbest swarm can be expected to return a better result even in unsuccessful trials, while the lbest swarm can be expected to be successful more often.

Results for a random search of the problem landscape over the same number of function evaluations are included as a point of comparison for the performance of each algorithm. It is interesting to note that for seven of the fourteen problems, the original PSO algorithm is outperformed by this method. This is in part due to the fact that in its original form, the algorithm had no means of guaranteed convergence, and was thus unable to fully exploit discovered optima. Additionally, and perhaps more importantly, the original algorithm used a global communication topology, from which the fast information exchange rate means that all particles are attracted to the same point, making discovery of new peaks more difficult. This is supported by the fact that each of the problems which were better optimized by the random search are multimodal, and further by how the random search gave equal or superior performance to the constricted gbest swarm on f_{12}, f_{13}, and f_{14}.

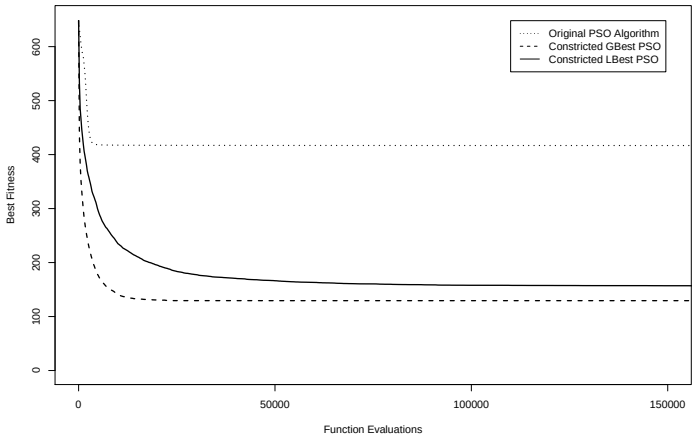
Statistical tests were performed on these results to show whether performance improvements are significant. As the performance of the original PSO algorithm is poorer than the others for all but a single function where it was statistically equivalent to the constricted gbest swarm (f_{12}), these tests were limited to comparisons of the two constricted swarm models over all functions. These results are shown in table 3.6. The lbest swarm showed significantly superior performance over the gbest swarm for six of



(a) f_1 (Sphere/Parabola)

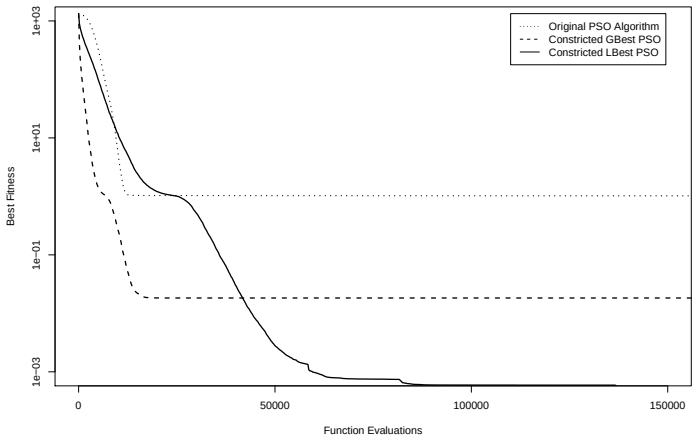


(b) f_3 (Generalized Rosenbrock)

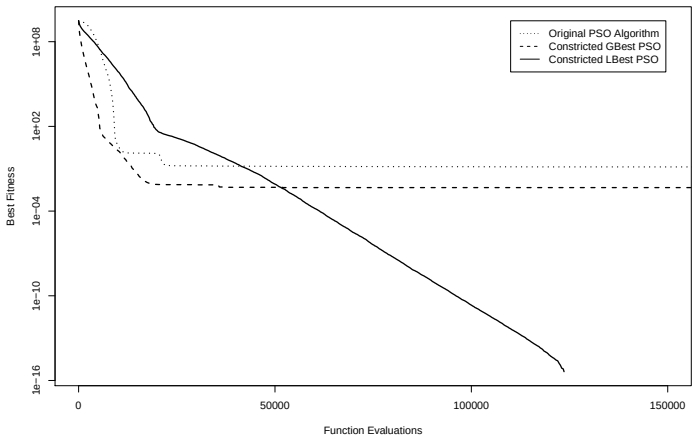


(c) f_5 (Generalized Rastrigin)

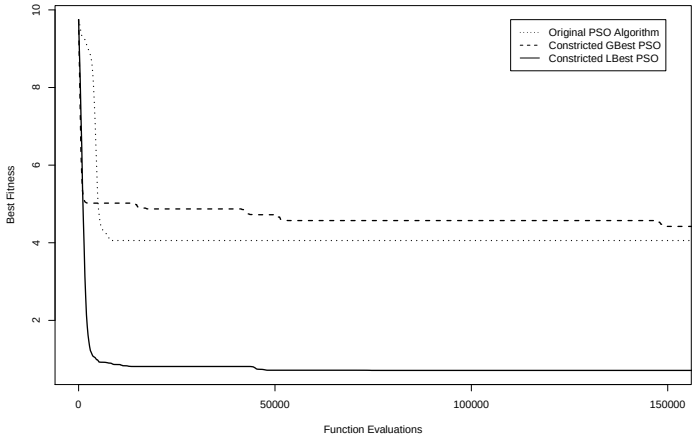
Figure 3.3: Mean algorithm convergence for problems f_1, f_3, and f_5.



(a) f_7 (Generalized Griewank)



(b) f_9 (Penalized Function P16)



(c) f_{12} (Shekel 5)

Figure 3.4: Mean algorithm convergence for problems f_7, f_9, and f_{12}.

		Random Search	**Original PSO**	**Constricted** *LBest*	**Constricted** *GBest*
f_{10}	Success	0%	0%	**100%**	**100%**
	Best	8.03E-6	2.38E-10	**0.0**	**0.0**
	Mean	1.75E-4±2.26E-5	1.25±0.191	**0.0±0.0**	**0.0±0.0**
	Worst	6.97E-4	3.13	**0.0**	**0.0**
	FEvals	-	-	13528±228	**13266±301**
f_{11}	Success	0%	0%	**100%**	**100%**
	Best	7.82E-6	4.87E-9	**0.0**	**0.0**
	Mean	2.15E-3±3.23E-4	239.8±53.2	**0.0±0.0**	**0.0±0.0**
	Worst	1.03E-2	837.0	**0.0**	**0.0**
	FEvals	-	-	9313±81	**7258±66**
f_{12}	Success	0%	0%	**86%**	28%
	Best	0.774	9.05E-5	**0.0**	**0.0**
	Mean	3.45±0.183	4.06±0.489	**0.708±0.251**	4.42±0.42
	Worst	6.06	7.52	**5.10**	7.52
	FEvals	-	-	**21751±3860**	29565±11193
f_{13}	Success	0%	0%	**88%**	44%
	Best	0.771	5.79E-5	**0.0**	**0.0**
	Mean	3.44±0.182	4.64±0.491	**0.823±0.323**	3.66±0.48
	Worst	**6.06**	8.57	7.64	8.57
	FEvals	-	-	28871±12526	**25992±10598**
f_{14}	Success	0%	0%	**90%**	54%
	Best	0.772	3.28E-5	**0.0**	**0.0**
	Mean	3.44±0.182	5.82±0.437	**0.759±0.326**	3.06±0.49
	Worst	**6.07**	8.86	8.11	8.86
	FEvals	-	-	**17690±861**	37553±15140

Table 3.5: Results for original PSO vs constricted form on simple multimodal problems.

the fourteen test functions (f_4, f_7–f_8, and f_{12}–f_{14}) while the gbest swarm was significantly superior on two functions (f_2 and f_5). There was no significant difference in performance between the two swarm models for the other six test functions.

These results show that both the global and the local models of constricted swarms return significantly improved performance over the original PSO algorithm. Further, it is clear that in many of the test cases, the lbest model swarm can be reliably expected to return better results than the gbest model swarm. This improved performance is sufficient to make the local ring topology the single recommended topology for the purposes of this study, though it must be stressed that any complete comparison should at a minimum make mention of the performance of the gbest configuration.

3.3.2 Behaviour

The phenomenon previously demonstrated in figure 2.3 from section 2.5.2.1 helps to account for the very different behaviour and performance of swarms with an lbest communication topology when compared to those using a gbest topology. While a globally-connected swarm passes information about the best found position to all particles instantaneously, the locally-connected swarm only passes on the information about the best found position of a limited subset of the entire population. This leads to individual particles "knowing" about a point of attraction that may or may not be the same point that is known to other parts of the swarm, which has the effect of dividing the population into multiple sub-populations, each one attracted to a different area of the problem landscape.

With this concept in mind, we can postulate that generally speaking, there will be three different

Func	p-value	Rank	Inverse rank	New α	Significant
f_{12}	5.05E-11	1	11	0.004545	Yes
f_4	2.86E-8	2	10	0.005	Yes
f_7	3.09E-6	3	9	0.005556	Yes
f_{13}	0.000005	4	8	0.00625	Yes
f_2	0.000011	5	7	0.007143	Yes
f_{14}	0.000196	6	6	0.008333	Yes
f_5	0.000261	7	5	0.01	Yes
f_8	0.001373	8	4	0.0125	Yes
f_9	0.02791	9	3	0.016667	No
f_3	0.5245	10	2	0.025	No
f_6	0.7592	11	1	0.05	No
f_1	=	12	-	-	No
f_{10}	=	13	-	-	No
f_{11}	=	14	-	-	No

Table 3.6: Detailed significance findings for constricted swarms with gbest vs lbest topologies

phases that a swarm goes through during an optimization. Two of these three phases are already well-known in the optimization literature as exploration and exploitation. What is proposed here is a third phase, the *discovery* phase.

This phase occurs between exploration and exploitation, at the point where the swarm has discovered *multiple* optima, and is simultaneously exploiting each one in an effort to determine which contains the best single point. Depending on the relative fitness of the optima and the exploitation ability of the sub-population on each one, this phase can be very short, or extremely prolonged.

The most extreme example of this phase is found when a swarm is set to optimize a landscape containing multiple global optima with equal fitness. As none of the sub-swarms is able to find an position with a superior level of fitness over the other sub-swarms, each of them maintains its attraction to the peak on which its particles have set their personal best positions via the cognitive component of the velocity update, $(p_i - x_i)$. A particle at the "edge" of the sub-swarms may be attracted to the different optimum being exploited by its neighbors via the social component $(p_g - x_i)$, but as it will be unable to find a position that is an improvement to its own p_i which will allow it to update this value, this attraction will only ever account for a portion of its movement – it will never be able to break away from its attraction to its current personal best position on the peak that the subswarm it is a part of has exploited.

In practice, this situation can arise even on landscapes where there is a single discovered global optimum. Given infinite function evaluations, every particle in the swarm will eventually be drawn to the single best point discovered. In finite terms, however, the number of function evaluations required

for this to happen is based on multiple factors, most notably the weighting of the social term, and the speed of communication between particles. A globally-connected swarm will settle down to a single point relatively quickly, as every particle will be immediately attracted to the best found point. With a minimally-connected ring topology, however, a particle on the opposite side of the ring from the particle at the global optimum must wait until every intervening particle, $N/2$ of them, has also found this global optimum before it can be informed of its location. This can result in situations where the entire swarm never converges to a single optimum within the lifetime of the optimization.

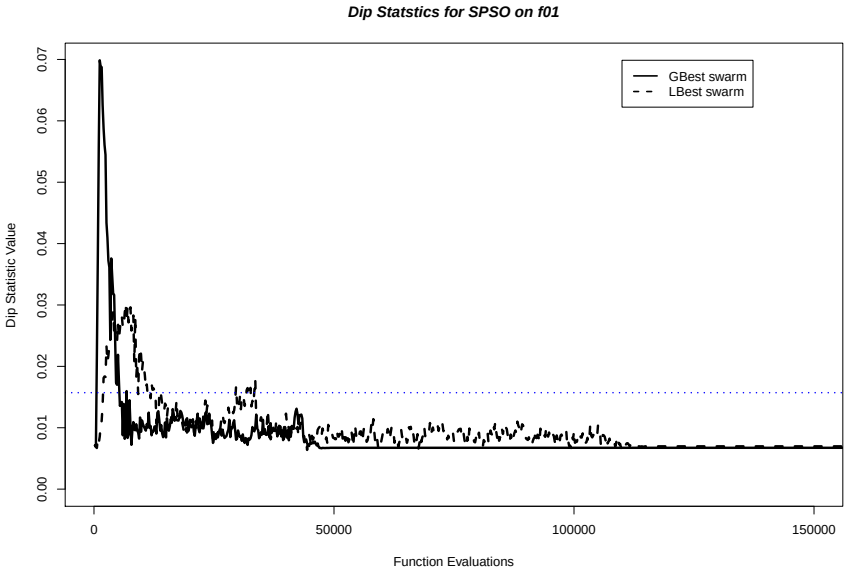
Naturally, whether or not a prolonged discovery phase is beneficial to optimization depends on the problem being optimized. A swarm optimizing a unimodal problem has no need for discovery, so simultaneous exploitation of multiple points on the single peak would be of limited value. A swarm on a highly multimodal problem, on the other hand, would benefit from the ability to temporarily optimize several different peaks, eventually "choosing" the single best for exploitation by the entire swarm.

While the concept of a discovery phase seems reasonable, and can be directly observed via images like figure 2.3, what is needed is an actual measure that indicates the current phase of the swarm. Hartigan's *dip test of unimodality*[127] was selected to fulfill this role. This test produces a single value that indicates the level of uni/multimodality in the distribution of the distances between the best found positions of the particles in a swarm by calculating the difference between that distribution and the unimodal distribution that it is nearest to, i.e. the "best fit." A swarm with particles that are perfectly uniformly distributed will have a dip statistic value of 0.0, which increases as the distribution becomes more and more multimodal. Whether or not the calculated value indicates a multimodal distribution is dependent on the sample size and the desired level of probability.

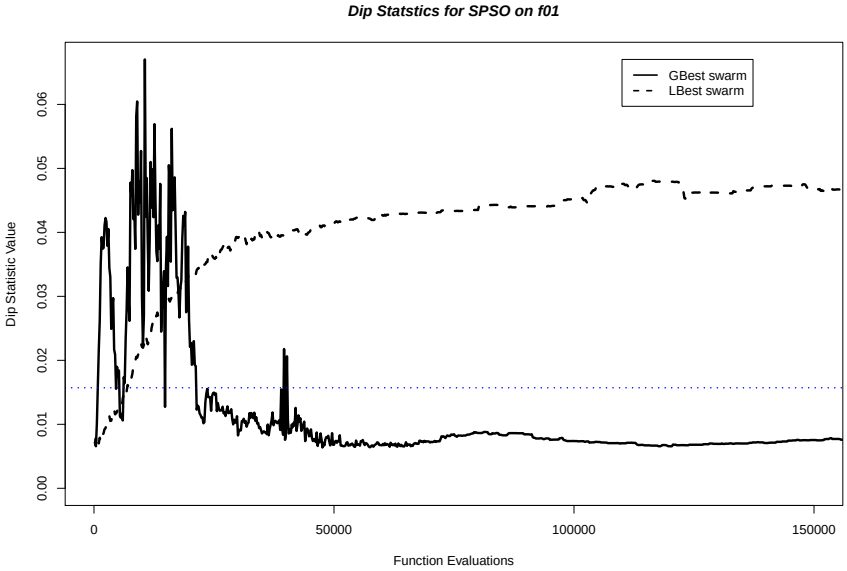
In the case presented here, the swarm size is set to $N = 50$ particles (see section 3.2.3), giving a sample size of $\frac{50 \times 49}{2} = 1225$ particle separations, and our desired probability is the typical $\alpha = 0.95$. According to a reference table of values calculated for numerous sample sizes[128], the threshold of multimodality was set to $dip \approx 0.0157$ – distributions of this size with values below this level will be unimodal with 95% probability, those above will be multimodal with the same probability.

Figures 3.5 and 3.6 compare the dip values of a swarm with a global topology and one with a ring topology over selected problems, averaged over 50 trials. The threshold of multimodality is shown as a dotted horizontal line at $dip \approx 0.0157$. These plots demonstrate the concept of the discovery phase, and support the assertion that a more limited communication topology will prolong the phase beyond that of a globally-connected swarm.

On all problems, both the global and the ring topologies start out in a unimodal distribution, as would be expected after their uniform initialization within the appropriate region. The distribution quickly becomes multimodal, as particles begin to explore the search space, attracting other particles when improved positions are discovered. This lasts only a short time for both topologies on the uni-

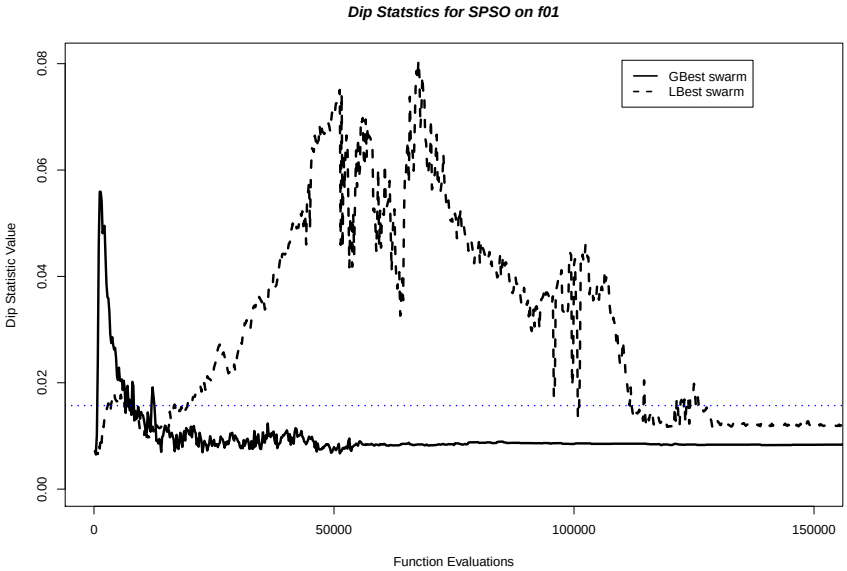


(a) f_1

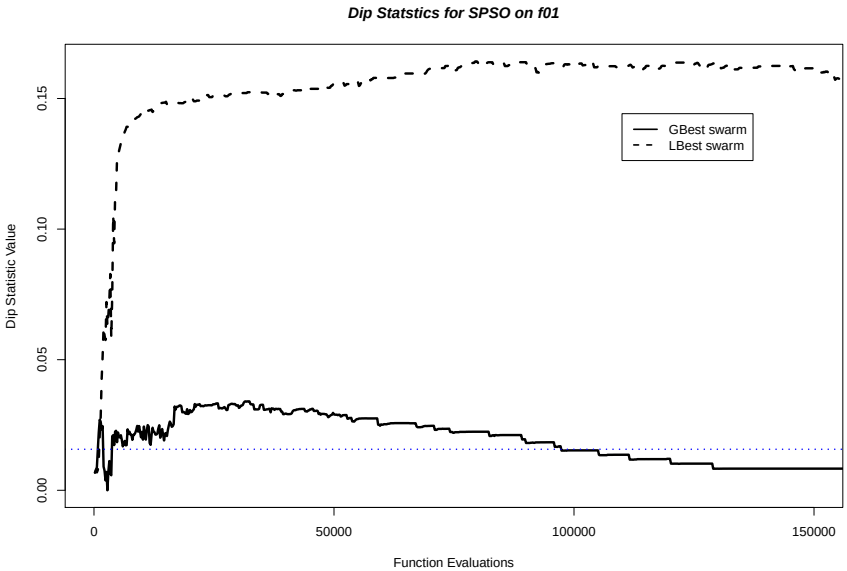


(b) f_5

Figure 3.5: Dip statistic values of swarms with global and ring topologies on problems f_1 and f_5



(a) f_8



(b) f_{12}

Figure 3.6: Dip statistic values of swarms with global and ring topologies on problems f_8 and f_12

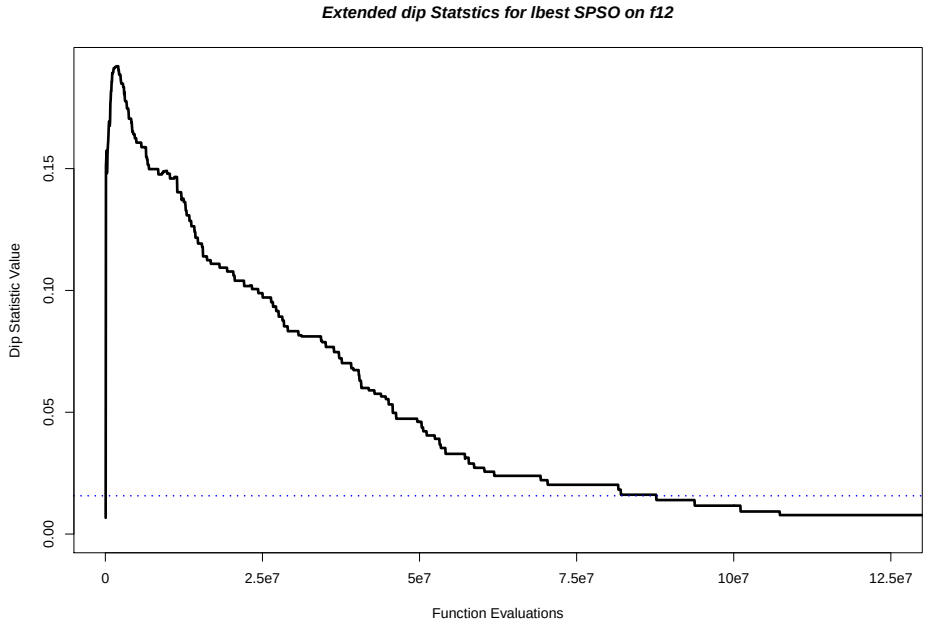


Figure 3.7: Dip statistic values for an lbest swarm on f_{12} until unimodality

modal f_1 landscape before the swarm again assumes a unimodal distribution, with all particles clustered around the same point at the fittest point of the optimum.

While the two swarms converge in much the same manner on the unimodal problems, the multimodal problems show the expected differences in behaviour. For these, the globally-connected swarm again expands quickly into the exploration phase, but contracts down to a unimodal distribution nearly as quickly, with all particles converged to the same point on the landscape. The distribution of the locally-connected swarm, however, maintains its multimodal distribution for a much longer time, indicating multiple groups of particles spread across multiple points. On several of the problems the swarm did not completely converge within the allotted 600000 function evaluations – this is expected if the swarm is unable to find the global optimum, as there may be multiple identical local optima that subswarms become split across. The Rastrigin problem, f_5, was one of these, where the dip statistic stayed above the threshold of modality for at least a tested $1E10$ function evaluations in all cases. Eventual distribution unimodality can be verified for problems with unique local optima given enough evaluations, e.g. as shown for f_{12} in figure 3.7.

Given the straightforward method of calculating and evaluating this measure over time, and its applicability to any optimization technique using a distributed population with definable "distances" between solutions, this dip statistic serves as a good description of the effect of communication on the behaviour of an algorithm.

3.4 Definition of Standard PSO

As defined above, the standard PSO algorithm proposed in this thesis is composed of:

- a local ring topology,
- the constricted update rules in equations 2.5 and 2.8,
- 50 particles,
- boundary conditions wherein a particle is not evaluated when it exits the feasible search space, and
- non-uniform swarm initialization *when used for testing purposes* (unnecessary for practical applications).

Using these components, we can expand the basic definition shown in the PSO algorithm on page 26 to the following "standard" form:

The Standard PSO algorithm

initialize constant $\chi = 0.72984$

initialize constants $c_1,\ c_2 = 2.05$

initialize swarm best found fitness $fg = maximum\ possible\ fitness$

for each particle i where $i = 1..50$ **do**

if performance testing **then**

initialize random particle position vector $\overrightarrow{x_i}$ within the search space using *region scaling*

else

initialize random particle position vector $\overrightarrow{x_i}$ within the search space

initialize particle best found position vector $\overrightarrow{p_i} = \overrightarrow{x_i}$

initialize particle velocity vector $\overrightarrow{v_i} = \overrightarrow{x_i}$

set particle left neighbor $nl_i = (i - 1)$ unless $i = 1$, in which case $nl_i = 50$

set particle right neighbor $nr_i = (i + 1)\ mod\ 50$

calculate particle current fitness $f_i = f(\overrightarrow{x_i})$

initialize particle best fitness $fp_i = f_i$

if particle fitness $f_i < fg$ **then**

set swarm best found fitness $fg = f_i$

set swarm best found position $\overrightarrow{pg} = \overrightarrow{x_i}$

for each step prior to termination criteria **do**

for each particle i where $i = 1..50$ **do**

initialize random cognitive weight vector $\overrightarrow{\epsilon_1}$ with independent elements in $U[0..1]$

initialize random social weight vector $\overrightarrow{\epsilon_2}$ with independent elements in $U[0..1]$

update particle velocity $\overrightarrow{v_i} = \chi\left(\overrightarrow{v_i} + c_1\epsilon_1 \bullet (\overrightarrow{p_i} - \overrightarrow{x_i}) + c_2\epsilon_2 \bullet (\overrightarrow{pg} - \overrightarrow{x_i})\right)$

update particle position $\overrightarrow{x_i} = \overrightarrow{x_i} + \overrightarrow{v_i}$

calculate particle current fitness $f_i = f(\overrightarrow{x_i})$

if particle current fitness $f_i < fp_i$ **then**

set particle best found fitness $fp_i = f_i$

if particle current fitness $f_i < fg$ **then**

set swarm best found fitness $sf = f_i$

set swarm best found position $\overrightarrow{pg} = \overrightarrow{x_i}$

best found fitness for the optimization is fg

best found position for the optimization is $\overrightarrow{pg}$

Since its original proposal, PSO has spawned a considerable amount of research into modifications and variations on the original algorithm, many of which have been shown to be significant performance

improvements on the original algorithm while still being generally applicable. Given the wide array of choice, several improvements which should have become commonly known and used have not been universally adopted.

To reiterate, this definition for PSO should not be considered as the "best possible option" for all or any problem sets. There are a huge number of variations and hybridizations of the original algorithm that potentially offer better performance on any sort of problem. What is offered here is both a standard for algorithm comparison and a standard to represent modern PSO in the global optimization community.

It is not the intent of this definition to discourage exploration and alterations to the PSO algorithm, but rather to give researchers a common grounding to work from. This baseline will provide a means of comparison for development of the particle swarm algorithm, and will prevent unnecessary effort being expended on "reinventing the wheel" on rigorously tested enhancements that are being used at the forefront of the field.

With our basis of comparison now well-defined, we can move on to analyzing and adapting the PSO algorithm into a form that offers significant improvements in both performance and simplicity. The algorithm defined here still carries aspects of the original biological simulation which it was designed to imitate[2] – next we will examine the applicability and usefulness of these properties to a purely mathematical optimizer, and reduce the algorithm to a simplified, better performing formulation.

3.5 Conclusions

This chapter has undertaken to define a standard for the field of particle swarm optimization. This standard selects a single approach or value for all of the variable aspects of the optimizer, each chosen for reasons of simplicity, or in cases where this is inapplicable, for superior performance. The exact algorithm is given in a form intended to be reproducible in every aspect by any interested party. Establishing such a standard ensures that forthcoming research is able to take advantage of beneficial modern techniques.

A robust benchmark was defined, informed by previous work in the field[35, 126]. This benchmark is intended for use for general performance testing and comparison of PSO-based algorithms, and can be used for any such optimizer functioning in a real-valued problem space. As such, it is defined in this chapter to facilitate further use in chapters 4 and 6 in their development of a simple, adaptive particle swarm.

The variable characteristics of the particle swarm optimizer were described as the swarm communication topology, the initialization and boundary conditions, and the number of particles in the swarm. A fixed configuration or value was selected for each of these based on previous findings in the literature and, in the case of the number of particles, work performed here.

Performance tests were carried out comparing the PSO algorithm as originally defined in [1, 2] to

this new standard in two forms – with a global, or gbest, topology where every particle is connected to every other particle, and with a ring, or lbest, topology where every particle connected to its left and right neighbors when the swarm is laid out in an open ring configuration. Both of these forms of the updated standard algorithm showed significantly improved performance over the original algorithm. Further, the ring topology formulation of the standard algorithm showed significantly superior performance over the global topology formulation across the benchmark.

Behaviour of the various formulations was also examined by way of Hartigan's dip test of unimodality[127], which was used as an indicator as to the current phase of the optimization process – exploration, discovery, or exploitation. The discovery phase was newly proposed in this work as the period between general exploration of the search space and exploitation of the final optimum. It is reached when the swarm has separated into multiple connected sub-swarms, each exploring a separate basin of attraction in the search space, and is indicated by a returned value from the dip test indicating multimodality within the distribution of the swarm..

With the performance improvements in mind, the new standard was formally defined. This definition encourages replication by interested parties by providing complete information about every aspect of the algorithm.

Chapter 4

A Simplified, Recombinant PSO Algorithm

4.1 Introduction

With a baseline algorithm now defined, we can turn our attention to examining the supposedly integral properties and behaviour of this standard formulation of PSO. A reasonable method of doing so is to deconstruct the algorithm by adjusting and/or removing components to see the effects on the optimizing ability and associated behaviour of the altered swarms.

To accomplish this, we will first investigate a variation to PSO that makes the major alteration of removing the multiplicative randomness of the algorithm in the context of adapting it for other uses. Originally conceived as a modification to the standard algorithm for use on self-reconfigurable adaptive systems used in on-chip hardware processes[82], PSO with discrete recombination (PSO-DR) introduces several appealing and effective modifications to the standard algorithm. It is one of the more interesting advances in PSO research over the last few years because these modifications apparently do not degrade performance while removing various issues associated with the stochasticity of the PSO acceleration parameters that hinder theoretical analysis of the algorithm.

The physical creation of hardware-based optimizers is a rather more intricate undertaking than software implementations, so fast, simple algorithms are desirable in order to minimize complexity. The comparative straightforwardness of PSO to many other evolutionary optimization algorithms makes it a good choice for this purpose, and the further simplifications in the creation of a particle swarm relying on discrete recombination operators reduce the complexity still more. The resulting algorithm, which can be implemented using only addition and subtraction operators and a simple 1-bit random number generator, is well-suited for dedicated hardware settings.

Despite this rather specific original design specification, PSO-DR has shown to be a robust optimizer in its own right, equaling or surpassing the standard PSO implementation from the previous chapter on a few tested benchmarks [82]. Here the original proposal of the PSO-DR concept is expanded upon by means of alternative topologies and parameter settings, a more comprehensive test suite, sub-

jecting the model to a burst analysis, and deriving simplified variants of the algorithm. Specifically, this chapter introduces PSO-DR (known here as Model 1) as originally defined, and discusses the velocity burst behaviour of particle swarm algorithms[129]. Later sections introduce a series of simplifications to PSO-DR (Models 2 and 3), down to a form that retains the competitive optimizing speed and performance of the standard algorithm while dropping components of the update equations that are shown to be extraneous to effective use.

Once these new modifications to the algorithm are defined, the empirical performance of the new models is demonstrated alongside that of a standard PSO algorithm. Following this is an empirical investigation of bursting patterns in recombinant PSO and the implications of these patterns in the light of their effect on the standard PSO algorithm.

4.2 PSO with Discrete Recombination

The original proposal of PSO-DR worked off of the PSO formulation implementing inertia weight, rather than constriction. While the constricted formulation of PSO has undergone more rigorous analysis, the two formulations are algebraically identical, and much of the literature uses the widespread *IW* formulation for means of extension. As defined previously, the velocity and positional updates for particle i in standard PSO (SPSO) in the inertia weight formalism are:

$$v_{id} = wv_{id} + c\epsilon_1(p_{id} - x_{id}) + c\epsilon_2(p_{nd} - x_{id}) \tag{4.1}$$

$$x_{id} = v_{id} + x_{id} \tag{4.2}$$

where d labels components of the position and velocity vectors, $d = 1, 2, \ldots D$, $\vec{p}_i$ is the personal best position achieved by i, $\vec{p}_n$ is the best position of informers in i's social neighborhood and $\epsilon_{1,2} = U(0, 1)$[130].

PSO-DR introduces a recombinant version of PSO by replacing the personal best position $\vec{p}_i$ by a position formed using a recombinant. In the original PSO-DR formulation this recombinant position vector $\vec{r}$ is defined by:

$$r_{id} = \eta_d p_{ld} + (1 - \eta_d)p_{rd} \tag{4.3}$$

where $\eta_d = U\{0, 1\}$ and $\vec{p}_{l,r}$ are immediate left and right neighbors of i in a ring topology. Note that while separate random numbers η_d are used for separate dimensions d, a single value is generated for each individual dimension and used for both occurrences of η_d in that dimension. This places $\vec{r}_i$ at a corner of the smallest D-dimensional box which has p_l and p_r at its corners.

The velocity update for the original form of PSO-DR is:

$$v_{id} = w\, v_{id} + \frac{\phi}{2}(r_{id} - x_{id}) + \frac{\phi}{2}(p_{nd} - x_{id}) \tag{4.4}$$

where $\frac{\phi}{2}$ is merely the authors' rendition of c. As PSO-DR was designed to be as efficient an implementation as possible, it was argued that the removal of the random numbers $\epsilon_{1,2}$ from equation (4.1) could be mitigated by setting the parameter $\phi = 2$.

The choice of ϕ was based on the observation that $c \approx 2.0$ (i.e. $\phi \approx 4.0$) in the standard PSO algorithm, and because $\epsilon_{1,2}$ are uniform in $[0, 1]$, the expectation value of $\phi\epsilon_{1,2}$ is 2.0. Furthermore, by setting $w = 0.5$, the weighting of the previous velocity term can be implemented in hardware by a right shift operation.

Whilst optimal efficiency is desirable for hardware implementations, this issue does not concern us to the same degree in the following research into PSO-DR – it is, in fact, one aim herein to study the algorithm for arbitrary parameter values.

Although equation 4.4 contains a random element η in the recombinant position, the acceleration parameters $\frac{\phi}{2}$ for both terms are constant. This means that while SPSO depends upon multiplicative stochasticity in its velocity updates, PSO-DR has additive stochasticity [129, 131]. This has two ramifications: first, a stability condition can be computed based on the theory of second order, fixed parameter, difference equations and second, due to the removal of multiplicative stochasticity, recombinant PSO is predicted not to exhibit the particle velocity bursts seen in the baseline algorithm.

The stability condition is:

$$\begin{cases} |w| < 1 \\ 0 < \phi < 2(1 + w) \end{cases} \tag{4.5}$$

Details of the determination of this condition can be found in [129] and [131], as well as the chapter 5 of this work, along with a thorough examination of the statistical properties of PSO-DR.

4.3 Simplifying Recombinant PSO

While PSO-DR already involves a simplification to the standard algorithm by means of the removal of multiplicative stochasticity, further potential simplifications are apparent. The technique of "sweeping" through a multi-dimensional parameter space to find optimal combinations was originally applied to the standard PSO in a prior work focused on deriving a simpler form of that algorithm[76]. While the simplified form of the algorithm detailed in that publication showed promising performance improvements on some types of problems, across the board it could not match the canonical algorithm, and was developed mainly to allow for experimentation with a new approach to statistical analysis. That work became a precursor to a robust method of analyzing a non-stochastic PSO algorithm, which is explained and used

in the next chapter to deconstruct PSO-DR.

4.3.1 PSO-DR Model 2

With this means of evaluating the relative effects of different parameter combinations available for the standard algorithm, adapting it for use on the discrete recombinant formulation is a simple matter. Whereas the previous sweeps explored the results of multiple combinations of the χ and ϕ of the standard algorithm, here we can use the method to first find an optimal balance between the inertia weight coefficient w and the ϕ coefficients.

This first sweep, shown in figure 4.1 for selected functions, reveals that while the optimal region of combinations is spread across the parameter space, it also intersects the axis for the w term. This demonstrates that system is able to obtain good performance results even at $w = 0.0$, i.e. when there is no previous velocity term in the update equations. This is reminiscent of the very first formulation of PSO[2], discussed in the previous chapter, which also lacked a previous velocity term – that algorithm, however, proved to be less effective than desired, leading to the development of the constricted and the inertia weight formulations.

PSO-DR Model 2 therefore sets $w = 0.0$, resulting in a simplified two-term velocity update:

$$v_{id} = \frac{\phi}{2}(r_{id} - x_{id}) + \frac{\phi}{2}(p_{nd} - x_{id}) \tag{4.6}$$

As the velocity and positional update equations are both first-order in this representation, velocity now serves as a dummy variable, and Model 2 can be represented as a single, velocity-free, first-order update equation:

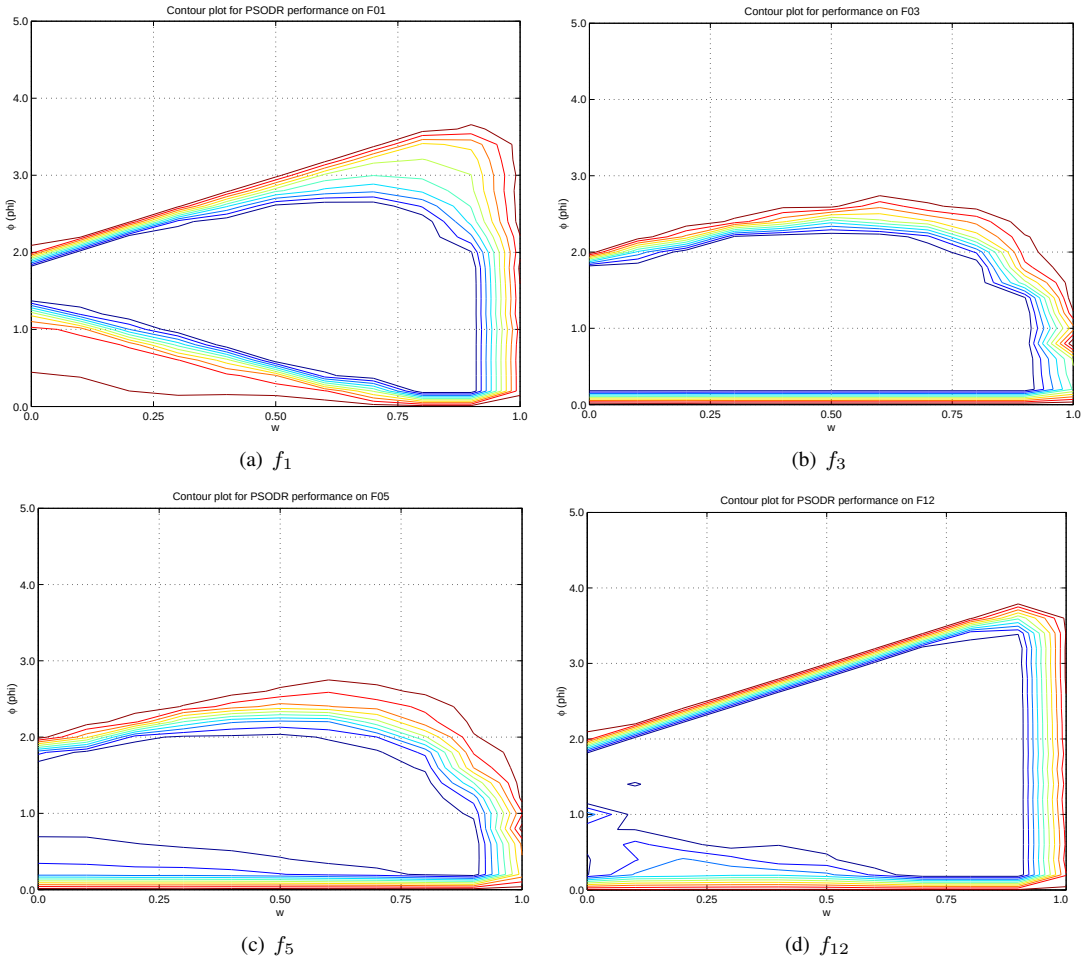
$$x_{id} = x_{id} + \frac{\phi}{2}(r_{id} - x_{id}) + \frac{\phi}{2}(p_{nd} - x_{id}) \tag{4.7}$$

4.3.2 PSO-DR Model 3

While Model 2 now only has a single parameter, ϕ, in its update equation, the multiple uses of that parameter allow for another examination of the effects of the possible combinations. Using the same approach, the two ϕ terms of PSO-DR Model 2 were next separated into ϕ_1 and ϕ_2, and another sweep through parameter space was performed. By doing this we can find the optimal set of combinations of the recombinant component, via its coefficient ϕ_1, and the neighborhood best component, via its coefficient ϕ_2.

Surprisingly, results again showed that the optimal region intersects an axis, this time for the neighborhood term $(p_{nd} - x_{id})$ (see figure 4.2 for plots on unimodal and multimodal problems).

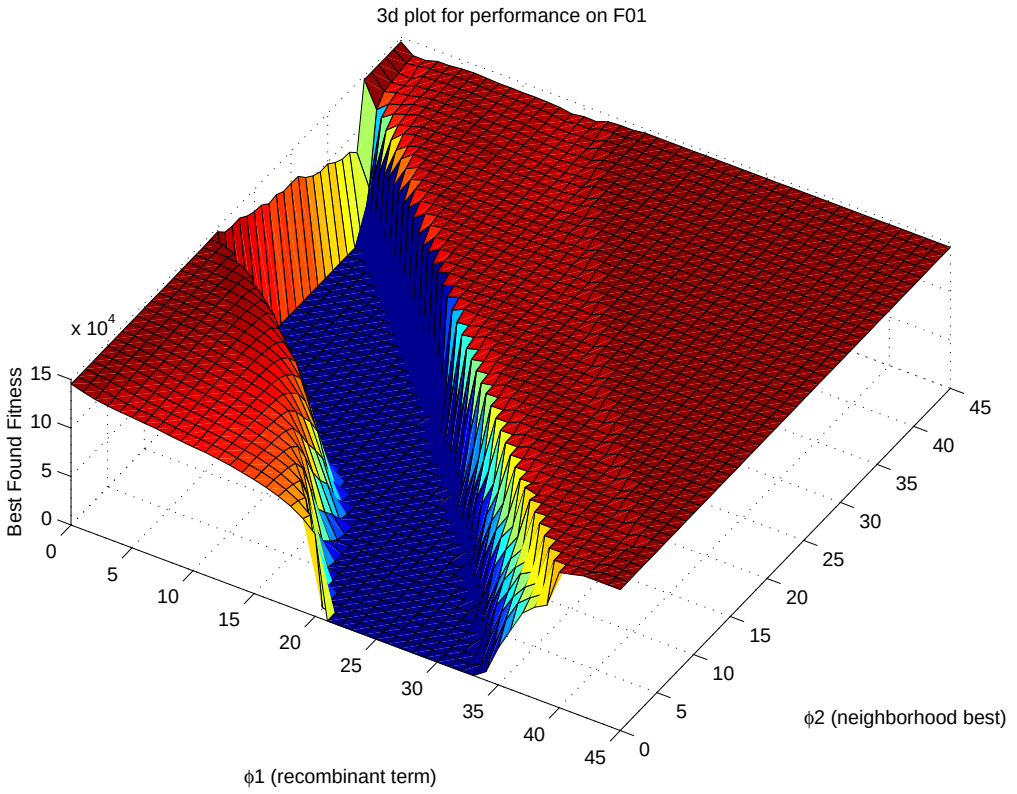
The knowledge that the algorithm is stable and able to return good performance even when the ϕ_2 multiplier is set to 0.0, effectively canceling this term and removing the influence of the neighborhood



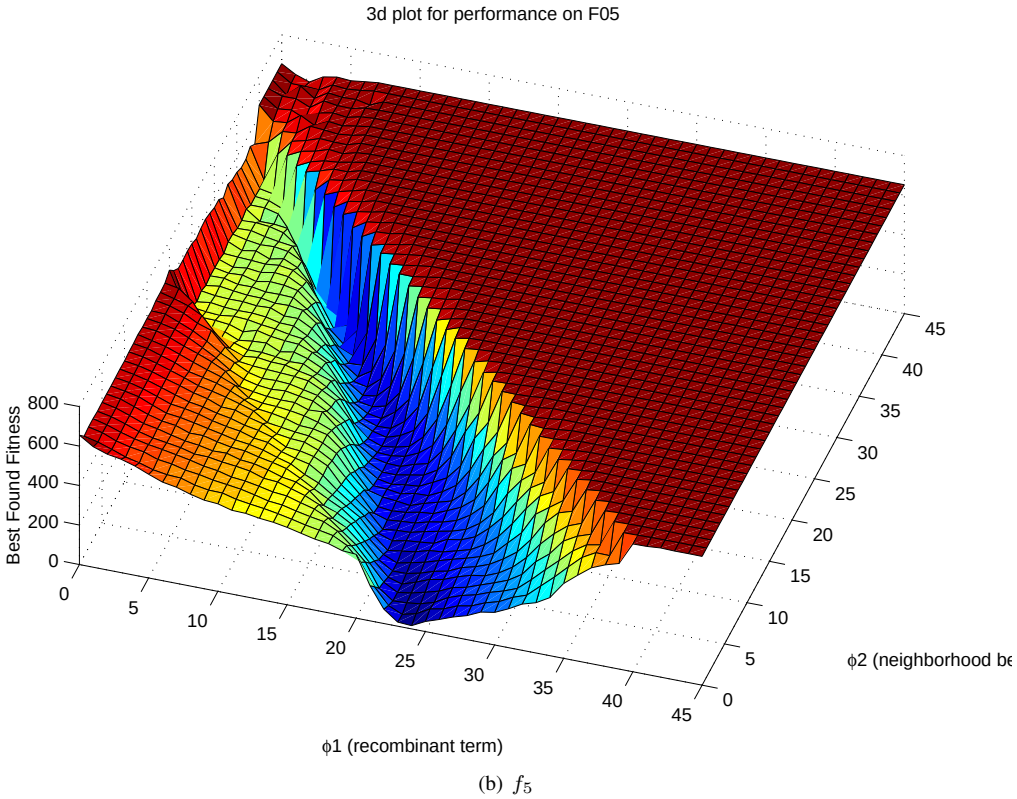
(a) f_1 (b) f_3

(c) f_5 (d) f_{12}

Figure 4.1: Optimal regions for combinations of w and ϕ in PSO-DR Model 1. Each contour line represents a 10% improvement in performance from worst (red) to best (blue).



(a) f_1



(b) f_5

Figure 4.2: Optimal regions for combinations of ϕ_1 and ϕ_2 in PSO-DR Model 2 from worst (red) to best (blue).

best position, allows a further simplification to the update equation (4.4), down to PSO-DR Model 3:

$$x_{id} = x_{id} + \phi(r_{id} - x_{id}) \tag{4.8}$$

This is clearly a substantial reduction of the original PSO-DR equation. The "velocity" component has been reduced from three terms in the originally-proposed PSO-DR Model 1, to two terms in Model 2, to a single social term that takes into account the entire neighborhood of the particle, $\phi(r_i - x_i)$, in Model 3.

This PSO variant, if shown to be viable and competitive, would raise some interesting questions. To what extent is velocity a necessary component of PSO – is it a relic of the biological origins of the algorithm, as has been suggested in the past[79] and is the case for the *vmax* term? How important is it that the neighborhood component be drawn from the single best neighbor?

The optimization process of Model 3, hereafter referred to interchangeably as a discrete recombinant swarm (DRS), is driven entirely by the recombinant component; this idea is reminiscent of fully informed particle swarms (FIPS) [81], where the entire neighborhood influences particle behaviour. Here, rather than every neighbor influencing behaviour in every dimension as in FIPS, a single randomly chosen neighbor fully influences the particle in each dimension. This gives the particle an updated position that is a combination of the best positions of all of its neighbors throughout all dimensions.

4.4 Performance

The various algorithms were tested over the series of 14 benchmark functions defined in tables 3.1 and 3.2 of the previous chapter. Particles were again initialized using the region scaling technique. As before, in instances where the global optimum was located at the center of the search space (i.e. $f_1, f_2, f_5 - f_7$), the function was *shifted* by a random vector with maximum magnitude of a tenth of the size of the search space in each dimension for each run to remove any chance of a centrist bias[120].

Problem	**Optimal ϕ Value**	**Problem**	**Optimal ϕ Value**
f_1	1.05	f_8	1.20
f_2	1.10	f_9	1.20
f_3	1.15	f_{10}	1.10
f_4	1.30	f_{11}	1.35
f_5	1.20	f_{12}	1.20
f_6	1.20	f_{13}	1.35
f_7	1.15	f_{14}	1.35

Table 4.1: "Optimal" values for ϕ using PSO-DR Model 3 with fixed parameters $NP = 50$ and $K = 2$

This investigation tested all three models of PSO-DR using both a global (as used in the originally

		Model 1 *Ring*	**Model 1** *Global*	**Model 2** *Ring*	**Model 2** *Global*	**Model 3** *Ring*	**Model 3** *Global*
f_1	Success	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals	61529±124	32565±106	35913±89	**16208±72**	76748±847	109708±348
f_2	Success	0%	**100%**	**100%**	**100%**	0%	0%
	Best	5.02E-10	**0.0**	**0.0**	**0.0**	3.61E-5	199.5
	Mean	7.5E-9±1.1E-9	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	3.4E-3±1.2E-3	1025±75
	Worst	3.61E-8	**0.0**	**0.0**	**0.0**	5.14E-2	2453
	FEvals	-	167655±990	528262±3224	**119241±1103**	-	-
f_3	Success	0%	0%	0%	**78%**	0%	0%
	Best	2.86E-1	5.07E-13	2.24E-3	**0.0**	2.72E-5	8.73E-3
	Mean	11.12±0.72	0.957±0.243	4.39±1.44	**0.877±0.236**	8.48±1.18	6.16±0.778
	Worst	20.4	**3.99**	69.61	**3.99**	33.1	18.96
	FEvals	-	-	-	**272574±2874**	-	-

Table 4.2: Results for PSO-DR models on unimodal problems

proposed algorithm) and a local ring topology for selecting the neighborhood operator used in both the recombinant and the neighborhood components. The parameter settings $w = 0.5$ and $\phi = 2$ are explained above, giving a velocity update with the form:

$$v_{id}^{t+1} = 0.5v_{id}^{t} + (r_{id} - x_{id}^{t}) + (p_{nd} - x_{id}^{t}) \tag{4.9}$$

Results shown for PSO-DR Model 2 use the value $\phi = 1.6$, while those for PSO-DR Model 3 use $\phi = 1.2$. As with determining the "best" number of particles in a swarm, selecting these values was done through an empirical examination of performance on the benchmark problems for multiple settings with each algorithm. Table 4.1 shows these best-performing values of ϕ for Model 3 on each problem – the mean of these is 1.21, the median and mode both 1.2, hence its selection as the default setting.

For comparison, results were tested for significance against the lbest standard PSO algorithm from the previous chapter, which operates using the constricted velocity update, equation 2.8, with the recommended $\phi = 4.1, \chi = 0.72984$ and with 50 particles. As explained in section 3.2.3, determining the best single choice for the number of particles in a swarm is a complex undertaking, with multiple tradeoffs between performance on various types of landscapes. Because of this, all PSO-DR tests were also carried out using 50 particles in order to be in line with SPSO. Algorithm performance was again measured as the minimum error $|f(x) - f(x^{\star})|$ found over the trial where $f(x^{\star})$ is the fitness at the global optimum for the problem. Results were averaged over 50 independent trials, and are displayed, with standard error, in tables 4.2 – 4.4. Values less than 10^{-15} have been rounded to 0.0.

These performance results for all models of PSO-DR compared to those seen for SPSO in tables 3.3 – 3.5 clearly indicate that it is a competitive variant. While the various PSO-DR models using a global topology are generally inferior to the ring-topology SPSO algorithm, they do compare well across the entire set of benchmarks against the same when used with a global topology. Those PSO-DR models using a ring topology perform extremely well against both ring- and global-topology SPSOs. Statistical tests were performed on these results to determine the significance of the performance differences

		Model 1 *Ring*	**Model 1** *Global*	**Model 2** *Ring*	**Model 2** *Global*	**Model 3** *Ring*	**Model 3** *Global*
f_4	Success	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	2105	3199	2961	3553	**829**	2961
	Mean	2645±34	3697±32	3357±21	3994±43	**1576±39**	3366±18
	Worst	3079	4132	3553	4753	**2013**	3553
	FEvals	-	-	-	-	-	-
f_5	Success	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	7.01	58.7	22.88	147.2	**2.98**	10.94
	Mean	23.85±1.57	117.92±5.27	45.02±1.94	263.4±9.3	**9.19±0.64**	17.69±0.63
	Worst	50.21	201.0	73.63	394.0	**21.89**	27.86
	FEvals	-	-	-	-	-	-
f_6	Success	**100%**	8%	2%	0%	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	19.65	**0.0**	**0.0**
	Mean	**0.0±0.0**	18.14±0.76	19.7±0.4	19.8±0.006	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	20.20	20.38	19.87	**0.0**	**0.0**
	FEvals	129505±3503	**58241±2245**	466163±0	-	226790±9485	191363±711
f_7	Success	**100%**	54%	98%	40%	94%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	6.7E-3±1.3E-3	1.5E-4±1.5E-4	1.3E-2±2.0E-3	4.4E-4±2.5E-4	8.9E-4±3.4E-4
	Worst	**0.0**	3.68E-2	7.4E-3	6.11E-2	7.4E-3	7.4E-3
	FEvals	99433±8130	33196±131	86834±12383	**16434±102**	70276±704	111643±392
f_8	Success	**100%**	70%	96%	42%	96%	96%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	8.1E-2±2.5E-2	1.0E-2±8.5E-3	0.709±0.202	4.2E-3±2.9E-3	4.1E-3±2.9E-3
	Worst	**0.0**	8.32E-1	4.15E-1	8.75	1.04E-1	1.04E-1
	FEvals	63281±223	32065±336	42500±382	**21576±686**	95725±630	168749±2502
f_9	Success	**100%**	86%	**100%**	36%	98%	86%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	6.6E-3±4.0E-3	**0.0±0.0**	0.552±0.166	2.2E-4±2.2E-4	1.5E-3±5.5E-4
	Worst	**0.0**	1.76E-1	**0.0**	4.5	1.1E-2	1.1E-2
	FEvals	64659±157	31877±122	39363±182	**20306±1058**	91635±454	169829±2347

Table 4.3: Results for PSO-DR models on complex multimodal problems

between the two algorithms.

Full results of the Bonferroni-adjusted significance tests are displayed in table 4.6. The comparisons shown are measured against the ring-topology standard constricted PSO. As expected, the global-topology PSO-DR models were significantly poorer on many problems (though better for a distinct few), but the performance of the ring-topology models is much better.

For the original Model 1 formulation using a ring topology, significant improvements over the canonical PSO are seen for four of the fourteen problems, and significantly poorer performance on one. The gains are especially beneficial as they come mainly on the highly complex multimodal problem such as f_5 (Rastrigin) while maintaining equal performance on the rest of the multimodals and the majority of unimodals.

Model 2, when using a ring topology, fared slightly poorer, with significantly improved performance on two problems and significantly decreased on two. Opposite to Model 1, the two significantly poorer problems were both highly complex multimodals – while improvements are possible, the standard PSO algorithm is still highly capable on these types of problems, especially when using an lbest topology. For the global-topology version of Model 2, performance is generally poor, but looking deeper than that, we can see that this formulation shows remarkable speed on the trials when it does manage to find a global optimum. It is the fastest algorithm of all of the PSO-DR models on ten of the eleven problems where it found the optimum, usually by quite a substantial margin, and is faster by far than the standard PSOs in every case.

Detailed results for the statistical tests on PSO-DR Model 3 and SPSO, both with ring topologies,

		Model 1 *Ring*	**Model 1** *Global*	**Model 2** *Ring*	**Model 2** *Global*	**Model 3** *Ring*	**Model 3** *Global*
f_{10}	Success	94%	**100%**	94%	**100%**	98%	98%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	7.2E-16±4.1E-16	**0.0±0.0**	9.6E-16±5.6E-16	**0.0±0.0**	1.6E-2±1.6E-2	2.4E-16±2.3E-16
	Worst	1.31E-14	**0.0**	2.11E-14	**0.0**	0.816	1.13E-14
	FEvals	128848±17573	28751±2974	124065±16799	21759±3599	**5622±122**	41764±9798
f_{11}	Success	**100%**	98%	90%	40%	98%	36%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	1.62±1.62	8.1±3.47	48.6±5.67	1.62±1.62	51.84±5.55
	Worst	**0.0**	81.0	81.0	81.0	81.0	81.0
	FEvals	4909±38	3940±24	5063±154	**3012±27**	6198±134	8288±1531
f_{12}	Success	**98%**	38%	88%	36%	**98%**	42%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.101±0.101**	4.01±0.47	0.703±0.278	4.46±0.49	0.149±0.149	4.31±0.52
	Worst	**5.05**	7.52	7.47	7.52	7.47	7.52
	FEvals	95865±9364	7740±1123	40758±6810	**4297±32**	16839±1415	6771±85
f_{13}	Success	**100%**	60%	94%	58%	98%	38%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	2.52±0.453	0.401±0.227	3.06±0.52	0.134±0.134	4.2±0.475
	Worst	**0.0**	7.65	6.68	8.57	6.68	7.65
	FEvals	28886±3560	24769±18169	36830±11070	**4243±23**	31861±2111	16100±9307
f_{14}	Success	**100%**	74%	98%	46%	**100%**	48%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean	**0.0±0.0**	1.82±0.44	0.134±0.134	3.85±0.514	**0.0±0.0**	3.43±0.482
	Worst	**0.0**	8.11	6.7	8.11	**0.0**	8.11
	FEvals	26915±2819	8212±1058	24356±2913	**4326±29**	25119±1795	31282±19768

Table 4.4: Results for PSO-DR models on simple multimodal problems

	M1 Ring	**M1 Global**	**M2 Ring**	**M2 Global**	**M3 Ring**	**M3 Global**
f_1	*	*	*	*	*	*
f_2	+	+	+	+	*	–
f_3	–	+	*	+	–	–
f_4	+	–	–	–	+	–
f_5	+	+	+	–	+	+
f_6	+	–	–	–	+	+
f_7	*	–	*	–	*	*
f_8	*	–	*	–	*	*
f_9	*	*	*	–	*	–
f_{10}	*	*	*	*	*	*
f_{11}	*	*	*	–	*	–
f_{12}	*	–	*	–	*	–
f_{13}	*	–	*	–	*	–
f_{14}	*	*	*	–	*	–

Table 4.5: Significance of results for *PSO-DR models* vs *ring-topology SPSO* where + = better, * = equivalent, – = worse

are shown in table 4.5 and confirm that performance is significantly improved on 3 of the 14 tested functions (f_4-f_6), equivalent for 10 functions, and significantly worse for 1 function (f_3) for PSO-DR Model 3 vs SPSO with ring topology. This is nearly as good as Model 1, and even more impressive given the replacement of multiple integral components of the algorithm with a single recombinant operator. Given the superior performance of Model 3 combined with its simplified nature, closer examination of its performance and behaviour is warranted. From this point any references to either the SPSO or the PSO-DR Model 3 algorithms will be speaking of the ring-topology construction of the swarm.

Due to the high number of function evaluations that were performed to obtain these results relative to previous work (where only 30k-60k function evaluations might be performed), selected convergence plots are shown in figure 4.3. These show that SPSO obtains superior results at the very start of the optimization process, up to approximately 5000 function evaluations across all functions (1000–3000 in

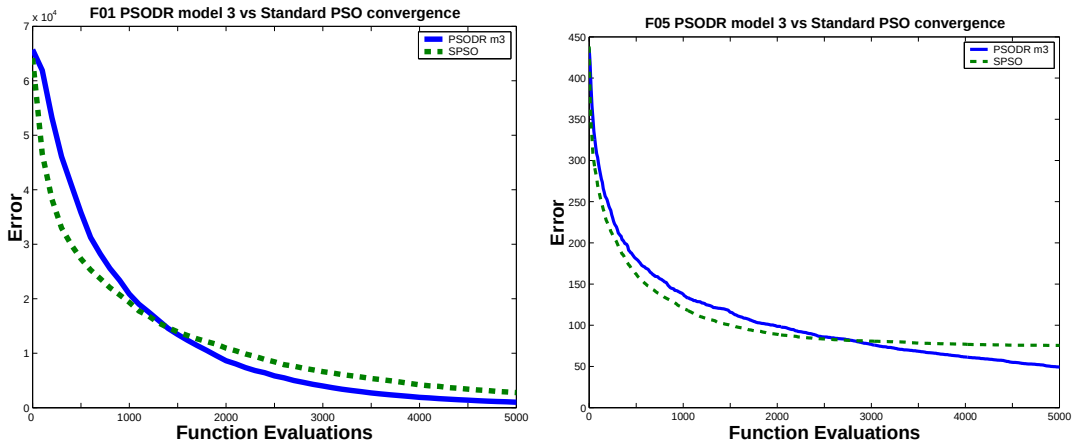
Func	p-value	Inverse rank	α'	Significant
f_4	$<$ 2.2E-16	13	0.003846	Yes
f_5	$<$ 2.2E-16	12	0.004167	Yes
f_6	$<$ 2.2E-16	11	0.005	Yes
f_3	4.335E-05	10	0.004545	Yes
f_2	0.007728	9	0.005556	No
f_{14}	0.02406	8	0.00625	No
f_{13}	0.05285	7	0.007143	No
f_{12}	0.05922	6	0.008333	No
f_8	0.1594	5	0.01	No
f_7	0.3129	4	0.0125	No
f_9	0.3222	3	0.016667	No
f_{10}	0.3222	2	0.025	No
f_{11}	0.3222	1	0.05	No
f_1	=	-	-	No

Table 4.6: Detailed significance findings for PSO-DR Model 3 vs ring-topology SPSO

the two displayed). After this point, PSO-DR Model 3 surpasses the standard algorithm in performance, and maintains this advantage to the end of the trials on six of the fourteen tested problems ($f_4 - f_6, f_{12} - f_{14}$). On problems for which both algorithms were able to attain error levels of 0.0 in at least one run ($f_1, f_6 - f_{14}$), the point at which this occurs, i.e. when SPSO "catches up" to PSO-DR Model 3, can be taken from the number of function evaluations required in successful runs. For example, on f_7 both algorithms were able to find the global optimum on numerous trials, within an average of approximately 125k function evaluations for SPSO, and an average of approximately 70k evaluations for PSO-DR Model 3. Across these 10 problems, PSO-DR Model 3 on average found the global optimum faster than SPSO in 8 cases, 6 of these significant.

Finally, for the problems on which the SPSO outperformed PSO-DR Model 3, the same early performance is seen with PSO-DR Model 3 surpassing SPSO in performance early in the optimization process; in these cases, SPSO eventually repasses the other algorithm before the end of the 600k function evaluations.

A potential explanation for this behaviour lies in the diversity of the swarms at this point in the optimization process. Figure 4.4 shows the mean Euclidean distance between particles for the corresponding convergence plots of figure 4.3. It should be noted that uniform initialization was used in the trials used to generate these plots; relative performance between the algorithms was unaffected, and initializing particle positions uniformly throughout the search space removes an unrelated phenomenon in subspace initialization wherein the swarm expands greatly beyond the relatively small initialization region at the start of the optimization process to explore the search space. Expansion is common in the



(a) f_1 (b) f_5

Figure 4.3: Convergence plots for SPSO and PSO-DR Model 3 early in the optimization process

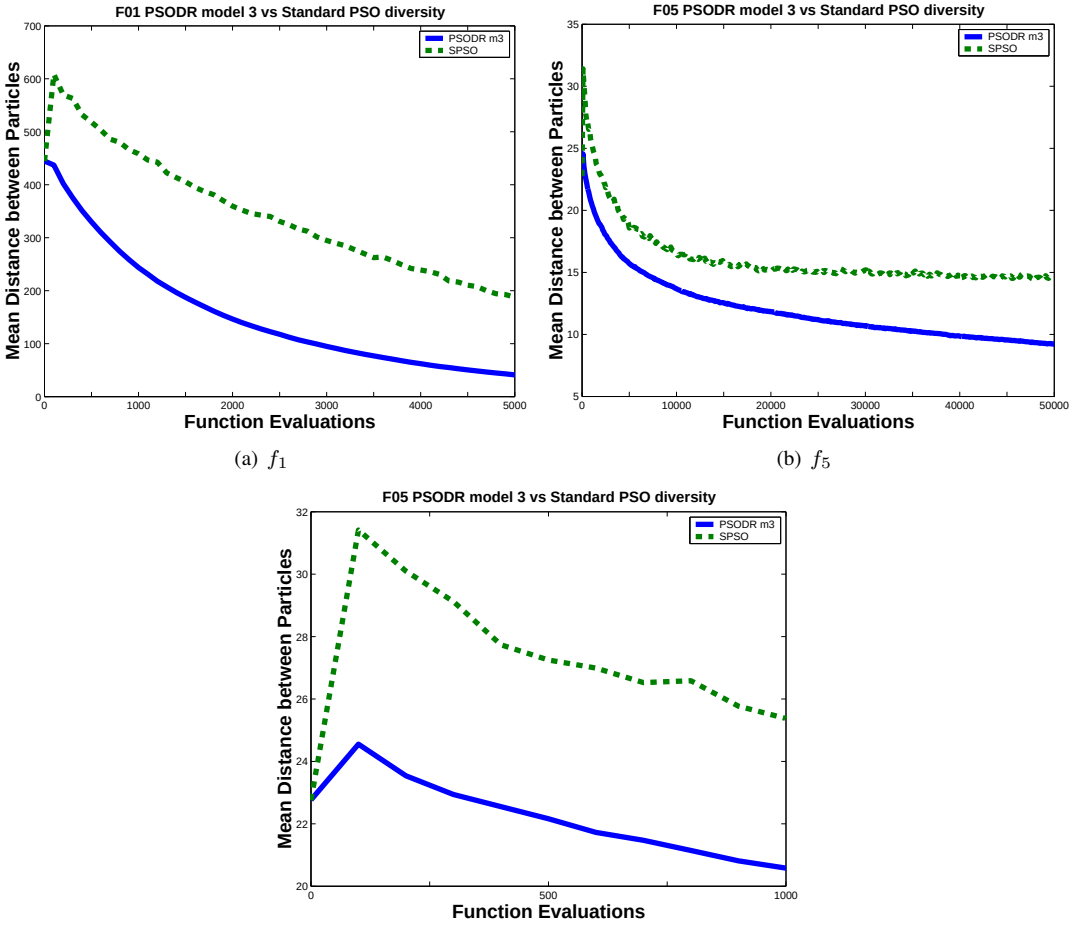
first few iterations using uniform initialization as well, but this is inherent to the swarm behaviour and influenced only by the topography of the problems landscape and the size of the search space.

As can be seen in the plots of figure 4.4, neither swarm type begins converging immediately following initialization but rather maintain their diversity or expand slightly. On a comparative basis, the standard PSO swarm expands substantially more than the PSO-DR Model 3 swarm; for example figure 4.4c shows that after the first approximately 100 function evaluations the mean real-valued distance between particles in the standard PSO swarm increases from 23 to 31.5 on a search space of size 10.24^{10}, while the PSO-DR swarm diversity increases only from 23 to 24.5. Similar disparities were observed for all other tested problems.

It is reasonable to gather from these results that the higher swarm diversity for the standard PSO algorithm early in the optimization process demonstrates a wider spread of particle dispersion, and hence an expanded search for the basin of attraction of global or local optima. PSO-DR Model 3 expands very little early in the optimization process, suggesting that its first phase of exploration is shorter than that of the standard algorithm.

4.4.1 Dip Statistics

The modality of the distribution of the best-found positions of each particle for the various PSO-DR models is generally comparable to that of the standard PSO, with a few exceptions. In cases where the swarm settles on multiple local optima with equal fitness ratings, the same behaviour arises, with multiple subswarms forming and leading to permanent multimodality in the distribution, represented by a dip statistic value that is higher than the threshold defined for the size of the swarm. As the same benchmark was used for both the empirical tests in the previous chapter and the ones presented here, this is seen on the same problems – like SPSO, once a PSO-DR particle has set a personal best position, the only way for this to be changed is through discovery of an improved optimum. If all other particles are



(a) f_1

(b) f_5

(c) $f_5(zoomed)$

Figure 4.4: Diversity plots for SPSO and PSO-DR Model 3 early in the optimization process

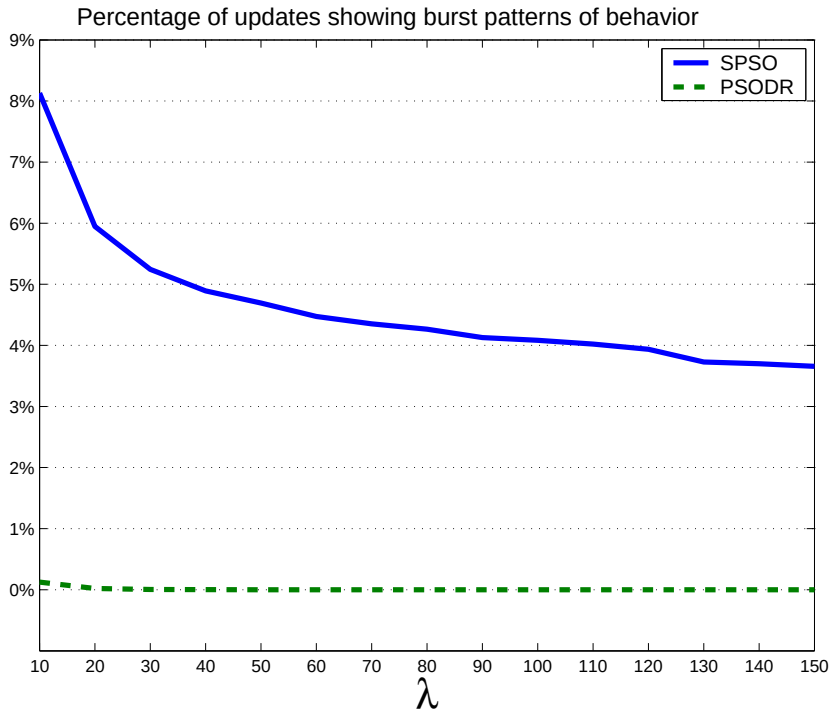


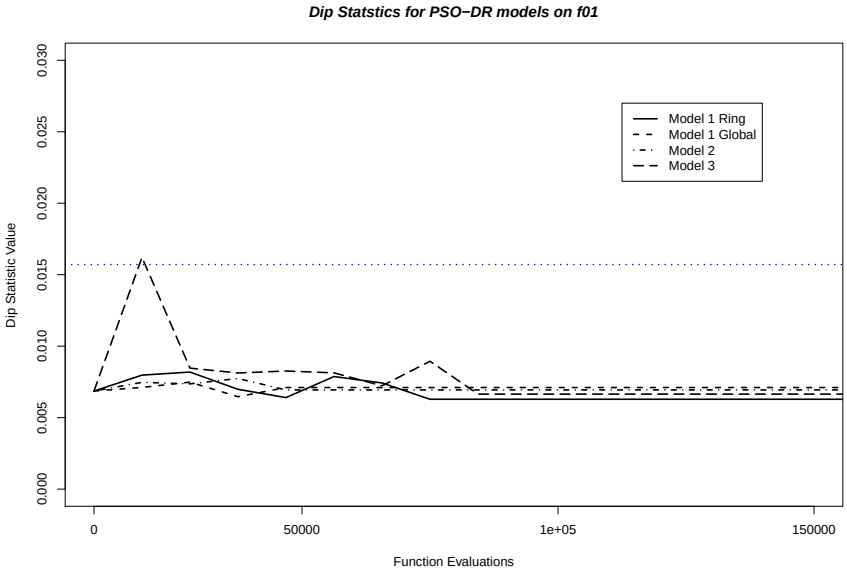
Figure 4.5: Frequency of updates showing burst behaviour for values of λ

converged to equal or inferior optima, the swarm will remain permanently divided. This is the case even with Model 3; despite its update equation lacking the personal best term, it uses those of its neighbors, which obey the same rules.

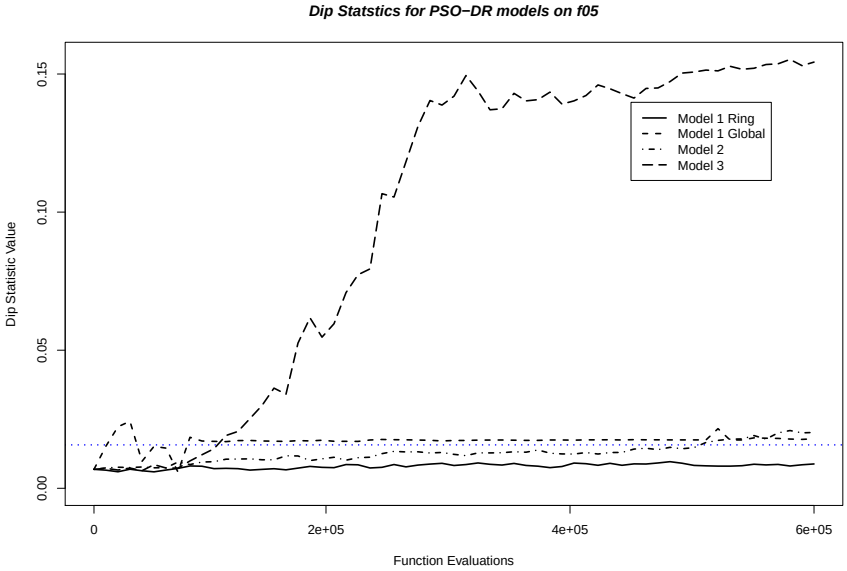
Plots of the dip statistic values over the same problems demonstrated in figures 3.5 and 3.6 in chapter 3 provide a comparison of the behaviour of the algorithms. Ring topology results are shown in figures 4.6 and 4.7 for all three models of PSO-DR – results for the global topologies were obtained, but are shown only for Model 1 due to the extreme similarity in appearance between all three models.

For problems with multiple equal local minima, exemplified in the displayed plots by f_5, average behaviour of all three over 50 trials results in a multimodal swarm distribution apart from in the case of the ring-topology Model 1. The ring-topology PSO-DR Model 3 swarm especially becomes progressively more and more multimodal as subswarms converge down to multiple distinct optima. On f_{12}, with its low dimensionality and multiple distinct optima, behaviour again mimics that of SPSO, with all three models of swarms quickly dividing into a multimodal distribution before eventually settling down to a single optimum after a very large number of function evaluations. In this case, the globally-connected swarms never attained multimodality, instead converging to a single optimum quickly and reliably.

The notable difference in behaviour to SPSO comes on problems f_1–f_3, those with a single optimum. This is demonstrated using f_1 in figure 4.6a. Unlike SPSO swarms, which quickly attain multimodality and nearly as quickly again converge back down to unimodality, PSO-DR swarms maintain a

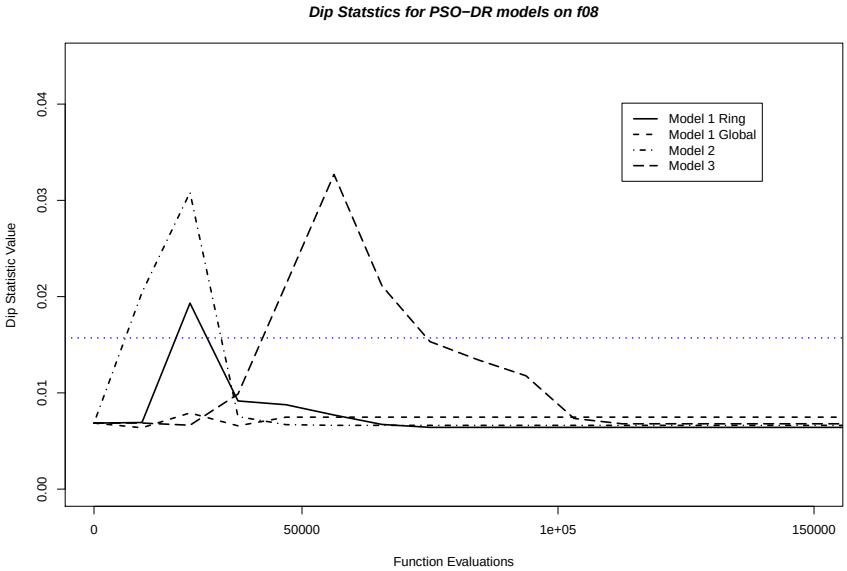


(a) f_1

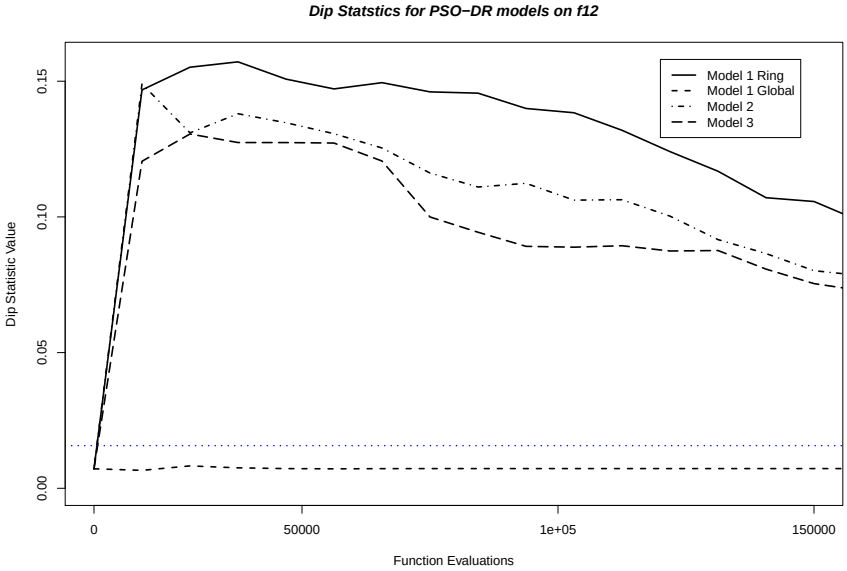


(b) f_5

Figure 4.6: Dip statistic values of PSO-DR swarms on f_1 and f_5



(a) f_8



(b) f_{12}

Figure 4.7: Dip statistic values of PSO-DR swarms on f_8 and f_12

unimodal distribution throughout the entire optimization, apart from an exceedingly brief period where Model 3 barely crests into multimodality near the beginning. This helps to explain the nearly universally faster performance of the PSO-DR algorithms on these problems – these swarms use almost no function evaluations on dividing into subswarms to exploit multiple points, instead converging to the single global optimum from the very start of the optimization.

4.5 Velocity Bursts

It has been shown that a PSO particle at stagnation (i.e. when no improvements to personal bests are occurring and the particles have effectively decoupled) exhibits *bursts* of outliers[88] – temporary excursions of the particle to large distances away from the attractors. A burst will typically grow to a maximum and then return through a number of damped oscillations to the region of the attractors. The origin of bursts, and of the associated fattening of the tails of the positional distribution at stagnation, can be traced to the second order stochastic difference equation:

$$x(t+1) + a(t)x(t) + bx(t-1) = c(t) \tag{4.10}$$

which is equivalent to SPSO with the identification $a(t) = c(\epsilon_1 + \epsilon_2) - w - 1$, $b = w$ and $c(t) = c(\epsilon_1 p_1 + \epsilon_2 p_2)$ for fixed attractors $p_{1,2}$. Since $max(|a|) > 0$, amplification of $x(t)$ can occur through repeated multiplication of x(t) by a despite the second order reduction by multiplication by the constant b. Interestingly, the distribution tail of $|x|$, by virtue of the bursts that become increasingly less probable for increasing size, is fattened compared to an exponential fall-off as provided by, for example, a Gaussian. A theoretical justification of these power laws and some empirical tests can be found in [129].

PSO bursts differ from the random outliers generated by PSO models which replace velocity by sampling from a distribution with fat tails such as a Levy distribution[80]. In contrast to the outliers of these "bare bones" formulations[79], the outliers from bursts occur in sequence, and entirely in a single dimension. Bursting will therefore produce periods of rectilinear motion where the particle will have a large velocity parallel to a coordinate axis.

Furthermore, large bursts may take the particle outside the search space. Although this will not incur any penalty in lost function evaluations if particles that exit the feasible bounds of the problem are not evaluated, as previously defined, they are not contributing to the search whilst in this outer space. PSO-DR, which is predicted not to have bursts[129], therefore provides a salient comparison.

4.5.1 Bursting under PSO-DR

In order to investigate bursting behaviour in PSO-DR and SPSO an empirical measure was devised. This bursting measure was implemented to highlight when a particle had a velocity in a single dimension that

was considerably higher than the velocities in all other dimensions. Bursting patterns of behaviour were detected by reporting every function evaluation where particle velocity in a single dimension was a set amount λ times higher than the next-highest dimensional velocity.

Bursting behaviour is demonstrated in figure 4.8, where the velocity of a single particle in a 10-dimensional problem is shown. On the plot of the multi-dimensional velocity of the SPSO particle, it can be seen that velocity in a single dimension increases suddenly and dramatically while remaining relatively level and low in all other dimensions. This is a prime example of a velocity burst. Velocity for a PSO-DR Model 3 particle is also shown in figure 4.8, and demonstrates the absence of such bursts.

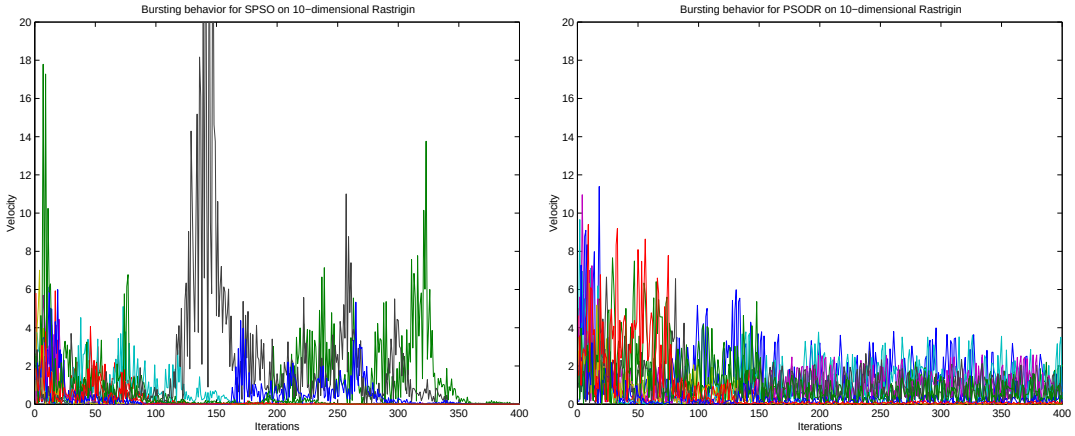
Examination of these empirical analyses show that PSO-DR lacks any bursting behaviour on the scale of SPSO while demonstrating equal or superior performance on 13 of the 14 benchmark functions. Figure 4.5 shows the percentage of particle updates where burst patterns of behaviour were seen for various values of λ during the performance tests of section 4.4.

Analysis performed on statistics of several functions shows that particle updates involving bursts are far less effective than more common non-bursting updates. For example, results obtained in the course of this study showed that for a SPSO algorithm with a ring topology on f_5, 20.1% of *all* particle updates over 300k function evaluations result in an improvement to the particle's best found position. When λ is set to 100, i.e. a burst is registered when a particle is moving in one dimension at least $100\times$ its velocity in any other dimension, only 1.8% of updates involving these bursts result in an improvement to the personal best. Even more tellingly, on average 0.9% of *all* particle updates on this problem improve the best found position of the entire swarm, while only 0.01% of updates for bursting particles. Even when the λ burst identifier is set to the relatively low value of $10\times$ the velocity of any other dimension, particles in this state only update their personal best position in 12.45% of evaluations, still much lower than the aforementioned 20.1% of all updates.

The results obtained and described here support the findings of an associated theoretical study of burst behaviour[129], and help to show that by eliminating what seems to be another extraneous feature of the standard particle swarm - its multiplicative stochasticity - PSO-DR again removes an integral component of the algorithm that is in fact both unnecessary and potentially detrimental to optimization functionality. Up to this point it has been suggested in PSO literature that velocity bursts are necessary to the optimization process for various hypothesized reasons [3, 68, 79, 90] – the theoretical proofs and empirical demonstrations in [129] and this work indicate that this is not the case.

4.6 Discussion

Simplification of the standard PSO algorithm is an important step toward understanding how and why it is an effective optimizer. By removing components of the algorithm and seeing how this affects performance, we are granted insight into what those components contribute to overall particle and swarm



(a) SPSO (b) PSO-DR

Figure 4.8: Representative particle velocities for SPSO and PSO-DR on 10D Rastrigin (color used to differentiate dimensions)

behaviour.

In particular, a very simple particle swarm optimizer is proposed, PSO-DR Model 3:

$$x_{id} = x_{id} + \phi(r_{id} - x_{id})$$

which offers competitive performance to standard PSO but removes multiplicative randomness, inertia, and the personal memory term p_i from the position update. The full PSO-DR Model 3 algorithm is provided in pseudocode for easy replication.

There is still much to be done before questions concerning PSO behaviour can be completely answered, and it is expected that future areas of PSO research will be focused on understanding the basic algorithm that powers both the standard implementation and its variants. That work is already underway, most notably in the defining of a method of analysis for the SPSO algorithm[78, 76, 77]. Adapting this method for use on the various models of PSO-DR is the focus of the next chapter.

In that light, the PSO-DR variant is important not only because of its improved performance on several benchmark functions, but also because it's simplified state allows us to examine what happens to the standard algorithm when pieces are modified or removed. Based on the results presented here and in [129], it can be argued that large bursts are not generally beneficial or integral to PSO performance, and may possibly be detrimental. Although the presence of particle outliers is demonstrably important for swarm optimization (as shown in an analysis of the bare bones variant[79]), bursts, which are sequences of extreme particle positions, occurring along an axis and reaching outside the search space, remain a feature of velocity-based swarms. This work, which compares standard PSO to a burst-free but comparable optimizer suggests that such bursts are disadvantageous in general.

Further, the replacement of the direct personal influence operator p_i from SPSO with the recom-

The PSO-DR Model 3 (PSO-DRS) Algorithm

initialize constant $\phi = 1.2$
initialize swarm best found fitness $fg = maximum\ possible\ fitness$
for each particle i where $i = 1..50$ **do**
  if performance testing **then**
    initialize random particle position vector $\overrightarrow{x_i}$ within the search space using *region scaling*
  else
    initialize random particle position vector $\overrightarrow{x_i}$ within the search space
  initialize particle best found position vector $\overrightarrow{p_i} = \overrightarrow{x_i}$
  set particle left neighbor $nl_i = (i - 1)$ unless $i = 1$, in which case $nl_i = 50$
  set particle right neighbor $nr_i = (i + 1)\ mod\ 50$
  calculate particle current fitness $f_i = f(\overrightarrow{x_i})$
  initialize particle best fitness $fp_i = f_i$
  if particle fitness $f_i < fg$ **then**
    set swarm best found fitness $fg = f_i$
    set swarm best found position $\overrightarrow{pg} = \overrightarrow{x_i}$
for each step prior to termination criteria **do**
  for each particle i where $i = 1..50$ **do**
    initialize random vector $\overrightarrow{\eta}$ with independent elements either 0 or 1
    initialize discrete vector $\overrightarrow{r_i} = \overrightarrow{\eta}\overrightarrow{nl_i} + (1 - \overrightarrow{\eta})\overrightarrow{nr_i}$
    update particle position $\overrightarrow{x_i} = \overrightarrow{x_i} + \phi\,(\overrightarrow{r_i} - \overrightarrow{x_i})$
  calculate particle current fitness $f_i = f(\overrightarrow{x_i})$
  if particle current fitness $f_i < fp_i$ **then**
    set particle best found fitness $fp_i = f_i$
    if particle current fitness $f_i < fg$ **then**
      set swarm best found fitness $sf = f_i$
      set swarm best found position $\overrightarrow{pg} = \overrightarrow{x_i}$
best found fitness for the optimization is fg
best found position for the optimization is $\overrightarrow{pg}$

binant term r_i, derived from the particle neighborhood, in PSO-DR strengthens assertions that swarm algorithms is largely reliant on social interaction as opposed to personal "cognitive" experience[61, 87]. This is further supported by the effectiveness of PSO-DR Model 3, which lacks a cognitive term, and in fact any reference to a particle's personal best position, in the update equations altogether. The social behaviour occurring inside of a swarm is still a wide-open area in the field, and will hopefully constitute a great deal of the future research devoted to the development of a better understanding of this deceptively simple optimizer.

4.7 Conclusions

This chapter has focused on the development of a simplified form of the standard PSO algorithm defined previously. This new formulation is based on the introduction of a recombinant term into the update equations, taking the place of the personal best found position of a particle, also known as the cognitive term. This recombinant term is established as a positional vector formed from a combination of the best found positions of the particle's neighbors.

Simplifications to the original PSO-DR algorithm, referred to here as Model 1, were applied after empirical results indicated that the w term was extraneous for the attainment of performance equivalent

to that of Model 1 given appropriate adjustments to the ϕ parameter. Further simplifications to this Model 2 formulation were applied when it was found that the entire neighborhood best term, aka the social term, was extraneous as well.

The new formulation of the algorithm, referred to as PSO-DR Model 3, was found to return performance results equivalent or superior to the standard PSO defined in the previous chapter despite the removal of large parts of the update equations. This simplified form also showed similar behaviour in terms of particle interaction and swarm movement. Further, the removal of the multiplicative stochasticity of the standard PSO algorithm with PSO-DR was shown to correspond to a removal of the velocity bursting behaviour seen there.

The removal of multiplicative stochasticity also simplifies the process of applying a mathematical analysis of the PSO-DR algorithm, allowing for an investigation of the sampling distribution. This investigation is the focus of the next chapter.

Chapter 5

Mathematical Analysis of PSO-DR

5.1 Introduction

A mathematical analysis of the dynamics of the PSO-DR algorithm provides a clear picture of its sampling distribution as well as a proof of convergence for appropriate parameter settings. Such analyses have been performed on numerous characteristics of the traditional PSO algorithm equivalent to that defined here in chapter 3, and are detailed in section 2.6.

Of these, most have required a significant number of assumptions to be made about the algorithm in order to simplify it to a more easily analysable state. The most common of these are the removal of stochasticity from the update equations, and the assumption of stagnation, i.e. that a particle is unable to update the personal-best or swarm-best positions and is in a constant state of searching for an improved point in the fitness landscape. While the latter of these, the assumption of stagnation, is necessary to provide a static range for the sampling distribution, the removal of stochasticity is a fundamental, unrepresentative alteration to the behaviour of updating particles.

With the publication of his exact analysis of the sampling distribution of PSO, Poli removed all assumptions beyond stagnation[77, 78], allowing for exact determination of the characteristics of the sampling distribution in the presence of stochasticity. Given the general applicability of Poli's means of analysis, this chapter provides a statistical analysis of the various forms of PSO-DR using an adapted form of his procedure. Like Poli's method, the approach taken here assumes only stagnation in a particle.

In performing this analysis, we are able to move beyond the information granted by empirical study about *whether* the PSO-DR algorithms are able to effectively optimize non-linear problems, and look at *how* this optimization is carried out. More practically, this information grants insight into the range of appropriate parameter settings required for the algorithm that allow it to operate in a desired fashion. The previous chapter empirically demonstrated that the PSO-DR algorithms were able to optimize to at least some extent given a ϕ setting between approximately 0.0 and 2.0 – an exact analysis of the sampling distribution will allow us to predict with certainty how they will behave for any setting of this parameter.

Going beyond a greater understanding of the behaviour of the PSO-DR algorithms, by proving the convergence ability of the PSO-DR algorithms, we can determine the moments of the sampling distribution under convergence conditions. These moments, the first two (corresponding to the mean and the standard deviation) of which are calculated in this chapter, allow us to "get around" the limitations imposed by No Free Lunch theorem (see section 2.1.3) by matching the algorithm to problems for which we know that this sampling distribution is well-suited. If the sampling distributions for all conceivable optimization algorithms were known, the best approach for solving any conceivable problem could be obtained. By providing the sampling distribution of PSO-DR, we contribute to the sum of this knowledge.

5.2 Generalization of PSO-DR

After removing the dimensional and individual subscripts and adding time-series markers t, the three forms of the discrete recombinant swarm discussed in the previous chapter are given below.

First, the originally proposed discrete recombinant PSO algorithm[82], in its *local*, i.e. 2-neighbor formulation appears as:

$$DR\,M1 : x_{t+1} = x_t \; + \; w(x_t \; - \; x_{t-1}) \; + \; \frac{\phi}{2}(r - x_t) \; + \; \frac{\phi}{2}(p_1 - x_t) \tag{5.1}$$

where p_1 is the best position found by any neighbor. Note that the velocity term has been combined into the positional update here.

The velocity-less refinement of the original PSO-DR algorithm, Model 2, appears as:

$$DR\,M2 : x_{t+1} = x_t \; + \; \frac{\phi}{2}(r - x_t) \; + \; \frac{\phi}{2}(p_1 \; - \; x_t) \tag{5.2}$$

Finally, the form obtained when the neighborhood term was determined to be unnecessary for convergence, Model 3, is:

$$DR\,M3 : x_{t+1} = x_t \; + \; \phi(r \; - \; x_t) \tag{5.3}$$

aka PSO-DRS. For each of these equations:

$$r = \eta p_1 \; + \; (1 \; - \; \eta)p_2 \tag{5.4}$$

where η is either 0 or 1 with equal probability, and p_1 and p_2 represent the best positions of the particle's two neighbors in a ring topology. Unlike the standard PSO formulation, PSO-DR involves only additive rather than multiplicative stochasticity due to this recombinant term[129].

Poli's technique uses the expectation operator to define a fixed point of the SPSO update equation

to be a combination of the best position of the particle and the best neighborhood position , i.e. $p = \frac{c_1 y + c_2 \hat{y}}{c_1 + c_2}$[77]. A fixed point can similarly be obtained for the PSO-DR equations that is entirely dependent on the recombinant term r. To do this, the recombinant and the best-neighbor components of the first two PSO-DR equations (5.1 and 5.2) can be combined:

$$\begin{aligned} p_r &= \frac{\phi}{2}(r - x_t) + \frac{\phi}{2}(p_1 - x_t) \\ &= \frac{\phi}{2}\eta p_1 + \frac{\phi}{2}(1 - \eta)p_2 + \frac{\phi}{2}p_1 - \phi x_t \\ &= \frac{\phi}{2}\eta p_1 + \frac{\phi}{2}p_1 - \frac{\phi}{2}\eta p_2 + \frac{\phi}{2}p_2 - \phi x_t \\ &= \phi\left(\frac{\eta p_1 + p_1 - \eta p_2 + p_2}{2} - x_t\right) \\ &= \phi\left(\frac{(\eta + 1)p_1 + (1 - \eta)p_2}{2} - x_t\right) \end{aligned} \tag{5.5}$$

We can extract the fixed-point recombinant term from this equation, namely:

$$\kappa_{1,2} = \frac{(\eta + 1)p_1 + (1 - \eta)p_2}{2} \tag{5.6}$$

Obviously the recombinant component of PSO-DR Model 3 ($\phi(r - x_t)$) fits this form, with simply:

$$\kappa_3 = \eta p_1 + (1 - \eta)p_2 \tag{5.7}$$

per equation 5.4.

With this generalized recombinant term κ in place, we can now describe any of the PSO-DR formulations using a single equation:

$$x_{t+1} = x_t + w(x_t - x_{t-1}) + \phi(\kappa - x_t) \tag{5.8}$$

where $w = 0.0$ for Model 2 and Model 3.

As per Blackwell[129], we can rewrite equation 5.8 as a stochastic second order difference equation of the standard form:

$$x_{t+1} + a_t x_t + b x_{t-1} = c_t \tag{5.9}$$

with:

$$\begin{cases} a_t &= -(1+w-\phi) \\ b &= w \\ c_t &= \phi\kappa \end{cases} \tag{5.10}$$

In PSO-DR Model 1 $w = 0.5$ per Peña[82], while PSO-DR Models 2 and 3 simply set $w = 0$ to remove the velocity component. Given that $E[\eta] = \frac{1}{2}$, applying the expectation operator to the recombinant term for PSO-DR Models 1 and 2 (eq 5.6) gives:

$$\begin{aligned} E[\kappa_{1,2}] &= \frac{(1 + E[\eta])p_1 + (1 - E[\eta])p_2}{2} \\ &= \frac{\frac{3}{2}p_1 + \frac{1}{2}p_2}{2} \\ &= \frac{3p_1 + p_2}{4} \end{aligned} \tag{5.11}$$

and for the PSO-DR Model 3 term (5.6):

$$\begin{aligned} E[\kappa_3] &= E[\eta]p_1 + (1 - E[\eta])p_2 \\ &= \frac{p_1 + p_2}{2} \end{aligned} \tag{5.12}$$

From these two values, we can see that the fixed point for the first two models is on the line connecting the two neighborhood particles – 25% closer to the best neighborhood position p_1 for Models 1 and 2, and evenly spaced between the two positions for Model 3. This is apparent from a comparison of the algorithms, where the best neighbor position appears with full "weighting" in the social component $\frac{\phi}{2}(p_1 - x_t)$, and half in the cognitive component $\frac{\phi}{2}(r - x_t)$ for Models 1 and 2, while Model 3 discards the social component, leaving only the half-weighted cognitive component.

For use in determining the dynamics of higher moments in the next section, we calculate the expected values for κ^2 for the Models 1 and 2 formulation:

$$\begin{aligned}
\kappa_{1,2}^2 &= \left(\frac{(1\ +\ \eta)p_1\ +\ (1\ -\ \eta)p_2}{2}\right)^2 \\
\kappa_{1,2}^2 &= \frac{(\eta p_1\ -\ \eta p_2\ +\ p_1\ +\ p_2)^2}{4} \\
&= \frac{p_1^2\,(\eta^2\ +\ 2\,\eta\ +\ 1)\ +\ p_2^2\,(\eta^2\ -\ 2\,\eta\ +\ 1)\ +\ p_1\,p_2\,(2\ -\ 2\,\eta^2)}{4}
\end{aligned} \tag{5.13}$$

Applying the expectation operator, we obtain:

$$\begin{aligned}
E[\kappa_{1,2}^2] &= \frac{p_1^2\,(E[\eta^2]\ +\ 2\,E[\eta]\ +\ 1)\ +\ p_2^2\,(E[\eta^2]\ -\ 2\,E[\eta]\ +\ 1)\ +\ p_1\,p_2\,(2\ -\ 2\,E[\eta^2])}{4} \\
&= \frac{\frac{5p_1^2}{2}\ +\ \frac{p_2^2}{2}\ +\ p_1p_2}{4} \\
&= \frac{5p_1^2\ +\ p_2^2\ +\ 2\,p_1\,p_2}{8}
\end{aligned} \tag{5.14}$$

The Model 3 formulation differs only slightly:

$$\begin{aligned}
\kappa_3^2 &= (\eta p_1\ +\ (1\ -\ \eta)p_2)^2 \\
&= \eta^2 p_1^2\ +\ \eta^2 p_2^2\ -\ 2\eta^2 p_1 p_2\ +\ 2\eta p_1 p_2\ -\ 2\eta p_2^2\ +\ p_2^2 \\
&= p_1^2\,\eta^2\ +\ p_2^2\,(\eta^2\ -\ 2\,\eta\ +\ 1) \qquad (5.15) \\
E[\kappa_3^2] &= p_1^2\,E[\eta^2]\ +\ p_2^2\,(E[\eta^2]\ -\ 2\,E[\eta]\ +\ 1) \\
&= \frac{p_1^2 + p_2^2}{2} \qquad (5.16)
\end{aligned}$$

5.3 Dynamics of the PSO-DR Sampling Distribution

With our generalized form of the PSO-DR update equations, we can now compute the dynamics of the first two moments of the sampling distribution. These correspond to the mean and the variance of the distribution. As per Poli's definition of this method[76, 77, 78], we are assuming stagnation, i.e. that the particles under analysis are in search of an improved found best position.

5.3.1 First Moment

By rewriting equation 5.8 to combine the constant terms, we obtain:

$$\begin{aligned} x_{t+1} &= x_t + w(x_t - x_{t-1}) + \phi\,(\kappa - x_t) \\ &= x_t + wx_t - wx_{t-1} + \phi\,\kappa - \phi x_t \\ &= \gamma\, x_t - w\, x_{t-1} + \phi\,\kappa \end{aligned} \tag{5.17}$$

where $\gamma = (1 + w - \phi)$. This allows for a simple application of the expectation operator:

$$E[x_{t+1}] = \gamma E[x_t] - wE[x_{t-1}] + \phi E[\kappa] \tag{5.18}$$

We can find the fixed point for this equation by substituting p for $E[x_t]$ and some relatively simple algebra:

$$\begin{aligned} p &= \gamma\, p - w\, p + \phi\, E[\kappa] \\ &= p + w\, p - \phi\, p - w\, p + \phi\, E[\kappa] \\ \phi\, p &= \phi\, E[\kappa] \\ p &= E[\kappa] \end{aligned} \tag{5.19}$$

From this we can see that the recombination operator determines the point of convergence of the algorithm. As PSO-DR Models 1 and 2 use the same recombination operator, we know that either algorithm will converge to the same point, assuming stability. PSO-DR Model 3, on the other hand, will converge to a separate point, again assuming stability.

The stability of equation 5.18 can be found by reformulating this second-order equation into a vector first-order system of equations $z_t = [x_{t+1}\ x_t]^T$. This would take the form:

$$z_{t+1} = M_1 z_t + b \tag{5.20}$$

where the matrix M_1 is:

$$M_1 = \begin{pmatrix} \gamma & -w \\ 1 & 0 \end{pmatrix} \tag{5.21}$$

and the forcing vector:

$$b_1 = \begin{pmatrix} E[\kappa]\phi \\ 0 \end{pmatrix} \tag{5.22}$$

Now we are able to determine the stability of the system at the order under investigation (in this case first-order) using the values of ϕ for which all of the eigenvalues of M_1 are less than 1, i.e.:

$$\Lambda_m = max_i|\lambda_i| < 1 \tag{5.23}$$

The eigenvalues of M_1 are:

$$\begin{aligned} \lambda_{1,2} &= \frac{\gamma \pm \sqrt{(\gamma)^2 - 4w}}{2} \\ &= \frac{1 + w - \phi \pm \sqrt{(1 + w - \phi)^2 - 4w}}{2} \end{aligned} \tag{5.24}$$

$$\tag{5.25}$$

As the eigenvalues of M_1 do not have a dependence on κ, we can conclude that while the point of convergence depends on the choice of recombinant operator, order-1 stability is independent of this term. For the mean of $E[x_t]$ to converge to a fixed point, the magnitude of the largest eigenvalue must be less than 1. The region inside of the lines, which correspond to stability conditions, shown in figure 5.1 shows the combinations of the w and ϕ terms for which the eigenvalues are less than 1. We refer to this as the *stability region*.

It is notable that the regions of best performance for PSO-DR seen in figure 4.1 of the previous chapter match up well to this analytically derived stability region. This empirical evidence supports the indication that the parameter settings within the stability region allow the particle to converge to and hence search the area around its best found positions for improvements.

As the algorithms for PSO-DR Models 2 and 3 remove the velocity term w, i.e. set $w = 0$, it can be seen from the above generalized stability conditions that the region of stability for these two algorithms is defined simply by $0 < \phi < 2$. This too is confirmed in figure 5.1.

5.3.2 Second Moment

While the region of order-1 stability guides us in choosing values for w and ϕ that guarantee convergence of the mean of the individual particle under stagnation, this does not guarantee that the particle will approach and "settle" on that single point, but only that the point will be the mean of the sampling

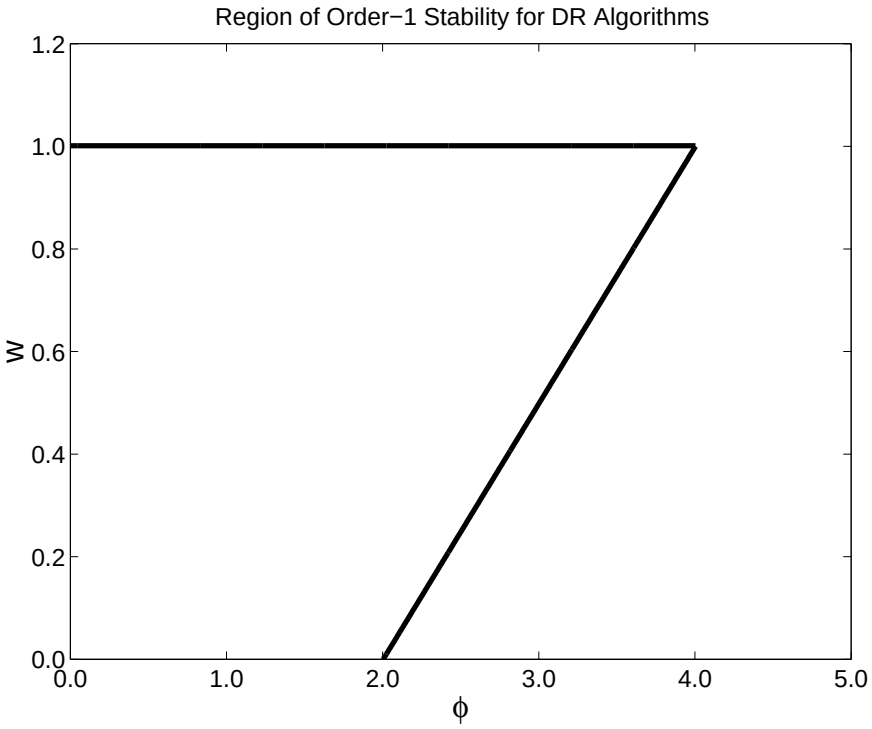


Figure 5.1: Region of order-1 stability for PSO-DR algorithms

distribution – i.e. the particle's motion will be centred around it. For this approach and the associated search of the immediate area of the mean we need parameter choices that guarantee convergence of the variance as well, which is described by the second moment, i.e. order-2 stability.

We can determine the dynamics of the second moment via $E[x_{t+1}^2]$, $E[x_{t+1}x_t]$, and $StdDev[x_t]$. To begin, we obtain x_{t+1}^2:

$$\begin{aligned} x_{t+1}^2 &= (\gamma\, x_t \;-\; w\, x_{t-1} \;+\; \phi\, \kappa)^2 \\ &= \gamma^2\, x_t^2 \\ &+\; 2\,\gamma\,\phi\,\kappa\, x_t \\ &-\; 2\,\gamma\, w\, x_t\, x_{t-1} \\ &+\; w^2\, x_{t-1}^2 \\ &-\; 2\, w\,\phi\,\kappa\, x_{t-1} \\ &+\; \phi^2\,\kappa^2 \end{aligned} \tag{5.26}$$

Applying the expectation operator to both sides, we obtain:

$$\begin{aligned} E[x_{t+1}^2] \;=\; & E[x_t^2]\,\gamma^2 \\ & +\; E[x_t]\,E[\kappa]\,2\,\gamma\,\phi \\ & -\; E[x_t\,x_{t-1}]\,2\,\gamma\,w \\ & +\; E[x_{t-1}^2]\,w^2 \\ & -\; E[x_{t-1}]\,E[\kappa]\,2\,w\,\phi \\ & +\; E[\kappa^2]\,\phi^2 \end{aligned} \tag{5.27}$$

Next we compute $x_{t+1}x_t$ by multiplying both sides of x_{t+1} by x_t:

$$x_{t+1}x_t \;=\; x_t^2\,(\gamma) \;-\; x_t\,x_{t-1}\,w \;+\; x_t\,\kappa\,\phi \tag{5.28}$$

and thereby:

$$E[x_{t+1}x_t] \;=\; E[x_t^2]\,(\gamma) \;-\; E[x_t\,x_{t-1}]\,w \;+\; E[x_t]\,E[\kappa]\,\phi \tag{5.29}$$

As Poli found for a PSO algorithm equivalent to the standard defined in chapter 3, we can see that both of these equations depend on x_t, which tells us that neither can attain a fixed point unless x_t has done so. In other words, the variance of the system cannot converge unless the mean has done so as well. The swarm as a whole cannot converge to a fixed point unless an individual particle has done so.

Once derived, equations 5.18, 5.27, and 5.29 can be put into matrices as above, in an extended first order system to facilitate an analysis of the stability of the system where:

$$z(t)_2 \;=\; \begin{pmatrix} E[x_t] \\ E[x_{t-1}] \\ E[x_t^2] \\ E[x_{t-1}^2] \\ E[x_t x_{t-1}] \end{pmatrix} \tag{5.30}$$

the matrix:

$$M_2 = \begin{pmatrix} \gamma & -w & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ E[\kappa]2\gamma\phi & -E[\kappa]2w\phi & \gamma^2 & w^2 & -2\gamma w \\ 0 & 0 & 1 & 0 & 0 \\ E[\kappa]\phi & 0 & \gamma & 0 & -w \end{pmatrix} \tag{5.31}$$

and the forcing vector:

$$b_2 = \begin{pmatrix} E[\kappa]\phi \\ 0 \\ E[\kappa^2]\phi^2 \\ 0 \\ 0 \end{pmatrix} \tag{5.32}$$

The eigenvalues of M_2 are shown in the system of equations 5.33. As with the order-1 PSO-DRS system stability, as well as the order-1 SPSO system Poli derived[77], the lack of dependency on the positional value x of the update equation indicates that the second-order stability of PSO-DR is wholly independent of the physical location or distribution of the particles comprising the swarm and is determined completely by the values of w and ϕ.

$$\begin{cases} \lambda_1 = w \\ \lambda_{2,3} = \frac{\gamma \pm \sqrt{(w-\phi)^2 - 2\phi - 2w + 1}}{2} \\ \lambda_{4,5} = \frac{1 - 2\phi + (\phi - w)^2 \pm \gamma\sqrt{1 - 2(\phi + w) + (\phi - w)^2}}{2} \end{cases} \tag{5.33}$$

As above, we determine the stability of the system using the values of ϕ and w for which all of the eigenvalues in the system of equation 5.33 are less than 1. Empirical methods were again used to determine this region – results are shown in figure 5.2. Comparison of the regions of stability for the order-1 and the order-2 systems reveals that, unlike for the standard PSO, the two coincide perfectly.

Deriving the fixed points of the order-2 equations $E[x_t^2]$ and $E[x_{t+1}x_t]$ can be done in the same way as above with the order-1 equation $E[x_t]$. As we know that $E[x_t]$ converges to the fixed point equal to the recombinant term $p = E[\kappa]$, we can substitute this into the equations $E[x_t^2]$, replacing $E[x_t]$ and $E[x_{t-1}]$, which gives us:

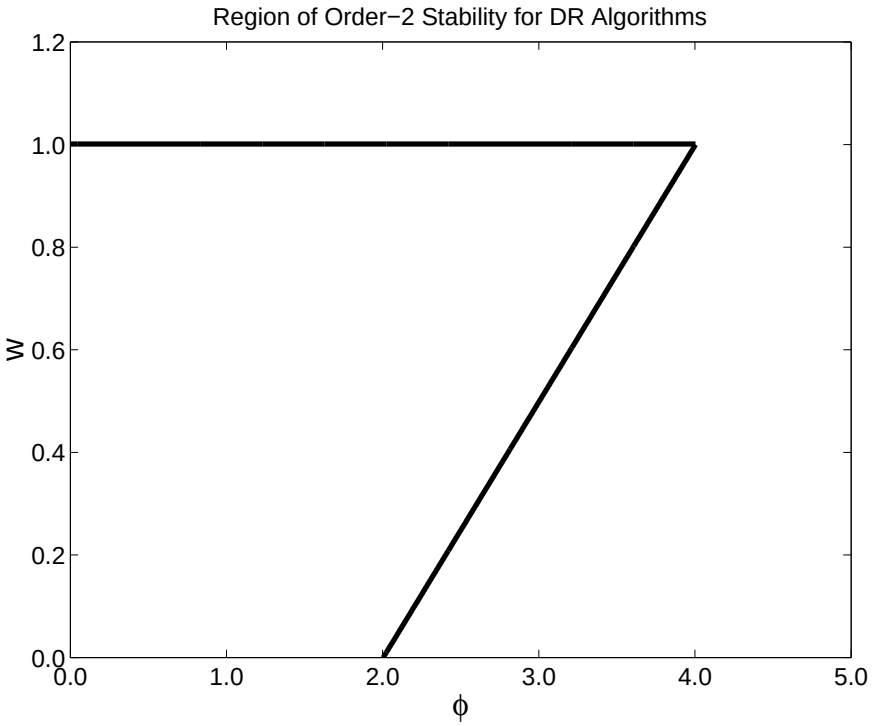


Figure 5.2: Region of order-2 stability for PSO-DR algorithms

$$\begin{aligned} E[x_{t+1}^2] \;=\; & E[x_t^2]\,\gamma^2 \\ & -\; E[x_t\; x_{t-1}]\; 2\,\gamma\, w \\ & +\; E[x_{t-1}^2]\, w^2 \\ & +\; E[\kappa]^2\,(2\,\gamma\,\phi \;-\; 2\,w\,\phi) \\ & +\; E[\kappa^2]\,\phi^2 \end{aligned} \tag{5.34}$$

and for $E[x_{t+1}x_t]$:

$$E[x_{t+1}x_t] \;=\; E[x_t^2]\,\gamma \;-\; E[x_t\; x_{t-1}]\, w \;+\; E[x_t]\, E[\kappa]\,\phi \tag{5.35}$$

We can now determine the fixed points for the three order-2 equations by substituting in these points p_{x^2} and p_{xx}:

$$p_{x^2} \;=\; p_{x^2}\,\gamma^2 \;+\; p_{x^2}\,w^2 \;-\; p_{xx}\,2\,\gamma\,w \;+\; E[\kappa]^2(2\,\gamma\,\phi \;-\; 2\,w\,\phi) \;+\; E[\kappa^2]\phi^2 \tag{5.36}$$

$$p_{xx} \;=\; p_{x^2}\,\frac{\gamma}{1\;+\;w} \;+\; E[\kappa]^2\,\frac{\phi}{1\;+\;w} \tag{5.37}$$

Note that the solved equation for p_{xx} is equivalent to the one found by Poli[77] for SPSO. Substituting this into the equation for p_{x^2} and a bit of algebra, we obtain the fixed point:

$$\begin{aligned}
p_{x^2} \;&=\; p_{x^2}\,\gamma^2 \;+\; p_{x^2}\,w^2 \\
&-\; \left(p_{x^2}\,\frac{\gamma}{1\;+\;w} \;+\; E[\kappa]^2\,\frac{\phi}{1\;+\;w}\right)\,2\,\gamma\,w \\
&+\; E[\kappa]^2(2\,\gamma\,\phi \;-\; 2\,w\,\phi) \;+\; E[\kappa^2]\phi^2 \\
\\
&=\; p_{x^2}\,\left(\gamma^2 \;+\; w^2 \;-\; \frac{\gamma}{1\;+\;w}\right) \\
&+\; E[\kappa]^2\left(2\,\gamma\,\phi \;-\; 2\,w\,\phi \;-\; \frac{\phi}{1\;+\;w}\right) \;+\; E[\kappa^2]\phi^2 \\
\\
&=\; \frac{E[\kappa]^2\;2\,\phi\,(\gamma \;-\; w \;-\; w^2) \;+\; E[\kappa^2]\;\phi^2\;(1\;+\;w)}{1\;+\;w\;-\;\gamma^2\;+\;\gamma^2\,w\;-\;w^2\;-\;w^3} \\
\\
&=\; \frac{E[\kappa]^2\;(2\;-\;2\,\phi\;-\;2\,w^2) \;+\; E[\kappa^2]\;(\phi\;+\;w\,\phi)}{2\;-\;2\,w^2\;+\;w\,\phi\;-\;\phi}
\end{aligned} \tag{5.38}$$

which can then be substituted back into p_{xx} to obtain the other fixed point:

$$p_{xx} = \left(\frac{E[\kappa]^2\,(2\,-\,2\,\phi\,-\,2\,w^2)\,+\,E[\kappa^2]\,(\phi\,+\,w\,\phi)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi}\right)\,\frac{\gamma}{1\,+\,w}\,+\,E[\kappa]^2\,\frac{\phi}{1\,+\,w}$$

$$= \frac{E[\kappa]^2\,\gamma\,(2\,-\,2\,\phi\,-\,2\,w^2)\,+\,E[\kappa]^2\,\phi(2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi)\,+\,E[\kappa^2]\,\gamma\,(\phi\,+\,w\,\phi)}{(2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi)\,(1\,+\,w)}$$

$$= \frac{E[\kappa]^2(\phi^2+\phi^2w-2\phi-2w\phi+2w-2w^2-2w^3+2)}{(2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi)\,(1\,+\,w)}$$
$$+\,\frac{E[\kappa^2](-\phi^2-w\phi^2+\phi+2w\phi+w^2\phi)}{(2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi)\,(1\,+\,w)} \tag{5.39}$$

$$= \frac{E[\kappa]^2(\phi^2\,-\,2\,\phi\,-\,2\,w^2\,+\,2)\,+\,E[\kappa^2](-\phi^2\,+\,w\,\phi\,+\,\phi)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi} \tag{5.40}$$

Finally, we can determine the standard deviation of the system via the well-known definition $StdDev[x_t]\,=\,\sqrt{E[x_t^2]-E[x_t]^2}$:

$$\begin{aligned}
p_{sd} &= \sqrt{p_{x^2}\,-\,p_x^2} \\
&= \sqrt{\frac{E[\kappa]^2\,(2\,-\,2\,\phi\,-\,2\,w^2)\,+\,E[\kappa^2]\,(\phi\,+\,w\,\phi)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi}\,-\,E[\kappa]^2} \\
&= \sqrt{\frac{E[\kappa^2]\,(\phi\,+\,w\,\phi)\,+\,E[\kappa]^2\,(2\,-\,2\,\phi\,-\,2\,w^2)\,-\,E[\kappa]^2\,(2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi}} \\
&= \sqrt{\frac{E[\kappa^2]\,(\phi\,+\,w\,\phi)\,-\,E[\kappa]^2\,(\phi+w\,\phi)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi}} \\
&= \sqrt{\frac{(\phi\,+\,w\,\phi)(E[\kappa^2]\,-\,E[\kappa]^2)}{2\,-\,2\,w^2\,+\,w\,\phi\,-\,\phi}}
\end{aligned} \tag{5.41}$$

Like the fixed point for the first moment, x_t, we can see that while the stability of the second order is dependent only on parameter choices for w and ϕ, the fixed points for all three second moment terms are also dependent on the choice of recombinant operator. In other words, we can guarantee convergence through appropriate selection of these two parameters, and we can choose which point will be converged to through selection of the recombinant term.

5.4 Initial Conditions

To evaluate the initial conditions of the PSO-DR update equations we must bring in knowledge of the swarm's initialization properties. Region scaling has been used in previous chapters for empirical optimizations, but this technique is used only to ensure that performance is not impacted by the structure

of the problem landscape – here we will use the more natural random initialization within a symmetric space $[-\Omega, \Omega]$. This gives us $E[x_0] = 0$.

While the PSO-DR Models 2 and 3 updates are first-order equations, Model 1 is second order, like SPSO. In order to maintain generality we will split equation 5.8 into separate velocity and positional updates, respectively $v_{t+1} = wv_t + \phi(\kappa - x_t)$ where $v_t = x_t - x_{t-1}$, and $x_{t+1} = x_t + v_{t+1}$. The velocity term is second-order in this formulation, but for Models 2 and 3 it is effectively first-order, given their parameter setting $w = 0$. As a particle's velocity is initialized in the same way as its position, we can similarly set $E[v_0] = 0$.

As the velocity term is second-order, we need to define the value of $E[v_1]$ before we can proceed. The equations above allow us to find the value:

$$\begin{aligned} E[v_1] &= w\, E[v_0] + \phi\,(E[\kappa] - E[x_0]) \\ &= 0 + \phi\,(E[\kappa] - 0) \\ &= \phi E[\kappa] \end{aligned} \tag{5.42}$$

Using the fixed point values for the recombinant term, this gives us the value $\phi\frac{3p_1+p_2}{4}$ for Models 1 and 2, and $\phi\frac{p_1+p_2}{2}$ for Model 3. Because $x_1 = x_0 + v_1$, we know that:

$$\begin{aligned} E[x_1] &= E[x_0 + v_1] \\ &= E[x_0] + E[v_1] \\ &= \phi E[\kappa] \end{aligned} \tag{5.43}$$

Now that we have initial conditions for the mean of the distribution, we can find the initial conditions to help us determine the variance. Because of the uniform initialization within $[-\Omega, \Omega]$ we know that $E[x_0^2] = E[v_0^2] = \frac{\Omega^2}{3}$. In order to derive $E[x_1^2] = E[(x_0 + v_1)^2] = E[x_0^2 + 2x_0v_1 + v_1^2]$ we will first need a value for $E[x_0\ v_1]$. Note that v_1 depends on x_0, rendering them inseparable for these purposes.

$$\begin{aligned} E[x_0\, v_1] &= w\, E[v_0]\, E[x_0] + \phi\, E[x_0]\, (E[\kappa] - E[x_0]) \\ &= 0 + \phi\, E[x_0]\, E[\kappa] - \phi\, E[x_0]^2 \\ &= 0 - \phi\, \frac{\Omega^2}{3} \\ &= -\phi\, \frac{\Omega^2}{3} \end{aligned} \tag{5.44}$$

which can also be used to determine $E[x_1 x_0]$:

$$\begin{aligned} E[x_1 x_0] &= E[(x_0 + v_1)\, x_0] \\ &= E[x_0^2] + E[v_1\, x_0] \\ &= \frac{\Omega^2}{3} - \phi\, \frac{\Omega^2}{3} \\ &= \frac{\Omega^2}{3}\, (1 - \phi) \end{aligned} \tag{5.45}$$

Next we will find the value for $E[v_1^2]$:

$$\begin{aligned} E[v_1^2] &= (w\, E[v_0] + \phi\, (E[\kappa] - E[x_0]))^2 \\ &= w^2 E[v_0^2] + 2w\phi E[\kappa]E[v_0] - 2w\phi E[v_0]E[x_0] + \phi^2 E[\kappa]^2 - 2\phi^2 E[\kappa]E[x_0] + \phi^2 E[x_0^2] \\ &= w^2\, \frac{\Omega^2}{3} + 0 - 0 + \phi^2\, E[\kappa]^2 - 0 + \phi^2\, \frac{\Omega^2}{3} \\ &= (w^2 + \phi^2)\, \frac{\Omega^2}{3} + \phi^2\, E[\kappa]^2 \end{aligned} \tag{5.46}$$

The various conditions we have collected now finally allow us to specify $E[x_1^2]$:

$$\begin{aligned} E[x_1^2] &= E[x_0^2 + 2x_0 v_1 + v_1^2] \\ &= E[x_0^2] + 2\, E[x_0\, v_1] + E[v_1^2] \\ &= \frac{\Omega^2}{3} - 2\, \phi\, \frac{\Omega^2}{3} + (w^2 + \phi^2)\, \frac{\Omega^2}{3} + \phi^2\, E[\kappa]^2 \\ &= (1 - 2\phi + w^2 + \phi^2)\, \frac{\Omega^2}{3} + \phi^2\, E[\kappa]^2 \end{aligned} \tag{5.47}$$

5.5 Behaviour of the System

With all of our equations in hand for determining the mean, standard deviation, and fixed points of the PSO-DR algorithms, demonstrating the behaviour of the systems under various parameter combinations only requires us to start the simulation using the initial conditions determined above.

Figure 5.3 shows a PSO-DR Model 1 algorithm using Peña's original parameter settings of $w = 0.5$ and $\phi = 2$[82]. Panel (a) of the figure shows behaviour when p_1 and p_2 are at separate points. As expected, the mean x_t converges to the fixed point p, which is nearer to p_1. Because the particle is at stagnation, and p_1 and p_2 are not equal, the standard deviation converges to a value greater than zero. The convergence of both the mean and the variance here confirms that the selected parameter settings are in the region of stability.

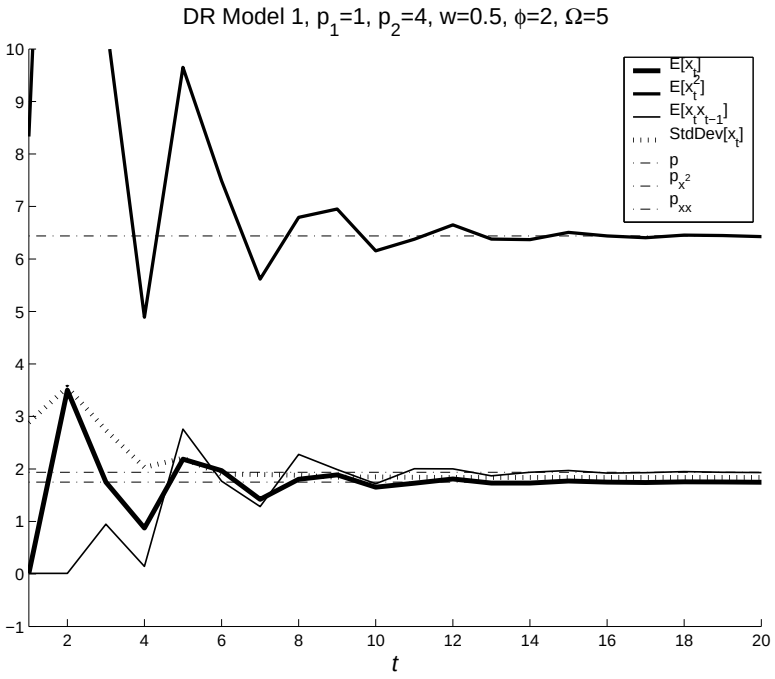
Panel (b) of figure 5.3 shows the same algorithm with the same parameter settings, but with p_1 and p_2 on the same point. In this case, the particle appropriately converges to that same point, and the standard deviation drops to zero, indicating that the particle has "come to rest" on this point.

Figures 5.4 and 5.5 show the same circumstances for the other two models of PSO-DR. For Model 2 $\phi = 1.6$, and for Model 3 $\phi = 1.2$, as per [132] and the previous chapter. When p_1 and p_2 are different, the Model 2 algorithm mean converges to the same point as the Model 1 algorithm, as shown in section 5.2, closer to p_1. The Model 3 algorithm mean converges to the midpoint of the two positions, again as previously asserted.

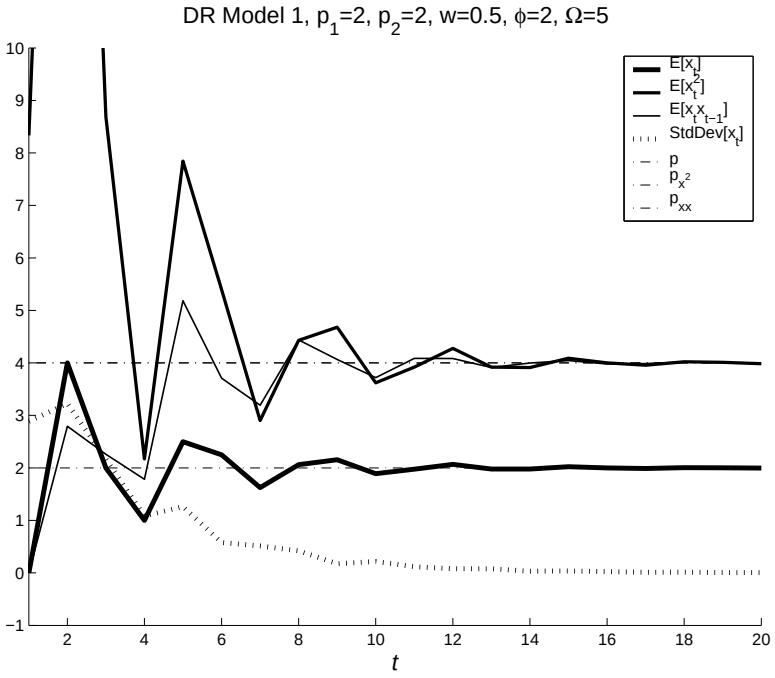
As with Model 1, the variance of Models 2 and 3 converges to a non-zero value when p_1 and p_2 are separate, and to zero when they are equal. From this we know that the default settings for all three algorithms are within the stable region. The difference among the algorithms is seen in the speed of convergence - Model 1 is the slowest to settle down, followed by Model 2. The Model 3 algorithm mean and variance both converge almost immediately under its default parameter settings. All three algorithms show complex oscillatory convergence to the mean with these settings, as predicted by the stability conditions in section 5.3.

Monotonic convergence can be seen when the w and ϕ parameters are appropriately set according to the corresponding conditions. This behaviour is shown for PSO-DR Models 1 and 2 in figure 5.6. For PSO-DR Model 1, where p_1 and p_2 are different, the standard deviation converges to a value lower than that seen in the oscillatory convergence, due to the decreased parameter values. The particle is converged, but is still jumping around the mean by a small amount - smaller than that seen above, when the converged-to standard deviation is at a higher value. As expected, the standard deviation for the second case, where p_1 and p_2 are equal, has converged to zero.

Unlike the oscillatory convergence settings, the mean under monotonic convergence conditions approaches the fixed point entirely from one direction, with no "searching" taking place on the other



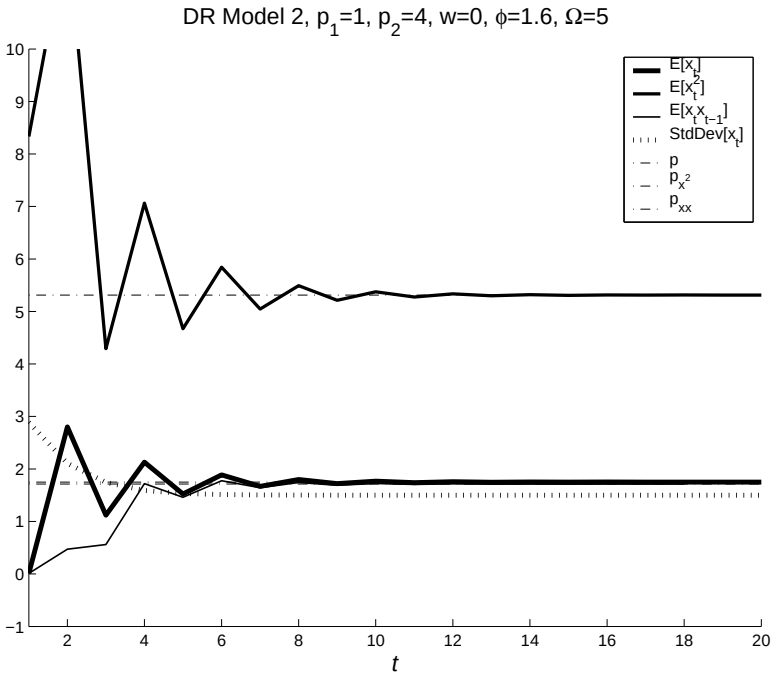
(a)



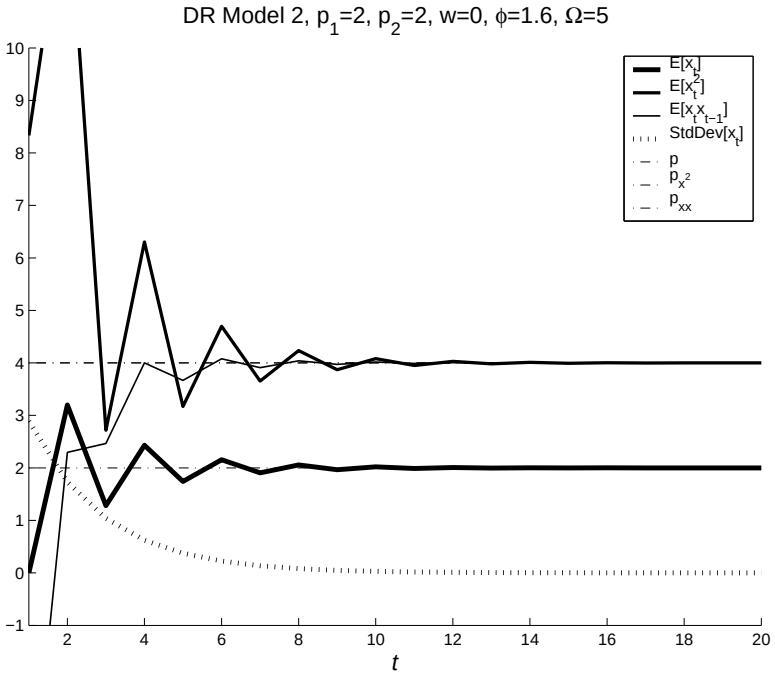
(b)

Figure 5.3: Selected plots of oscillatory convergence under PSO-DR Model 1

side. Monotonic convergence can be much faster than oscillatory under appropriate circumstances, as it basically takes steps toward the point of convergence, never overshooting and having to go back. Because this comes at the expense of an even exploration of the space about the mean, it is rare to see an optimization algorithm that relies wholly on this type of behaviour outside of locally-optimizing hillclimbers.



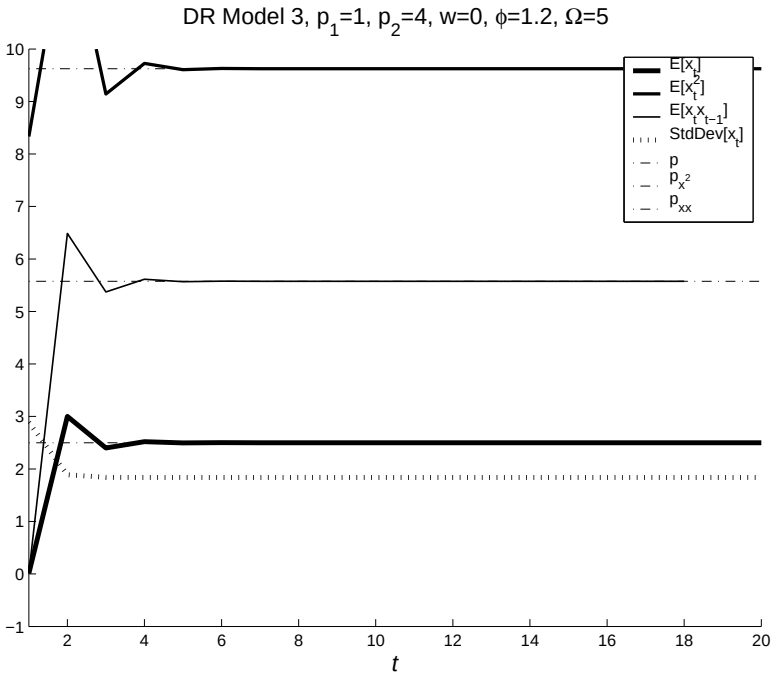
(a)



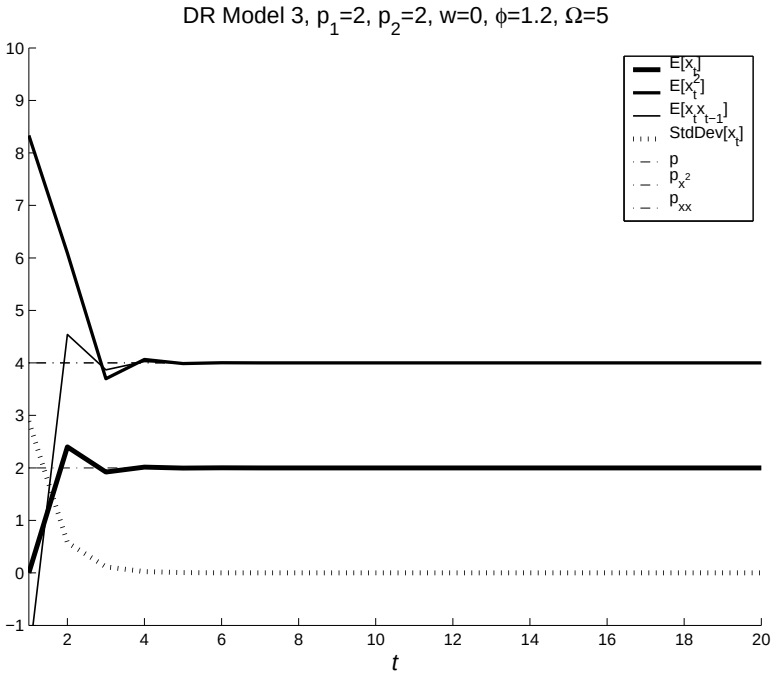
(b)

Figure 5.4: Selected plots of oscillatory convergence under PSO-DR Model 2

In the same way that parameter settings dictate whether convergence is monotonic or oscillatory, settings outside of the conditions for convergence can be selected to push the system to monotonic or oscillatory *divergence*. Figure 5.7 shows these behaviours under PSO-DR Models 2 and 3. When the value for ϕ is below the minimum of the stability range we can see the system diverging monotonically on one side of the mean, and when ϕ is greater than the upper bound of stability we see oscillatory



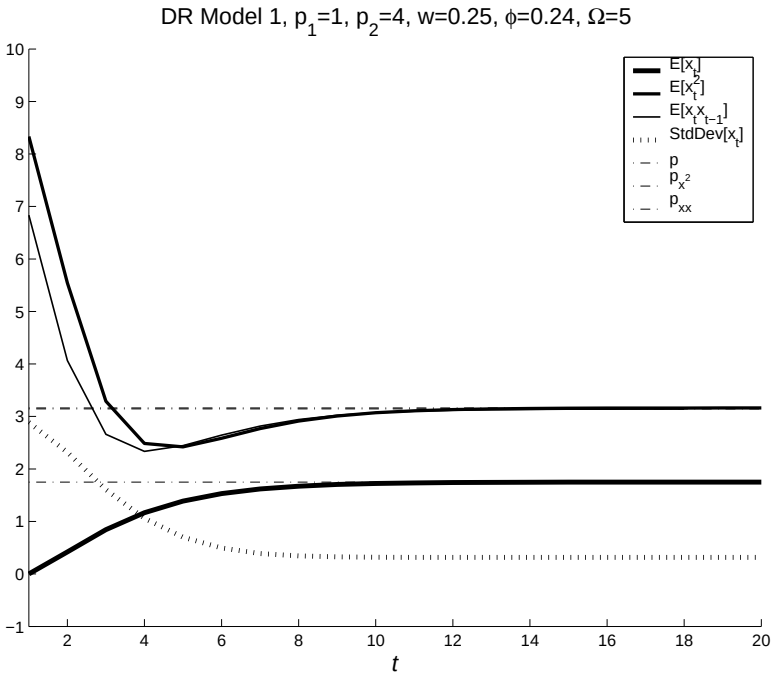
(a)



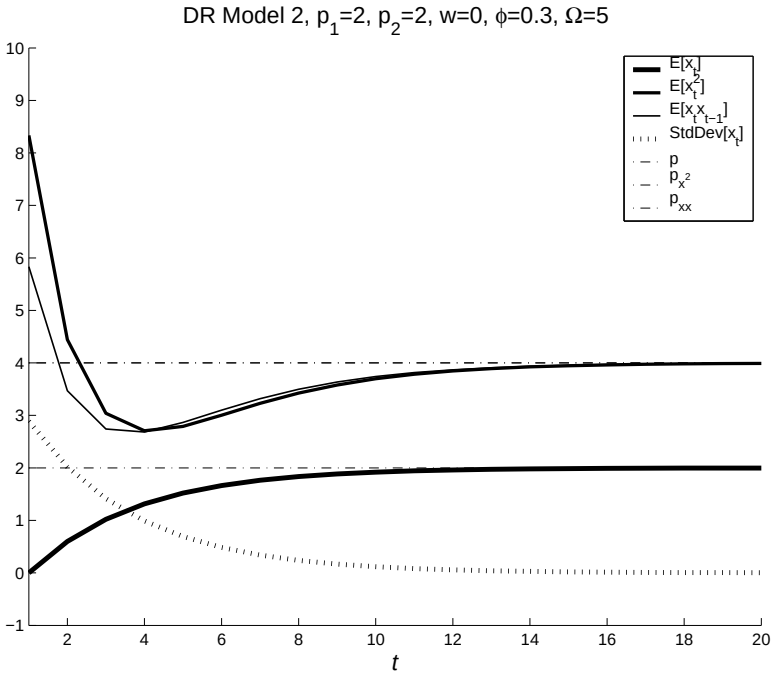
(b)

Figure 5.5: Selected plots of oscillatory convergence under PSO-DR Model 3

divergence. In both these cases the standard deviation increases without any limit as the particle gets further and further from the mean.



(a)

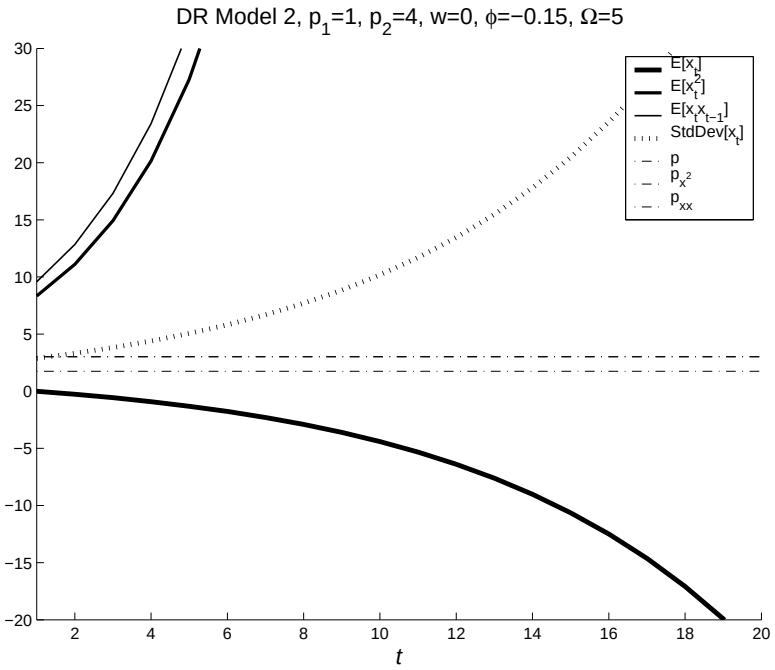


(b)

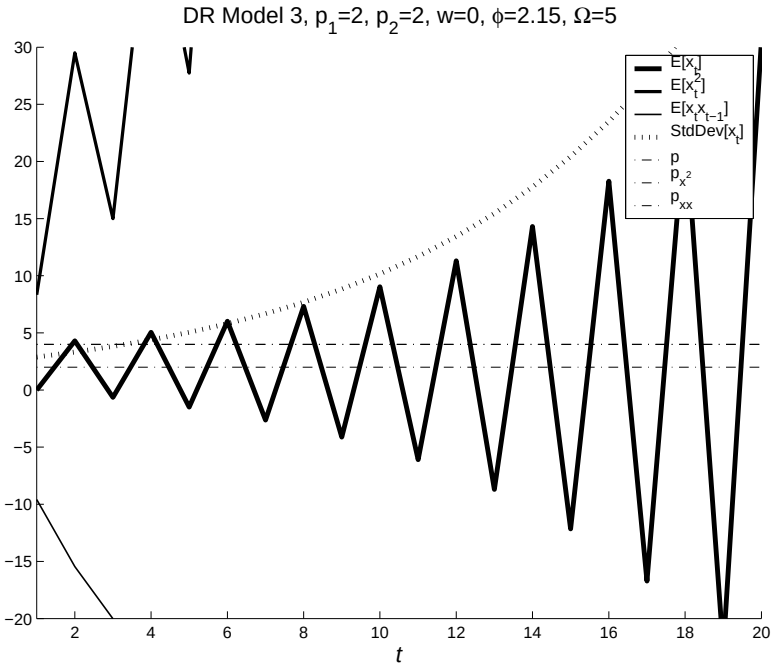
Figure 5.6: Selected plots of monotonic convergence

5.6 Discussion

As mentioned above, it can be seen by comparing figures 4.1 and 4.2 from the previous chapter with figures 5.1 and 5.2 that the empirically-determined performance of all three models of PSO-DR is best when the w and ϕ parameters fall within the analytically derived stability region. This supports the reason for deriving these regions – the ability of the swarm to find and exploit the optima of a problem



(a)



(b)

Figure 5.7: Selected plots of monotonic and oscillatory divergence

space is tied to the ability of its component particles to converge to a fixed point. When we know what parameter settings enable this convergence, we can select them accordingly so that we can reasonably expect optimization of the landscape to take place. Of course, selecting the ideal parameters from within this region of known stability is another undertaking.

Unlike the SPSO-equivalent algorithm analyzed by Poli, we saw that the regions of stability for

the mean and the variance of a PSO-DR algorithm are identical, meaning that there is no combination of w and ϕ parameters within this region that will not cause the variance to converge along with the mean. Whether or not this is beneficial to the performance of the algorithm is debatable. The ability to guarantee the eventual convergence of both the mean and the variance simultaneously potentially offers better optimization of individual minima than that of a swarm without this ability. While the SPSO algorithm with default settings sits quite near the edge of its region of order-2 stability stability[77], the default settings for the PSO-DR algorithms are comfortably within theirs.

This may explain the poor performance of SPSO on highly complex problems with deep, narrow minima, such as Rastrigin - problems on which the PSO-DR algorithms return much better results. Taking into account the analysis above, a possible explanation for this is that PSO-DR algorithms are able to reduce their variance much more quickly than SPSO, avoiding the surrounding local minima of the landscape, and restricting the area of exploration of the particle under stagnation to the single basin of attraction much more quickly than SPSO. Perhaps this slower variance convergence of SPSO dooms its constituent particles to continually find improved minima in basins of attraction outside the one currently being exploited, making optimization of the individual peaks more difficult.

On the other hand, fast convergence can be a drawback when the swarm has restricted itself to a single minima and does not possess the means to escape associated with a larger and/or diverging variance. SPSO's non-identical order-1 and order-2 stability regions allow for parameter settings that keep a particle converged to a mean, but allow for a diverging variance, something that is not possible with a PSO-DR algorithm. To this point there has been no published work exploring the benefits and drawbacks of these varying capabilities, but a swarm that could both quickly and reliably optimize a landscape while retaining the ability to escape minima would be very valuable. What is needed is a framework that allows for the adaptation of the swarm parameters that allows us to adjust the speed of convergence. This is the focus of the next chapter.

5.7 Conclusions

This chapter offered a statistical analysis of the various formulations of the PSO-DR algorithm using the method pioneered by Poli for application to standard PSO. Given the multiple models for PSO-DR defined in the previous chapter, it was necessary as a first step to generalize all of these into a single representative form. This general form was then further investigated.

The first two moments of the sampling distribution of the generalized PSO-DR particle update equation were derived using eigenvalue calculations, giving the stability conditions for each moment. These conditions/regions were found to be identical for the two moments. We determined the fixed points of the equations associated with these moments, finding that while the stability conditions, in terms of parameters w and ϕ, applied to all three models, the fixed points were dependent upon the

choice of recombinant term. As the recombinant term differs for PSO-DR Model 3 from Models 1 and 2, the fixed point will accordingly differ – specifically, the fixed point of Model 3 lies equidistant from the positions of both neighbors, while that of Models 1 and 2 lies nearer to the position of the neighbor with the best fitness. This can be verified by the update equations themselves, in which Models 1 and 2 contain an extra term representing this best found neighborhood position.

Next we determined the initial conditions of the system based on the initialization procedure of the algorithm. Along with the update equations for each model, these initialization conditions allowed us to observe the behaviour under various parameter settings. This clearly demonstrated the effects of these settings on the ability of a particle to converge or diverge, and to do either in a monotonic or an oscillatory manner.

Finally, it was suggested that while this proof of convergence in stable regions gives us a better understanding of the behaviour of the system, the algorithm would benefit from a method for selecting optimal parameters from within these regions.

Chapter 6

An Adaptive PSO-DRS Framework

6.1 Introduction

PSO has been demonstrated in this work and in the broader literature highlighted in Chapter 2 to be an effective optimization system within its problem domain, and to be strongly competitive against other optimizers within its field. The algorithm today is included in many standard comparison benchmarks for new approaches or variations to current methods.

In spite of this popularity and proven effectiveness, there are still many areas where PSO shows need or potential for improvement. Perhaps the most obvious is one that affects almost all optimization algorithms - parameter selection. There are nearly always fixed values within the update equations for a process that can influence performance and behaviour in ways that range from inconsequential to highly significant. Defining proper values for these parameters constitutes a large part of the effort in obtaining optimal performance on a given problem.

When manual tuning is used for optimizing the algorithm, tuning these parameters can only be done through trial-and-error, guesswork (informed or uninformed), or a combination of the two. Even when these techniques can be used to find optimal, or even ”good” parameter settings for the optimizer in question, these settings cannot necessarily be used for any problem apart from the one they were tuned to, or even for any point in the optimization process apart from the one at which they were selected. This is a particularly difficult aspect of manual tuning, as parameter selection under these circumstances is practically required to take place prior to the start of the optimizer.

Introducing a system for adapting the parameters to the particular problem and/or current point in the optimization process avoids the shortcomings of manual tuning, but still requires care and effort to put into place. Such a system could take parameter settings from standard ”default” ranges, adapting them over the course of the process to values that provide the best combination of space exploration and peak exploitation. These values need not eventually settle to fixed constants, but can be continually adjusted to provide excellent performance throughout the lifetime of the optimizer.

This chapter proposes several forms that such an adaptive system could take when applied to PSO-DRS. The system discussed is by no means the only possible form that an adaptive PSO could take, or necessarily even the best possible form, but functions as an exercise in defining the important points of adaptation from the algorithm and experimenting with various tuning adjustments to those parameters in an effort to obtain performance at least equal to that of the fixed-parameter, non-adaptive algorithm.

Much of the research and literature in global optimization is focused on finding variations to established algorithms that deliver better performance. This is an important aspect of any such field, and has been demonstrated in previous chapters. What is often lacking, however, is the examination of new *methods* for working with an algorithm in order to facilitate such research. Improved performance is taken as an extra benefit of the research presented here - the improvements to the algorithm that are sought come from the removal of the need for either manual parameter tuning or compromising problem-specific optimal settings to find general ones that may not be well-suited to various problems the optimizer could encounter. The definition of such generic methods is an important part of directing future research, and consolidating the techniques that can be used in finding variations helps to open up new ways of thinking about making improvements to the currently-practiced standards.

With this in mind, introducing adaptations to the PSO-DRS algorithm can also be beneficial with respect to allowing for better optimization of problem types on which the fixed-parameter form performs poorly. To that end, a new class of problems based on the Lennard-Jones potential[133] is used here in order to demonstrate how reconstituting the parameters of the swarm to the landscape being optimized can greatly improve performance.

6.2 Adaptable Parameters

There are two types of parameters that are common across almost all forms of the PSO algorithm: *swarm parameters* and *particle parameters*. Swarm parameters are those that are associated with the structure and behaviour of the swarm as a whole. Particle parameters are those associated with the positional update equations used by each particle to obtain a new velocity and/or position. These two levels of parameters are equivalent to the first two defined in Angeline's review of the field of adaptive evolutionary computation[134], *population-level* and *individual-level* parameters. In the same work, Angeline further defined a third level, *component-level*, which is concerned with the individual-level parameters of specific population members – this level of adaptation is possible here by altering parameter values of single particles separately from other members of the swarm, but for the purposes of this work only population-level and individual-level adaptations are required.

Both these types of parameters are clear points of adaptation for the optimizer. Adjustments to particle parameters influence the behaviour of individual particles by way of the various components of the update equation. Adjustments to swarm parameters on the other hand influence the behaviour of the

swarm as a whole with an indirect effect on the movement details of particles through their connectivity and relationships.

As explained above, the issue of choosing appropriate values for any of these parameters is complicated by the fact that a single set of parameter settings can show varying behaviour and/or performance on different types of problems. Unimodal problems benefit from quick convergence, as there are no local minima in which the swarm could become trapped. Quick convergence can often be obtained with the use of small step sizes on the particle level and high levels of connectivity on the swarm level. Multimodal problems, on the other hand, require slower convergence and more exploration to reduce the risk of the entire swarm settling into the inescapable basin of attraction of a suboptimal minima, which is often found in an optimizer combining large step sizes with a sparsely-connected swarm[90].

6.2.1 Particle Level

In the original formulation of the PSO update equations, there are either one or two fixed values, depending on implementation[1]. These are the values for the constant multipliers c_1 and c_2 that are applied to the random values obtained for weighting the personal best and neighborhood best components. c_1 and c_2 are most often set to the same value c.

The two most popular current formulations of PSO, the inertia weight (IW) algorithm[55] and the constricted algorithm[10], each add an additional fixed value. In the case of the IW equations this is the multiplier w which is applied to the previous velocity term, while the constricted equations apply the multiplier χ to the entire update equation. This gives two to three values that must be chosen prior to beginning the optimization process (although by using the constriction method the value for χ can be derived from the values of c_1 and c_2, or vice-versa). Stable ranges and combinations for these values have been analytically determined[10], but these only demonstrate the values for which oscillatory convergent behaviour can be obtained, not necessarily optimal ones.

The new form of PSO presented here in chapter 4, PSO-DR Model 3, or PSO-DRS, reduces the number of required parameters within the update equations down to just one, ϕ. This multiplier is applied to the distance between the recombinant point and the current particle position. While the reduced update equations appear more straightforward with the removal of balanced social and cognitive components, tuning this single parameter becomes the only means of affecting behaviour on this level and thus finding an appropriate value is vital for the issue of performance. The stable range that ensures convergence has been defined in the previous chapter as $\phi = [0.0..2.0]$, with the range ensuring oscillatory convergence as $\phi = [1.0..2.0]$, but as before, this only demonstrates values which allow the swarm to converge with no claims made as to performance.

While these stable regions obtained for both standard PSO (SPSO) and PSO-DRS allow parameter values to be chosen that will ensure convergence, they do not reveal which of these values will lead

to superior, or even acceptable optimization performance. It is very possible for a value to be chosen from within the stable range that leads to poor performance, e.g. in the circumstance where the swarm converges, but to a very suboptimal point within only a few iterations. Knowing the stable regions for an algorithm are beneficial from a performance perspective in narrowing the choices for these values, but finding the actual optimal parameters within these regions is where adaptive processes come into play.

6.2.2 Swarm Level

There are two swarm parameters that are common to all types of PSO:

1. *size*, and

2. *connectivity*.

Size represents the number of individual particles in a swarm. The effects of this setting on swarm behaviour are easily explained, but this is not necessarily the case when it comes to performance. Small swarms react to new information quickly when using a sequential update process where each particle's velocity and position is updated in turn. Given two swarms, the smaller will update every particle once and at least one particle more than once in the same number of function evaluations that it takes the other to update all of its composite particles a single time.

For example: new information is introduced to a globally-connected swarm by means of a new global best. Assuming stagnation of that best position, over sixty function evaluations every particle in a thirty-particle swarm would take two steps that were informed by this position, while particles in a twenty-particle swarm would take three. Given the converging nature of the algorithm, overall the smaller swarm would move more toward the new global best position in that period than the larger one.

Balancing is required for this parameter in order to prevent premature convergence. There can be no single number of particles that is optimal at each individual point in the optimization, given the varying need for fast or slow convergence depending on the current state of the swarm. An 'optimal' choice for the number of particles required for fast convergence has been proposed based on an analysis of the relationship between convergence and problem dimensionality[90], but this is determined as the minimum population size for convergence, without accounting for relative performance. As such, using this single value in all cases has multiple drawbacks. On a multimodal problem, a swarm consisting of the minimum number of particles will converge to a potentially local optimum too quickly for new positions to be found and evaluated. Conversely, a swarm using a value too much larger than this minimum will still converge to the same position, but may not do so within a reasonable number of function evaluations. As most performance measures involve either minimizing the number of function evaluations required to find a minimum or finding the best possible minimum within a limited number of evaluations, finding an acceptable compromise between swarm size and performance is an important part of the tuning process.

Connectivity describes the communication topology between particles. While specific topologies are often referred to in colloquial terms (ring, square/von Neumann, global)[12], any one of them has an associated connectivity value that refers to the size of any given particle's neighborhood. A swarm using a ring topology, for example, would have a connectivity value of 2, as each particle is connected to only 2 neighbors, while a 20-particle swarm with a global topology would have a connectivity value of 19, representing its connection to every other particle in the swarm.

Connectivity influences the speed with which information is propagated throughout the swarm. A new global best position that is discovered by a particle in a fully connected swarm will be available to every other particle at its next evaluation, while the position will only become available to a particle in a 2-connected swarm when one of the particle's two neighbors discovers it. As particles generally move toward a newly introduced position rather than directly onto it, this results in a wider area of exploration for swarms with lower connectivity as particles follow an optima into an area and pass information about their surroundings - but not necessarily the optima itself - to their neighbors.

Some PSO variants use swarm topologies that are asymmetric, i.e. particles do not necessarily have the same number of connections. An example is the Tribes system[11], where each particle has a high number of connections within a small subswarm, and a limited number of particles in each subswarm also have connections to particles within other subswarms. The connectivity of any graph/swarm is given as the number of connections to the most-connected node/particle, so a single value can still be used to describe this feature. It also means that we cannot assume that for a K-connected swarm, every particle has K connections - all that it tells us is that *at least one* particle has K connections.

For the sake of keeping the approach taken here relatively simple and practical for general use, only K-regular topologies are used in the following investigation. If the K-connectivity value is reported as x, all particles in the swarm are connected to x other particles. This prevents situations from arising where unique topologies are created by circumstances of the adaptive optimization that are specific only to that particular problem and point in the process. While such topologies may be highly tuned to the problem at hand and potentially produce excellent performance, it is more beneficial in the interest of replication to restrict the range of available topologies to those that can be defined and understood prior to use.

6.3 Convergence Rate as an Adaptive Measure

It is important when adapting these three parameters to avoid doing so in an arbitrary way. Increasing the size of the swarm by 1 particle is only meaningful if we know how this change affects the optimization behaviour and capability of the algorithm. Also important is an understanding of how the values of the different parameters affect the swarm in different ways - for example, the relative effect of the number of particles versus the set value for ϕ.

The speed of convergence to a minimum was chosen as a measure of the effect of parameter adjust-

ment, as this phenomenon is one of the principle behaviours of a functional optimization algorithm. The simple sphere problem f_1 was used to determine the convergence rate of a swarm from random uniform initialization within the basin of attraction down to the minimum. With a single optimum available to the swarm along with a smooth slope leading to that optimum, the swarm's ability to converge will not be hindered by characteristics of the landscape, and thus can be expected to be consistent across multiple runs using the same parameter settings.

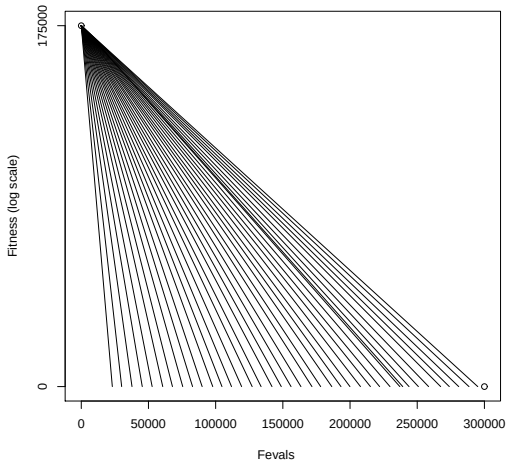
A steady logarithmic convergence is the norm for a swarm optimization algorithm on a smooth unimodal problem such as f_1. With this in mind, a convergence factor was calculated for multiple settings of each of the three adaptable parameters. This factor was defined as the negative of the simple gradient ($-\frac{\Delta y}{\Delta x}$) of the straight line that is seen on convergence plots of the mean fitness over 500 runs for the sphere problem when a logarithmic fitness scale is used, as shown in Figs 6.1(a), 6.1(b), and 6.1(c) for the 30-dimensional sphere problem. This is fully expressed as:

$$\gamma = \left\langle -\frac{\log_{10} f(T) - log_{10} f(1)}{T-1} \right\rangle \tag{6.1}$$

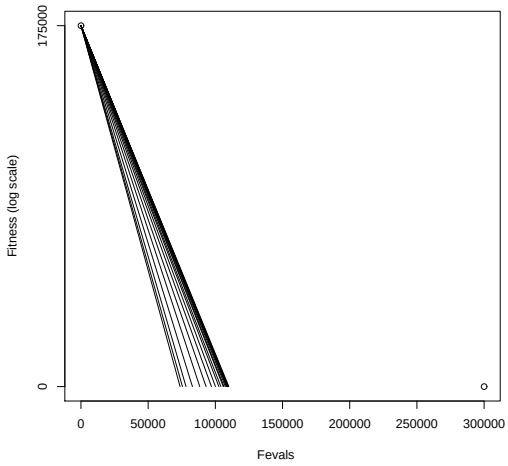
where f is the fitness after a specific number of function evaluations, and T is the number of evaluations necessary for the algorithm to find the global optimum to within $f(T) = 10^{-15}$. Using region-scaling for initialization in order to prevent an initial value fixed on the origin, the expected value of each individual component in the particle position will be either -75.0 or 75.0 (region scaling will initialize uniformly in the top or bottom quarter of each dimension; the search space for sphere is [-100..100]). Taking into account the exponential fitness function for the sphere function, the associated particle fitness at this expected value can be fixed at 175000. Using this for the value of $f(1)$, a convergence rate was obtained for each parameter setting that is dependent only on the number of function evaluations T. Using this method, higher, i.e. faster, rates of convergence were derived from parameter settings where fewer function evaluations were required to find the minimum of the search space.

These convergence rates were determined using a 30-dimensional search space. Convergence rates for problems from 5 to 50 dimensions were tested for all three parameters and are shown in Figs 6.2, 6.3, and 6.4. While the rates were higher for lower dimensions and lower for higher dimensions, the shape of the curve was generally independent of dimensionality, and the rates were derived from the 30-dimensional problem, which fell nearest the average levels. Further, the 30-dimensional problem equals or exceeds most of the benchmarks used in this work, apart from the most complex Lennard-Jones problems. This ensures that the value of the adapted parameter(s) will never be reduced below that associated with the fastest rate of convergence.

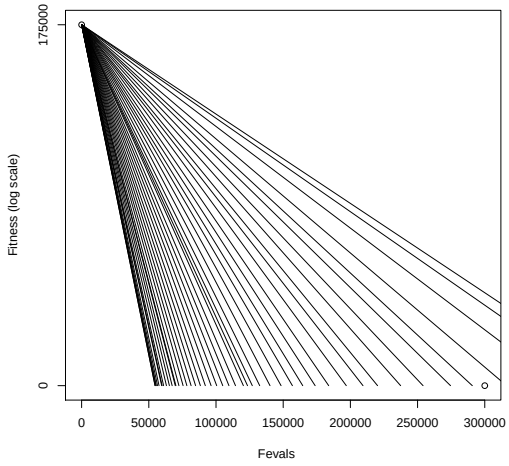
The convergence rates for the 30-dimensional problem are appropriate for the problems used here - benchmarks containing numerous higher-dimensional problem spaces may require them to be recalcu-



(a) *NP*



(b) *K*



(c) ϕ

Figure 6.1: Convergence factors for adapted parameter values on a 30-dim f_1 for PSO-DRS.

lated. This is a relatively quick process of determining the number of function evaluations required to solve f_1 in the desired dimensionality for various values of the tunable parameters.

The relation of these gradients / rates of convergence for the 30-dimensional problem space to various settings for each parameter are shown in Figs 6.1(a), 6.1(b), and 6.1(c). These figures each demonstrate a clear relationship between the setting of a parameter and the convergence speed of the swarm. Each rate was determined for multiple values of a single parameter, with the other two fixed to their default settings ($NP = 50$, $K = 2$, and $\phi = 1.2$). Zoomed views of each of the curves for the 30-dimensional problems are shown in Figs 6.5, 6.6, and 6.7.

Using this information, it is now possible to control how quickly the swarm converges with a high degree of accuracy by changing individual parameters. Reducing the convergence factor by 10% corre-

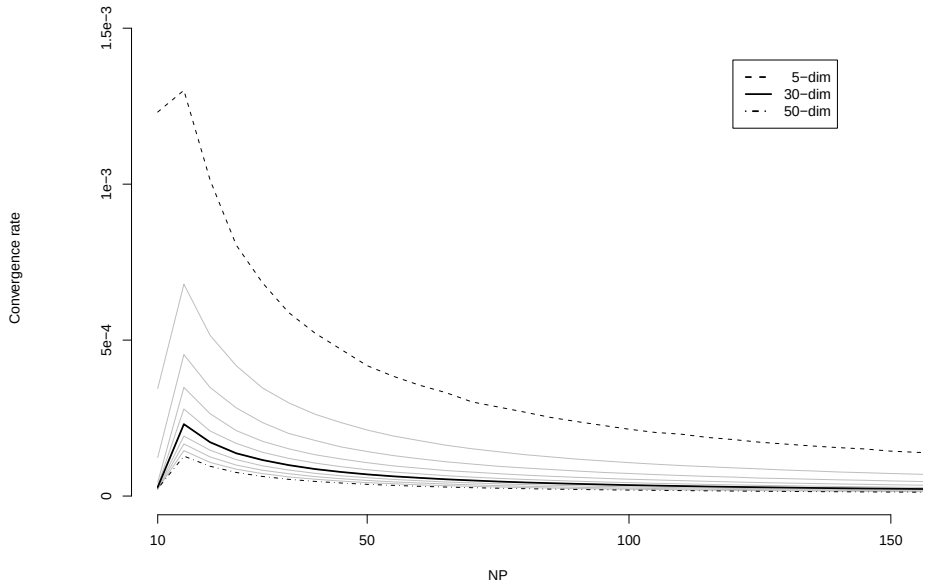


Figure 6.2: Convergence rates in multiple dimensions for values of *NP* on f_1, where *K*=2, ϕ=1.2

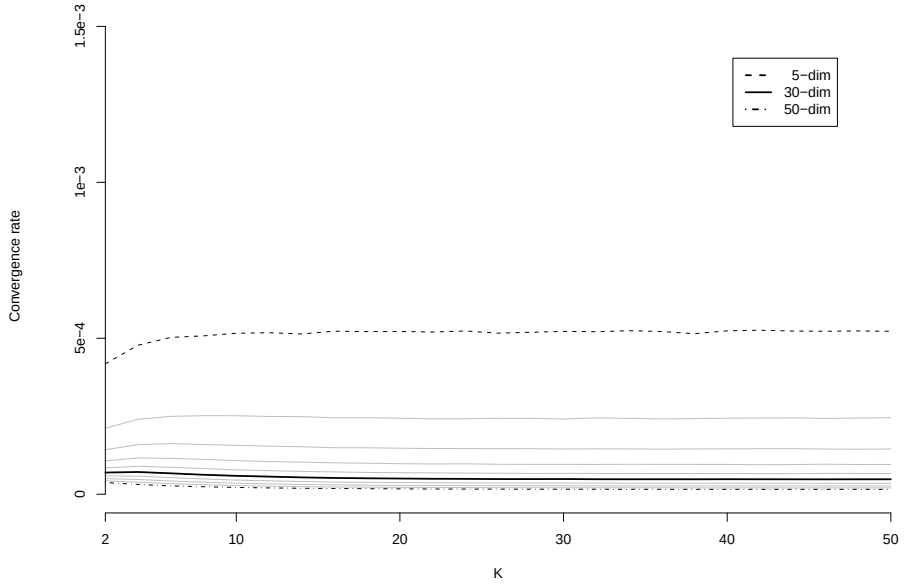


Figure 6.3: Convergence rates in multiple dimensions for values of *K* on f_1, where *NP*=50, ϕ=1.2

sponds to an increase in the value of the adapted parameter to the value associated with the new rate of convergence, and thus an increase in the number of function evaluations necessary for the swarm to find the minimum of a basin of attraction. In the same way, increasing the convergence factor resulted in a reduction of the adapted parameter, and faster convergence to a minimum.

As can be seen from the comparative plots of convergence factors vs parameter settings, the three

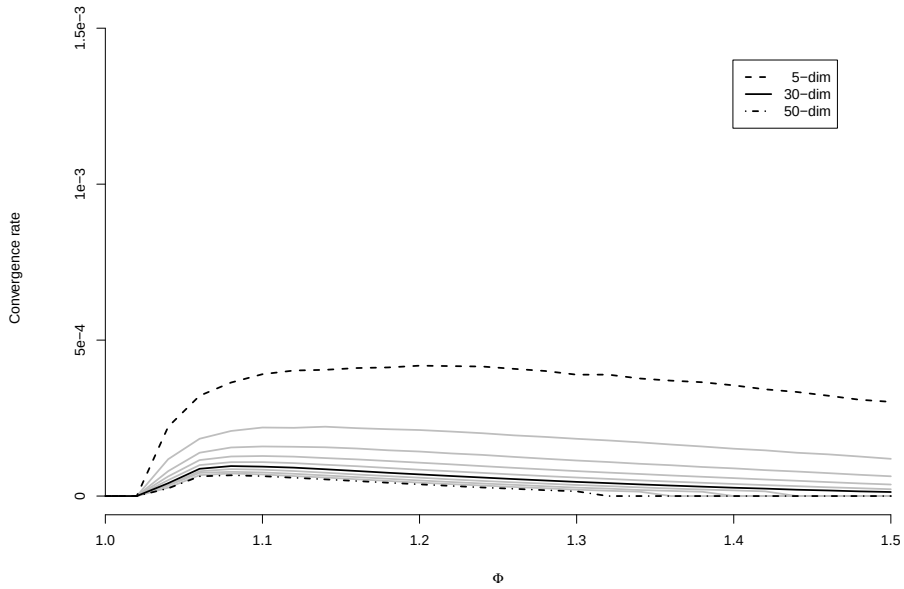


Figure 6.4: Convergence rates in multiple dimensions for values of ϕ on f_1, where *NP*=50, *K*=2

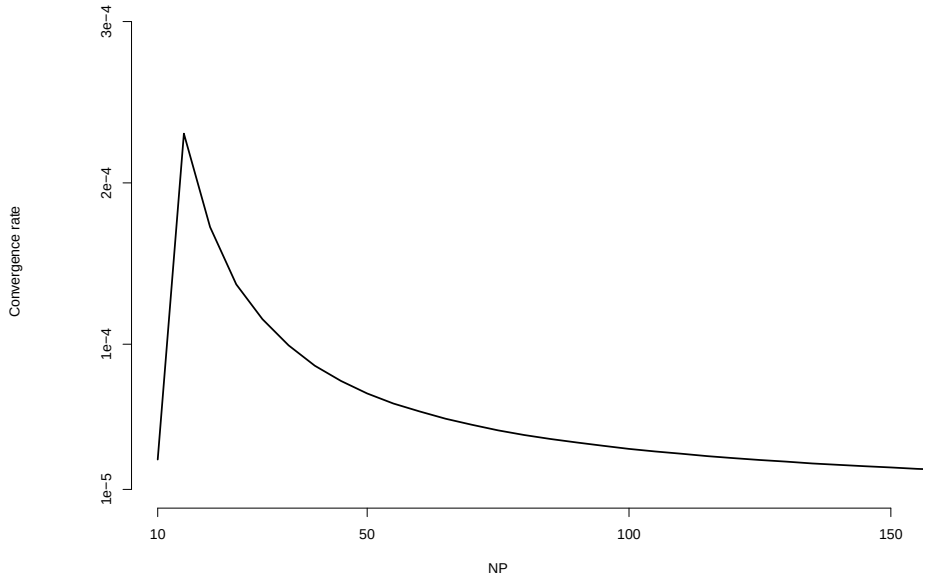


Figure 6.5: Convergence rates in 30 dimensions for values of *NP* on f_1, where *K*=2, ϕ=1.2

adaptable parameters do not all influence the convergence speed equally. The range of the convergence factor for various parameter settings for the number of particles, corresponding to approximately 25,000 to 300,000 required function evaluations, was nearly ten times the size of the range associated with various settings for the swarm's *K*-connectivity, which only covered approximately 75,000 to 110,000 function evaluations.

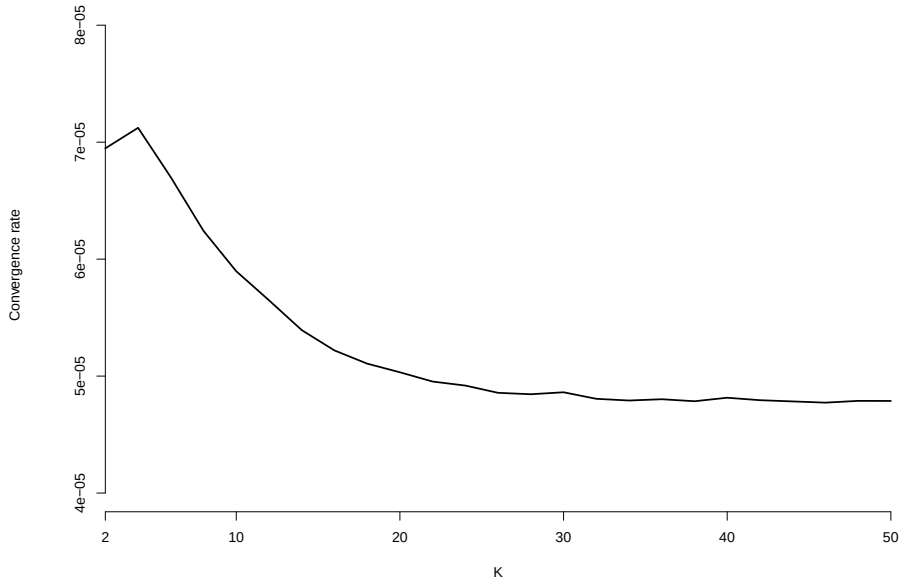


Figure 6.6: Convergence rates in 30 dimensions for values of K on f_1, where NP=50, ϕ=1.2

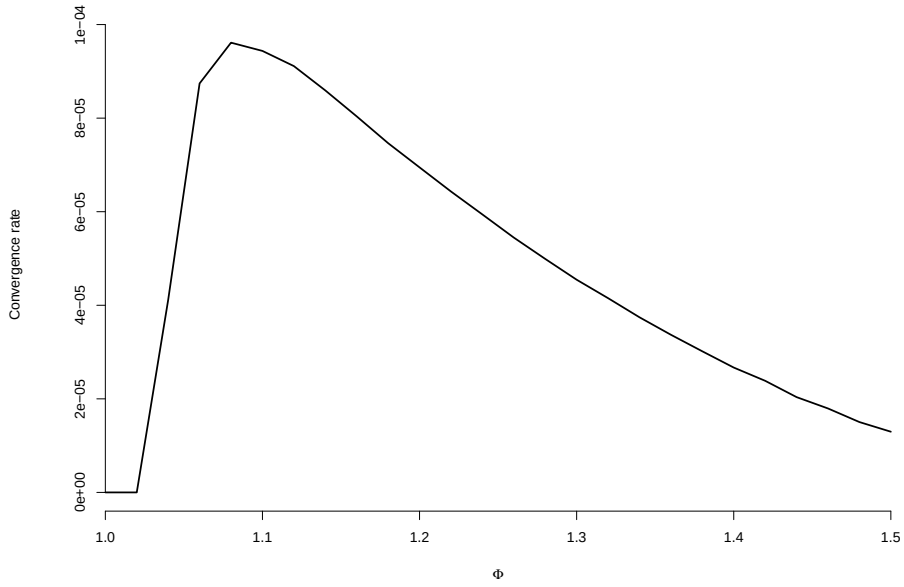


Figure 6.7: Convergence rates in 30 dimensions for values of ϕ on f_1, where NP=50, K=2

Seeing how the settings for the parameters affected convergence allowed for a range of validity to be set for each parameter. For example, the change in the rate of convergence for increased swarm sizes shown in Fig 6.2 indicates that the largest differences in convergence speed take place when the number of particles is low, and flattens out for swarms with more than approximately 150 particles. Using this information, the range for the allowable number of particles was restricted to fall between the point of

fastest convergence and the approximate point of slowest convergence, [15..150]. Similarly, the range for ϕ was restricted to [1.08..1.5]. No restrictions were added to the range of adaptation for K, as the range of values for this parameter is already bounded between a minimum connectivity of 2 and a maximum connectivity of the number of particles in the swarm.

6.4 Adaptation Rules

The three types of adaptations discussed here differ from one another by varying amounts. Adjusting the number of particles in the swarm can be seen as a similar process to increasing or decreasing the connectivity of the communication topology. While the change to the swarm structure and the effects on behaviour are quite dissimilar, the actual process is accomplished in the same way by adding or removing a component of the swarm - either a particle or a link between two particles.

Adjusting the value of ϕ is a rather different process. Instead of completely adding or removing a swarm component, this adaptation involves an adjustment to a pre-existing value. Rather than altering the form of the entire swarm, only the update equation for each individual particle is changed. Although this may seem minor in comparison to the other two adaptations, the value of ϕ is in fact extremely important to optimization performance, as it has been shown in the previous chapter to have a direct effect on the stability of the swarm and its ability to converge.

Despite the differences in the processes for the swarm-level and the particle-level adaptations, they can be broken down into the same general form. Only two adjustments can be made to any of these three parameters: an increase in value, or a decrease in value. ϕ can be adjusted by a real amount, while swarm size and connectivity are adjusted by integer amounts.

Three general adaptive rules were tested in this work; each was applied to the ϕ, size, and connectivity adaptations. The first of these was:

Adaptive Rule 1a

if *best found fitness has improved in the last iteration;*

then

| increase the value of the parameter;

else

└ decrease the value of the parameter;

and the opposite:

Adaptive Rule 1b

if *best found fitness has improved in the last iteration;*

then
| decrease the value of the parameter;

else
└ increase the value of the parameter;

These rules were intended as demonstrations of how alterations to each parameter affect swarm behaviour and performance. Rule 1a is used to show how an increase to the value of any of the three parameters effects a reduction in the speed of convergence: a larger swarm size adds diversity; a larger value for ϕ gives a larger size for the step taken by a particle; a larger K-connectivity for the swarm gives a larger potential search space for each particle (see section 6.5.3 for an explanation of this effect in a PSO-DR swarm). A swarm operating under rule 1a will slow its rate of convergence every time an improved position is found, resulting in better exploration of the search space, but slower exploitation of discovered minima. See previous section 6.3 for a complete explanation of this effect.

The opposite effects occur when the values are decreased, as in rule 1b - reducing the values of any of the parameters will speed up the rate of convergence of the swarm. Taking this into account, a swarm operating under rule 1b will increase its rate of convergence at every iteration where it has improved. In practice this means that once a swarm has started converging toward an attractor, it will continually increase the speed of this convergence until the minima is reached. Conversely to rule 1a, this means that rule 1b-adaptive swarms will be worse than a fixed-parameter swarm at exploring the search space, but better at exploiting individual minima.

The second adaptive rule tested was:

Adaptive Rule 2

if *best found fitness has improved in the last iteration;*

then
| do the previous adaptation again;

else
└ do the other adaptation;

where the possible adaptations are the increase or decrease in value to the parameter in question. This adaptation was applied after every iteration, and makes the assumption that if the previous adaptation has improved swarm performance, applying it again will be beneficial as well. If, however, the previous adaptation failed to improve swarm performance, the opposite measure should be applied.

Rule 2 is a somewhat more interactive approach to adaptiveness in the swarm - rather than simply increasing or decreasing the speed of convergence based on performance, this rule looks at historical data

to determine how to adapt the swarm. In practice, this means that a diverging swarm should continue to diverge as long as it continues to discover new attractors, as in rule 1a, and once convergence to one of these attractors begins, it will increase its speed to quickly optimize the minima, as in rule 1b.

The third adaptive rule tested was even more intricately tied to swarm performance than rule 2. While rules 1 and 2 were applied at a fixed point at the end of every iteration, the third rule adapted swarm properties based on a measure related to function evaluations:

Adaptive Rule 3

if *best found fitness has improved;*

then

  if *number of function evaluations between improvements has decreased;*

  then

    do the previous adaptation again;

  else

    do the other adaptation;

This immediate adaptation in response to the current swarm status makes this rule much more dynamic than the previous two, which use statically defined update points. It also avoids the need for setting a separate parameter, the number of iterations between adaptations (set to 1 for the other rules).

The selection of these rules is not intended as an exhaustive examination of methods by which adaptations can be applied to the parameters, but neither were the rules chosen arbitrarily. The rules were designed for simplicity, both to allow for easy replication, and to make the resulting performance and behaviour straightforward and comparable between the different approaches. The simple increases and decreases to the convergence rate, alongside the rules for applying them, were chosen specifically to prevent complex, unique swarm configurations from arising, as is the case in many adaptive optimizers. Tribes[11] in particular, while commendable for being arguably the only completely parameter-free particle swarm optimizer in the literature, has demonstrated very good performance on many benchmarks, but at the cost of extremely complex adaptive rules that have stymied replication and greatly limited general adoption.

The two variations on rule 1 show performance and behavioural patterns for the algorithm when the convergence rate is increased or decreased to the maximum or minimum possible values and held there. This provides a form of ‘bounds’ of potential performance / behaviour for the swarm when under adaptation. Rule 2 keeps the value somewhere between these two extremes, but only takes into account a single measure: whether or not the swarm has improved its best found position. Rule 3 expands on this, adapting parameters based not just on whether or not the swarm is improving, but also the rate at which the improvements are taking place.

These rules are incomplete in terms of presenting a full examination of possible applications of swarm adaptation (as most or all finite rulesets would be); with that said, the aim of this work is not to find a "best" rule for adapting the PSO-DRS algorithm, but to demonstrate the features and abilities of such a swarm when adapted. For those purposes, the rules defined here represent a good method for examining four very different approaches to adaptation.

6.5 Results for Individual Points of Adaptation

For the sake of brevity, the results tables shown in this section show only the significance of the mean performance of the adapted-parameter PSO-DRS when compared to a fixed-parameter PSO-DRS with default parameter settings of $NP = 50$, $K = 2$, and $\phi = 1.2$, in tables 6.1–6.4. The columns of these tables should not be used to compare adaptations or rules, but neither is this a goal of the work – what is being sought here is a method for improving the basic algorithm, rather than empirically determining the single best improvement. Full results tables for all combinations of algorithms, rules, and adaptive parameters can be found in Appendix A.

6.5.1 Adapting ϕ

The need for an adaptive form of PSO was demonstrated previously in table 4.1 on page 74. No single value for ϕ returns optimal results for all problems: the optimal value can range from 1.05 for f_1, the unimodal sphere problem, to 1.35 for f_{14}, one of the problems in the Shekel class. In the static non-adaptive formulation of the PSO-DRS algorithm, this value for ϕ is held constant throughout the whole of the optimization process, which prevents more appropriate values from being used in different phases.

Variations in optimal step size can be linked not only to the phase of the optimization process, but also to the configuration of the problem landscape. Smaller values of ϕ translate to smaller step sizes, larger values to larger step sizes. Multimodal landscapes require a larger step size to allow the swarm to jump over or out of local minima, while unimodal problems contain no such dangers. Under these circumstances a smaller step size would allow for faster optimization of a single peak, hence the lower values of phi for single-minima problems such as f_1 and f_2, and the simple multimodal f_{10}. Similarly, problems with a multiple local minima are shown to be best optimized by a swarm using a larger value for ϕ, especially when those minima are broadly spaced, as with f_4, f_{13}, and f_{14}.

Results for the adaptation of ϕ on the unimodal problems broadly support this reasoning. Rule 1a (table 6.1) shows significantly poorer performance than the fixed-parameter PSO-DRS for two of the three unimodal problems, as would be expected when the value of ϕ is increased as the swarm improves its best found position. That of rule 1b (table 6.2) is statistically equal to the fixed swarm on all unimodals, again expected due to the increasing rate of convergence as a peak is exploited under this rule. Rules 2 (table 6.3) and 3 (table 6.4) show similar equivalent performance, with rule 3 giving a

particularly good result on the deceptive and difficult Rosenbrock problem (f_3).

The results for the simple and difficult multimodals are however somewhat poorer for the adaptive swarms when compared to the fixed-parameter swarm. Compared to each other, however, the adaptive rules perform as expected, with the diverging-on-improvement swarm operating under rule 1a outperforming the converging-on-improvement rule 1b swarm on five of the six problems. Especially notable is the behaviour of rule 1b on f_4, where it was unable to escape a local optima positioned within the swarm initialization range on any of its 50 runs. Rules 2 and 3 performed better than one or both of the rule 1 variations on all of these problems.

Swarm behaviour for the ϕ adaptation followed predicted patterns almost exactly. The rule 1a-adapted swarm decreased the rate of convergence over time, and the dip statistic remained high on multimodal problems throughout the optimization process, indicating a dispersed, multimodal distribution of the particles across multiple peaks. Conversely, the rule 1b-adapted swarm increased the rate of convergence, and accordingly drove the dip statistic below the threshold of unimodality after the discovery phase, indicating convergence to a single optimum. Figs 6.8(a)–6.9(b) show the effect that these opposite approaches to adaptation of the convergence speed had on the modality of the swarm distribution on a complex multimodal landscape over 10 runs.

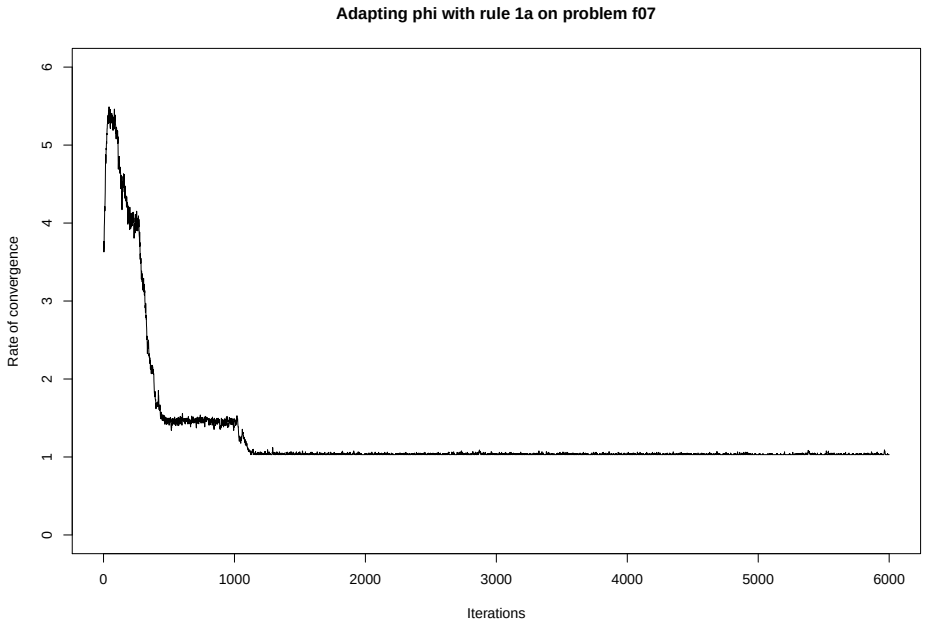
Rules 2 and 3 provided similar swarm behaviour to each other, with rule 3 maintaining a very slightly higher rate of convergence on average, and hence a very slightly lower dip statistic value overall. This similar behaviour led to very similar performance results.

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
f_1	–	*	*	*	–	*	–
f_2	–	–	–	–	–	–	–
f_3	–	+	–	–	–	*	*
f_4	–	–	–	–	–	–	–
f_5	–	–	–	–	–	–	–
f_6	–	*	*	–	–	–	–
f_7	*	*	*	*	*	*	*
f_8	*	*	*	*	*	–	–
f_9	*	*	*	*	–	–	–
f_{10}	*	–	*	*	*	–	*
f_{11}	*	*	*	*	*	*	*
f_{12}	*	–	–	–	*	–	–
f_{13}	*	–	*	–	*	–	–
f_{14}	*	–	–	*	*	–	–

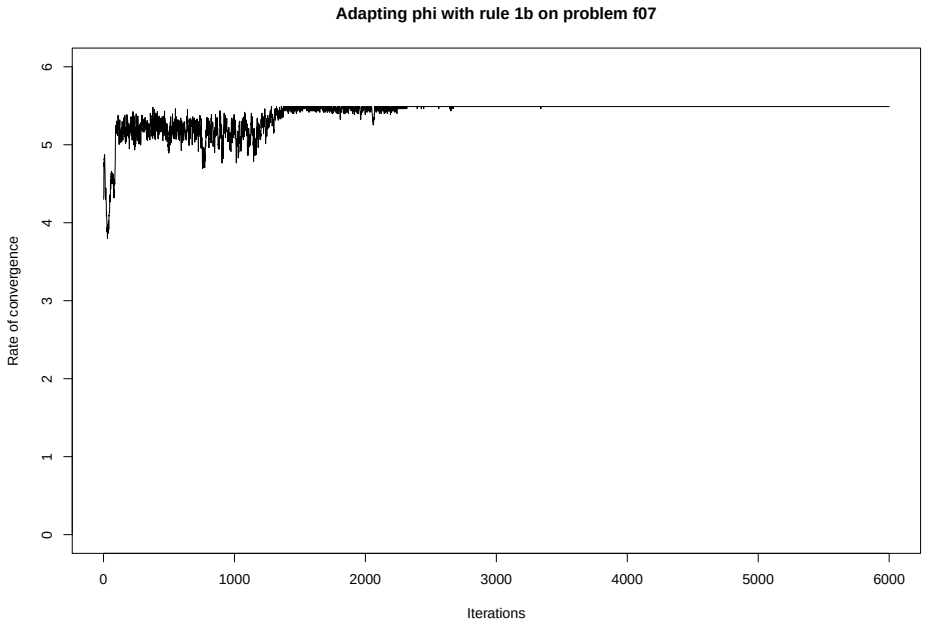
Table 6.1: Significance of results for *rule 1a-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* where + = better, * = equivalent, – = worse

6.5.2 Adapting *NP*

The curves in Fig 6.2 shows the strong effect that the number of particles comprising a swarm has on the speed at which it converges to an optimum. Swarms with fewer than 15 particles drop off quickly in the rate of convergence on a 30-dimensional sphere problem, while increases over approximately

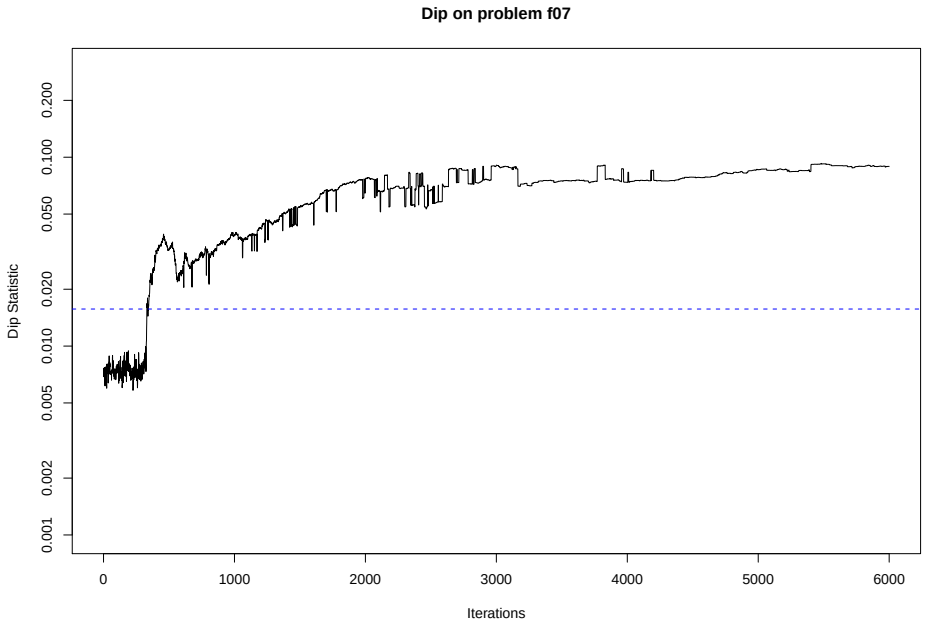


(a) Rule 1a

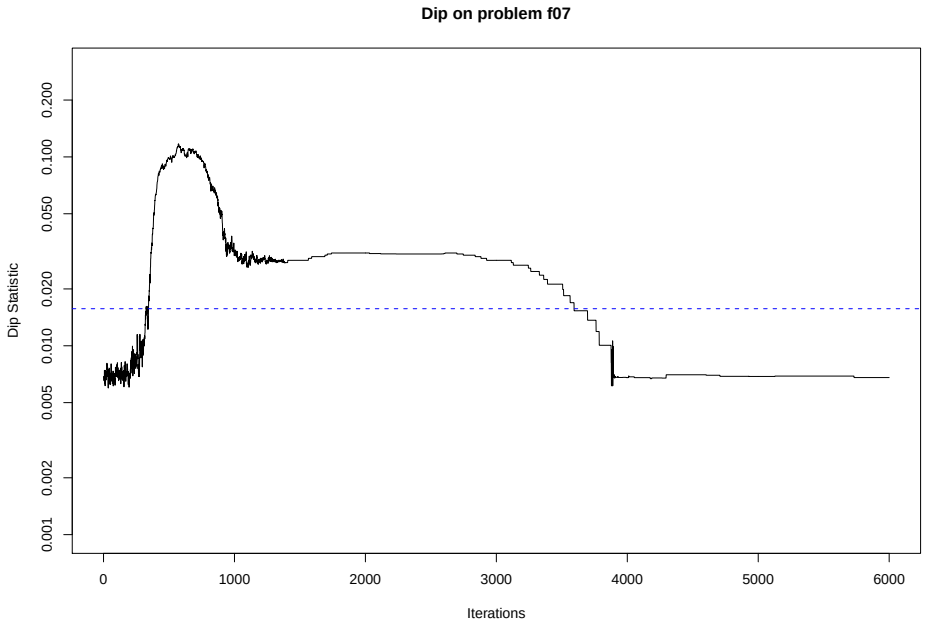


(b) Rule 1b

Figure 6.8: Values of ϕ on f_7 under adaptation



(a) Rule 1a



(b) Rule 1b

Figure 6.9: Dip values for swarms adapting ϕ on f_7

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
f_1	*	*	*	*	*	*	*
f_2	*	–	*	*	*	*	*
f_3	*	*	*	*	–	*	–
f_4	–	–	–	–	–	–	–
f_5	–	*	–	–	–	–	–
f_6	*	*	–	*	–	–	–
f_7	*	*	*	–	–	*	–
f_8	*	*	*	*	*	*	*
f_9	*	*	*	*	*	*	–
f_{10}	*	*	*	*	*	*	*
f_{11}	*	*	–	*	*	*	–
f_{12}	*	–	*	*	*	*	*
f_{13}	*	–	*	*	*	–	*
f_{14}	*	–	–	*	*	–	*

Table 6.2: Significance of results for *rule 1b-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* where + = better, * = equivalent, – = worse

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
f_1	–	–	*	–	*	*	*
f_2	–	–	*	–	–	*	*
f_3	–	–	*	*	*	*	*
f_4	–	–	–	–	–	–	–
f_5	–	*	–	*	*	–	–
f_6	–	*	*	–	–	–	*
f_7	–	*	*	*	*	–	*
f_8	–	*	*	*	*	*	*
f_9	–	*	*	*	*	*	*
f_{10}	*	–	*	*	*	*	*
f_{11}	*	*	*	*	*	*	*
f_{12}	*	–	*	–	*	–	–
f_{13}	*	*	*	*	*	*	*
f_{14}	*	*	–	–	*	*	*

Table 6.3: Significance of results for *rule 2-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* where + = better, * = equivalent, – = worse

150 particles have a negligible effect. The regular spacing between the convergence plots in Fig 6.1(a) indicate a linear relationship between the number of particles in the swarm and the number of function evaluations necessary to optimize the minima - the mean difference in function evaluations between two swarms of size NP and size $NP + 1$ for $NP = 15$ to 100 is calculated to be 1498.8 ± 3.9. As the mean value of $FEvals/NP$ (i.e. the number of iterations required for convergence) for the same values of NP is 1503.4 ± 4.4, we can surmise that each additional particle over the minimum of 15 is completely non-integral to the optimization of the single minima, and merely serves to increase the number of required function evaluations unnecessarily.

With this information we can predict that small swarms will perform best on unimodal problems, and on multimodals where the global attractor is easily found, and that large swarms will perform poorly on these same problems. This is borne out in the results obtained, where rule 1b, which reduces the number of particles in the swarm on improvement, shows excellent performance on all three unimodal problems, and several of the simple multimodals. Likewise rule 1a, which increases NP on improve-

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
f_1	*	–	*	*	*	*	*
f_2	–	*	*	*	*	*	*
f_3	*	–	*	*	*	*	–
f_4	–	–	–	–	–	–	–
f_5	*	–	–	*	*	–	–
f_6	*	*	*	*	*	*	*
f_7	*	*	*	*	*	*	*
f_8	*	*	*	*	*	*	*
f_9	*	*	–	*	*	*	*
f_{10}	*	–	*	*	*	*	–
f_{11}	*	*	*	*	*	*	*
f_{12}	*	–	*	*	*	*	–
f_{13}	*	*	*	*	*	*	*
f_{14}	*	*	–	*	*	*	*

Table 6.4: Significance of results for *rule 3-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* where + = better, * = equivalent, – = worse

ment, shows poor performance on the same problems.

Intriguingly, rule 2 showed much worse performance on the unimodal problems than either of the variations of rule 1. The simple multimodal problems f_{10}–f_{14} were the only ones for which rule 2 returned good results when adapting NP.

Rule 3 was equivalent to the fixed-parameter PSO-DRS on f_1 and f_3, and worse on f_2 (though not nearly as much so as rule 2). On the simple multimodals it showed very good performance, equivalent to the fixed-parameter PSO-DRS on all problems.

The more complex multimodal problems, which require a balance of exploration and exploitation, were mixed in results for the two basic rules, with each equalling the fixed-parameter swarm for f_7–f_9. Rule 1b was able to reliably find the global minimum on each of these three, and regularly find it on f_6. Rule 2 again performed extremely poorly on all of these, while rule 3 again performed very well on almost the entire range.

While it is straightforward to track the value of *NP* throughout the optimization process, which can be seen in fig 6.16(a), the dip statistic is not very applicable to this adaptation. Particles that are added to the swarm are placed randomly within its diameter, and removed particles are chosen randomly; both activities lead to large jumps in the dip value, depending on where these particles are added/removed, that do not reflect the actual behaviour. In addition, the increase and decrease in the sample size of the distribution with each adaptation leads to a constantly changing threshold for unimodality in the dip statistic. For this reason the measure was not used as an indicator of behaviour for swarms where *NP* was under adaptation.

6.5.3 Adapting K

The social aspect of the PSO-DRS algorithm, i.e. the recombinant position, is constructed in such a way that adjusting the *k-connectivity* of the swarm topology has the opposite effect to adjusting this parameter

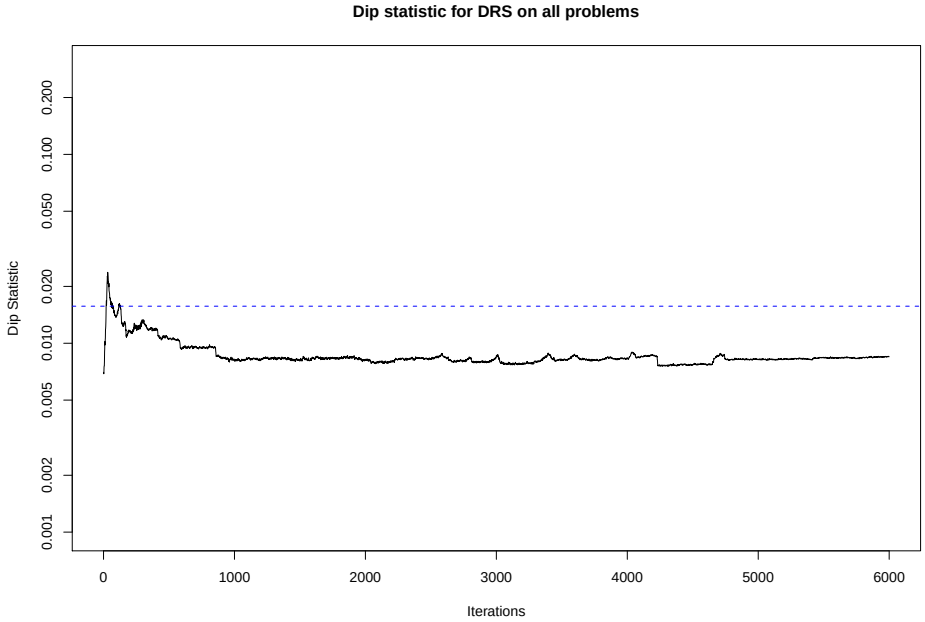
in a standard PSO swarm. The social component of the SPSO algorithm takes the single best position found by any member of the neighborhood of the updating particle, which means that an increase in K has the effect of making a single best found position available to an increased number of particles, in turn increasing the rate of convergence.

Conversely, the recombinant position in the PSO-DRS algorithm is composed of a combination of the positions of every neighbor, randomly chosen by dimension. A larger neighborhood means more potential values are available for each dimension of the recombinant term, which results in a larger number of possible recombinant positions. This increase in the searchable area for the particle slows convergence for the swarm, as seen in Fig 6.1(b).

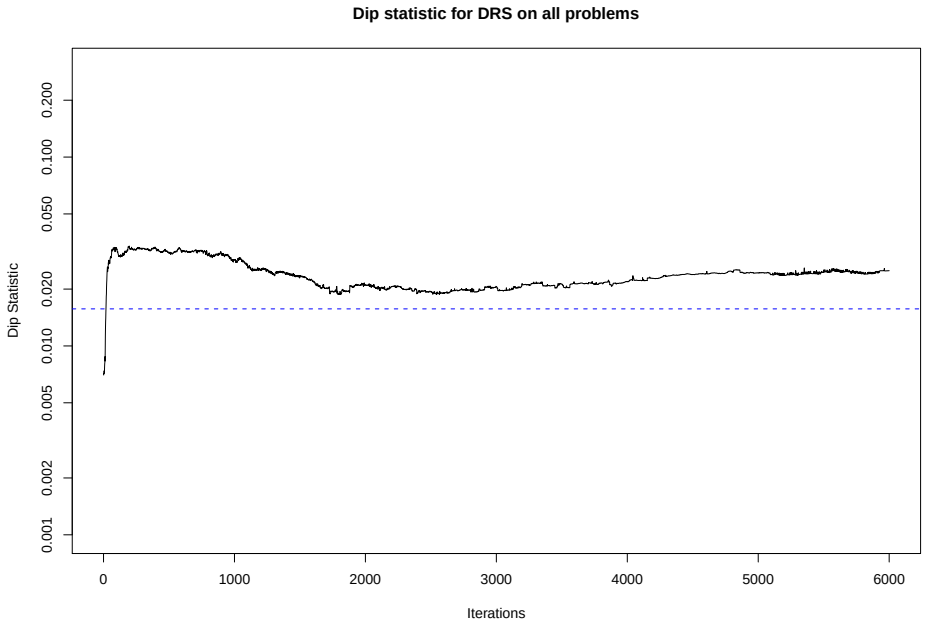
The effect of the rate of convergence is clearly seen in the results for the single-point adaptation of the K parameter for the unimodal sphere function, f_1. Comparing the number of function evaluations required for optimization using the 1a and 1b adaptation rules shows that a swarm adapted by rule 1b finds the optimal point much faster than one using rule 1a, in 75000 evaluations vs 136000. This is expected behaviour - as the swarm converges to the minima, the best found position is continually improving, resulting in a reduction in the value of K when using rule 1b. Each reduction reduces the potential search space for a particle, down to the value K=2, where it is only able to obtain possible corners of the hypercube from its immediate neighbors and limiting the influence of outliers to the minimum possible sub-section of the swarm. Given the lack of additional peaks in f_1, this focuses the optimization power of the swarm on the single minima and prevents unnecessary function evaluations from being expended in a generally fruitless discovery phase.

The more interactive adaptive techniques of rules 2 and 3 had mixed performance on the unimodal problems, with the sphere, f_1, proving particularly problematic. On f_2 both outperformed the diverging behaviour of rule 1a, and rule 2 significantly outperformed the other rules, though still falling short of the fixed-parameter PSO-DRS. Results were better on the multimodal problems, with either one or both rules providing equal or superior performance over both the fixed-K and rule 1a/1b-adaptive formulations on all but two problems (f_{10} and f_{12}).

Behaviour for the K-adapted swarms, seen in the dip values in figs 6.10(a) and 6.10(b), was quite interesting. Despite the slow convergence rates of the rule 1a-adapted swarm, apart from a very brief moment at the start of the optimization, the average dip value remained well below the modality threshold in general, demonstrating that the swarm had almost no period of multi-peak discovery. Rule 1b, on the other hand, kept the swarm on average above the threshold throughout the optimization process, indicating that convergence of the entire swarm to a single peak was rare and difficult. This is seemingly opposite to what should be expected - slow-converging rule 1a swarms should have high dip values on average, and fast-converging rule 1b swarms should have low average dip values. This discrepancy is explained by the details of how the value of K affects the update equations for particles.



(a) Rule 1a



(b) Rule 1b

Figure 6.10: Average dip values for swarms adapting K over all problems

The low average dip value seen in fig 6.10(a) for the rule 1a-adapted swarm is a result of the increased neighborhood size of each particle for increased values of *K*. With more possible positions available for a particle to move to, the likelihood of it moving nearer to a single neighbor in multiple dimensions is decreased, with a corresponding reduction in the ability of multiple particles to clump together in separate groups. This is reflected in the nearly constant unimodal distribution of the particle positions within the swarm. In the same way, the constant reduction of a particle's options for movement down to the positions of only a very few neighbors, as in the rule 1b-adapted swarm results seen in fig 6.10(b), will lead to individual particles being drawn together in many, or all of the dimensions of the problem landscape, hence the observed higher average dip values.

6.6 Results for Multiple Points of Adaptation

As each adaptable parameter influences the convergence rate of the swarm by different amounts, adapting multiple parameters in a single optimization process will lead to adjustments that are finer-grained and more complex. With 136 possible values for *NP* (15-150), there are equivalently 136 possible rates of convergence for a swarm that adapts this parameter and uses fixed values for ϕ and *K*. Similarly, the number of selectable rates for ϕ is 43 (1.08-1.50) and for *K* is 48 (2-49) in the case where *K* alone is under adaptation and *NP* is fixed at 50.

When more than one of these parameters is adapted simultaneously, however, the number of possible combinations of their associated rates of convergence is much higher, and given the explicit interaction between *NP* and *K*, not a fixed value over time. The lowest of these values, for the *K* + ϕ multi-point adaptation, is already 2064 $(48 * 43)$ possible combinations, while the highest, for *NP* + *K* + ϕ, is $6450\ NP - 12900$ $(150 \times (NP - 2) \times 43)$, depending on the value of *NP*, giving a range of $[83850..954600]$ possible combinations for $NP = [15..150]$.

Given this much finer level of granularity, it is possible to fine-tune the convergence rate of the swarm to a much higher degree. Multiple approaches to selecting which of the parameters under adaptation to adjust are available (e.g. random, weighted random, heuristic rules, etc.) and were tested - the results reported here used random selection for simplicity, as well as due to the minor effect this choice had on simulations. In cases where both *NP* and *K* were under adaptation, if $K = NP - 1$ and *NP* was selected and reduced, K was accordingly reduced to the new value of $NP - 1$. *K* was unchanged when *NP* was selected and increased.

While this information goes some way to explaining the increased complexity of multi-parameter adaptation over single-parameter, it is in no way a full picture of the swarm under these conditions. The interactions between the parameters, and the corresponding effect on behaviour, is a deep subject that is reserved for future work.

6.6.1 Adapting *NP + K*

As seen in the single-point adaptation results, the unimodal problems were best optimized by swarms that had the fastest rates of convergence, i.e. those for which the adaptable parameters are at low values. This was exemplified by rules 1a and 1b: the swarm being adapted by rule 1a increased the parameters as the minima was optimized, slowing the convergence speed, and giving poor results, while the rule 1b swarm decreased the parameters during optimizations and returned excellent results for all unimodal problems.

The other two adaptive rules - designed to strike a balance between extremes - were expectedly more mixed. Rule 2 was unable to successfully optimize f_1, but results were good for the other two unimodals. Rule 3 did poorly on f_2, though it is notable that this was largely due to the wide disparity in performance between runs, reflected in the high standard error. On "good" runs, the rule 3-adapted swarm did much better than the baseline fixed-parameter swarm, while on "bad" runs, it did much worse.

For the complex multimodal problems, rule 3 showed the best performance overall, with competitive success rates and means. The same disparity between good and bad runs was again seen via the excellent minimums, very poor maximums, and high standard error on the means. Rule 2 performed well in most cases when only the mean performance is looked at - success rates were abysmal throughout. The rule 1 variations each performed poorly on $f_4 - f_6$, but better on $f_7 - f_9$.

Rule 3 again performed very well on the simple multimodal problems, competitive with the fixed-parameter swarm on all, though slightly slower. Rule 2 attained good mean performance on the shekel problems, but had low success rates for these.

6.6.2 Adapting *NP* + ϕ

Unimodal results for the rule 1 variations were very poor for rule 1a, and more mixed for 1b, which performs well on f_1, and despite a high mean on f_2, the variance is high enough to make it statistically equivalent to fixed-parameter PSO-DRS. The combination of adapting both *NP* and ϕ, the two most influential PSO-DRS parameters, causes the swarm to either diverge too quickly, missing the opportunity to find good minima, or to converge too quickly, in a suboptimal region of the global optimum. This is supported by the improved results seen when using the less extreme rules 2 and 3.

Performance on the complex multimodals was somewhat similar to that seen in the *NP+K* adaptation. The rule 1 variations both gave mostly abysmal results on $f_4 - f_6$, and better on $f_7 - f_9$ - 1b was particularly good on the latter three problems, performing competitively with the fixed-parameter PSO-DRS. In contrast, the same rule fell victim to the same trap seen in the single-point ϕ adaptation, where the swarm being adapted converged upon and was unable to escape a local minima located within the scaled initialization region.

Rule 2 again returned very good results in terms of mean performance, but was again unable to

find the optimum on any of these problems, giving 0% success rates. Rule 3, meanwhile, was very good on the entire range of problems, and particularly effective on the very difficult Rastrigin problem, f_5. All other adaptive PSO-DRS, fixed-parameter PSO-DRS, and SPSO swarms were unable to find the optimum for this problem (barring two minor exceptions of 6% and 4%), making the 32% attained by this adaptation a standout result.

The simple multimodal problems were fairly easily optimized by all of the adapted swarms, though it is interesting to see that while rule 1b had excellent success rates, the means are actually quite high. This is explainable by the very high standard errors - on the few occasions where the swarm did badly, it was only able to find very sub-optimal attractors.

6.6.3 Adapting *K* + ϕ

Results for unimodal problems $f_1 - f_3$ follow the same general pattern seen previously, with 1a giving very poor performance overall, and 1b doing very well. Rule 2 performs very similarly to 1b, while rule 3 does well on f_1 and much poorer on f_2 and f_3, although the high standard error on the latter problems indicates a high disparity between good and bad runs.

Complex multimodal results show that the extreme behaviour of rules 1a and 1b results in performance inferior to the fixed-parameter swarm on all but one problem for each. Rules 2 and 3 each did rather well, equaling the excellent performance of PSO-DRS on several of the problems when measured by mean performance. However in most cases, all of the rules gave at best mediocre performance on the complex multimodal problems in terms of successful optimizations. None was able to equal the results of the fixed-parameter PSO-DRS by that measure, and many of the results were the worst obtained for any of the adaptive swarms.

Similarly, success rates were inferior for the simple multimodal problems to almost all other swarm configurations, though not as much so as seen in $f_4 - f_9$, though means were equivalent to the fixed-parameter swarm in most cases for all but rule 1a. The most notable feature of the performance on these problems is in the speed of the successful optimizations, with a significantly lower number of function evaluations needed to find the optimum in the relevant cases. When the swarm is unable to find the optimum, as is the case much of the time, the alternative point of convergence tended to be very poor.

6.6.4 Adapting *NP* + *K* + ϕ

The most-adapted form of the PSO-DRS works with all three parameters, giving both the most finely-grained degree of adaptation, and the most extreme rates of convergence. The effects of these extremes are represented in the unimodal results for rules 1a and 1b, with each one giving extremely poor results, with the exception of 1b on the basic sphere problem. Fast, successful performance is to be expected in that individual instance, given that the algorithm needs only to converge as fast as possible to a very easily-found minima. The other problems, f_2 and f_3, require slightly better discovery ability alongside

fast convergence, resulting in extremely poor performance by the two rule 1 variations – with that said, statistical equivalence to PSO-DRS was achieved by 1b on f_2 by means of the high variance. Rules 2 and 3 provide this better balance, reflected in the much-improved results.

This same pattern is seen in the complex multimodal problems, although here rule 1b is able to return at least mediocre results on $f_7 - f_9$. Rules 2 and 3 again are better overall, and although neither is able to match the fixed-parameter PSO-DRS on a majority of the problems, mean performance is equivalent in the majority of trials. Similar results are obtained for the simple multimodals, particularly in the case of the performance of rule 1a on f_{10}, which is the worst obtained by any of the algorithms examined here.

6.7 The Lennard-Jones Atomic Cluster Optimization Problem

6.7.1 Background

The Lennard-Jones potential is a mathematical model that closely approximates the interaction between neutral atoms[133]. Determining the structural arrangement of multiple atoms that minimizes this potential energy function presents an ideal problem for global optimization, in that it is simultaneously easy to describe and formulate, but difficult to solve. This problem has existed in the literature for quite some time and is well understood [135][136]. Its exponentially increasing number of local minima with the number of atoms makes it a straightforward matter to attempt to solve formulations of increasing difficulty.

On a practical note, chemistry literature commonly cites the knowledge of the global minima of Lennard-Jones clusters as "a fundamental step towards a better understanding of some molecular conformation problems"[137]. Because of this a great deal of effort has gone into determining these minima, and to date they are believed to be known for all cluster sizes of up to 250 atoms[138].

Lennard-Jones problems are fairly well known in the global optimization literature, and are used to demonstrate the applicability of algorithms to "real world" problems. Purpose-built genetic algorithms have been demonstrated to be able to reliably solve the problem for up to at least $N = 110$, when the algorithm was specifically designed for solving only this specific problem[139][140]. PSO algorithms have also been applied in varying capacities, from GA-comparative works[141] to use as examples of multi-funnel landscapes[142]. These examinations were limited to at most $N = 15$, and showed mostly poor performance – [142] was in fact an attempted demonstration of the unsuitability of the standard PSO algorithm for this type of landscape.

Better results were obtained for the $N = 26$ Lennard-Jones problem using an enhanced and extensively specialized form of the inertia-weight formulation of PSO[143], indicating that the algorithm is not inherently unsuited to Lennard-Jones problems, but can be adapted from the default configuration to provide improved performance. This reformulation of the algorithm introduced a number of features

implemented exclusively for the atomic structure problems, such as a measure of "distance" between discovered atomic structures and the associated implications to the organization of the problem space. Further modifications included a vast increase to swarm size to 1000 particles, and the use of periods of attraction and repulsion between particles in order to maintain diversity within the swarm. These modifications allowed the algorithm to reliably find the global optimum of the 26-atom Lennard-Jones problem (it was not tested on any other configuration), but restricted its applicability entirely to the optimization of this and similar atomic structuring problems[143].

6.7.2 Definition

The potential energy in a Lennard-Jones atomic cluster can be represented by the equation:

$$E = 4\epsilon \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[(\frac{\sigma}{r_{ij}})^{12} - (\frac{\sigma}{r_{ij}})^{6}) \right] \tag{6.2}$$

where N is the number of atoms in the cluster and r_{ij} is the distance between two atoms i and j. ϵ and σ represent specific features of the atomic structure, and here are set to 1, as per previous studies[140][143].

The large number of very deep minima requires an optimization algorithm to be able to both efficiently explore the landscape, and then rapidly exploit the discovered basin of attraction prior to premature convergence. 21 different formulations of the problem were used here, of increasing difficulty: starting with the simplest configuration, 2 atoms, all configurations up to 20 atoms were tested, along with the 26-atom configuration, and the extremely complex 38-atom configuration. Minimum potential energies for all 21 configurations are shown in table 6.1. Each atom was represented in 3-dimensional space, giving problems with dimensionalities ranging from 6 for the 2-atom configuration, to 114 for the 38-atom configuration.

Problem	Energy	Problem	Energy	Problem	Energy
LJ_2	-1.000000	LJ_9	-24.113360	LJ_{16}	-56.815742
LJ_3	-3.000000	LJ_{10}	-28.422532	LJ_{17}	-61.317995
LJ_4	-6.000000	LJ_{11}	-32.765970	LJ_{18}	-66.530949
LJ_5	-9.103852	LJ_{12}	-37.967600	LJ_{19}	-72.659782
LJ_6	-12.712062	LJ_{13}	-44.326801	LJ_{20}	-77.177043
LJ_7	-16.505384	LJ_{14}	-47.845157	LJ_{26}	-108.315616
LJ_8	-19.821489	LJ_{15}	-52.322627	LJ_{38}	-173.928427

Table 6.5: Minimum potential energy for tested Lennard-Jones configurations.

The 2-atom through 20-atom problems provide both a sizable sample for obtaining a full view of the performance of each algorithm, and a good point of comparison for other algorithms in the literature, most of which test on this range or a subset. The 26-atom configuration was used purely for comparison to the aforementioned PSO variation that reliably solved it due to having been explicitly built and tuned for this specific problem and configuration[143]. The 38-atom configuration of the problem was chosen for its interesting property of having two unique minima, each in a separate global basin of attraction.

This quality both simplifies the problem by providing two available solutions, and increases the difficulty by making convergence to and exploitation of only one of those peaks by the entire swarm more difficult.

There is no commonly-defined initialization or bounding strategy for an optimization population on the Lennard-Jones class of problems. Global optimization literature favors initializing the population with a uniform spread throughout the search space[139], or a region-scaled approach[142], while publications concerned with the real-world chemical aspect of the problems are more forgiving, disallowing unfeasible solutions and enforcing minimum distances between atoms[143][140]. No bounds are explicitly defined either; none of the referenced literature makes mention of them.

As both of these properties can influence the performance of the algorithm, a conservative approach was taken here. The population was uniformly initialized throughout the space, but the initialization space was defined to extend beyond the optimal points in each dimension by 10 times the maximum space between them. For example, the optimal solution for the 3-atom configuration is shown in table 6.2 (precision has been abridged for the sake of this example).

	Dim 1	Dim 2	Dim 3
Atom 1	0.44	0.11	-0.46
Atom 2	-0.52	0.39	0.05
Atom 3	0.08	-0.50	0.40

Table 6.6: Optimal structure for the 3-atom Lennard-Jones problem

In the first dimension, the optimal points for each atom are 0.44, -0.52, and 0.08. The maximum distance is therefore $0.44 + 0.52 = 0.96$, and hence the initialization region in that dimension will cover a space with a size of 9.6, i.e. $[-4.8..4.8]$. Because this value is tied to the configuration of the specific problem under optimization, initializing the swarm in this manner will not predispose any configuration to a performance result that is better or worse than that of any other configuration, as could be the case with bounds of initialization that were fixed and common to all configurations.

Feasible bounds during optimization also vary in the literature. Again, the strategy used here can greatly influence overall behaviour and performance – a feasible search space that only barely encompassed the optimal solution would render the problem much simpler than a space in which the range containing all components of the optimal solution represented only a small area of the whole. To avoid any semblance of problem structure influencing algorithm performance, no boundaries were defined for the atomic configurations tested here. All particle positions were considered feasible and evaluated.

These conservative approaches to the problem search space will almost certainly negatively affect the performance of the algorithms tested here, at least when compared to that of the algorithms proposed in other studies. As the focus of this work is mostly confined to the algorithms defined and tested within, however, this only needs to be taken into account in comparisons to the other studies, none of which define their search spaces to the extent of being reproducible.

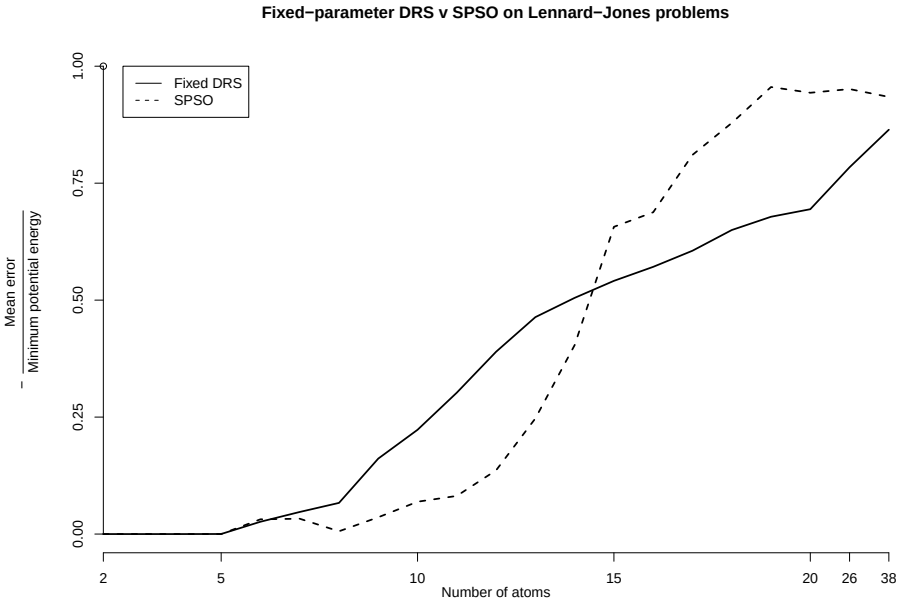


Figure 6.11: Fixed-parameter PSO-DRS v SPSO on full range of Lennard-Jones

6.8 Lennard-Jones Performance

6.8.1 Fixed-Parameter PSO-DRS and SPSO

Both the fixed-parameter discrete recombinant swarm and the standard PSO swarm show uninspiring performance on much of this class of problems – results are shown in tables 6.14 – 6.15. While the SPSO swarm does well on the 2- to 11-atom configurations, mean performance quickly drops after this point, with results for the 18-, 19-, 20-, 26-, and 38-atom configurations only slightly better than a random search. Taking into account the differences in population initialization, these results were in line with those reported in the literature for a basic PSO algorithm[141].

DRS performance shows the opposite general trend, underperforming SPSO in all configurations between 7 and 14 atoms, but giving significantly better mean fitness levels for configurations of 15-20 atoms, as well as the 26- and 38-atom configurations. This can be seen in figure 6.11, which plots $-\frac{\langle error \rangle}{E}$, i.e. negative of the mean error found by the algorithm over E =minimum potential energy (see equation 6.2 and table 6.1). This allows for a scale-free comparison of performance across all 21 problems, from 0.0 indicating a mean error of 0.0, to 1.0 indicating a mean of the maximum error possible.

The marginal performance overall illustrates the potential for an adapted algorithm to give improved performance over the base fixed-parameter swarms.

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
LJ_2	*	*	*	*	*	–	–
LJ_3	*	–	–	–	–	–	–
LJ_4	–	–	–	–	–	–	–
LJ_5	–	–	–	–	–	–	–
LJ_6	–	–	–	–	–	–	–
LJ_7	–	–	–	–	–	–	–
LJ_8	–	–	–	–	–	–	–
LJ_9	–	–	–	–	–	–	–
LJ_{10}	–	–	–	–	–	–	–
LJ_{11}	–	–	–	–	–	–	–
LJ_{12}	–	–	–	–	–	–	–
LJ_{13}	–	–	–	–	–	–	–
LJ_{14}	–	–	–	–	–	–	–
LJ_{15}	–	–	–	–	–	–	–
LJ_{16}	–	–	–	–	–	–	–
LJ_{17}	–	–	–	–	–	–	–
LJ_{18}	–	–	–	–	–	–	–
LJ_{19}	–	–	–	–	–	–	–
LJ_{20}	–	–	–	–	–	–	–
LJ_{26}	–	–	–	–	–	–	–
LJ_{38}	–	–	–	–	–	–	–

Table 6.7: Significance of results for *rule 1a-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* on Lennard-Jones problems where + = better, * = equivalent, – = worse

6.8.2 Adaptive PSO-DRS

The Lennard-Jones problems are where the benefit of adapting the parameters of the PSO-DRS is most clearly seen. Rules 1b, 2, and 3 all show significantly improved performance over the fixed-parameter swarm on many of these difficult problems.

The two variations of rule 1 show the most extreme effects of the adaptive behaviour on performance. Rule 1a, which enforces swarm diversity at the expense of convergence ability, was significantly worse than the fixed-parameter formulation of the PSO-DRS algorithm on all but 6 of the 147 configurations of the problem – it was statistically equivalent on those 6, without a single demonstration of improved performance. Table 6.7 and figures 6.12(a) and 6.12(b) clearly indicate the poor performance of this strategy on optimizing all but the simplest configurations.

As seen on the f_1–f_{14} problems, when one of the variations of rule 1 performs poorly, the other performs well. This is very much the case with rule 1b, which showed exceptional performance on the entire range of Lennard-Jones problems. Of the 147 problems, significantly better performance was obtained in 100 instances, and equivalent performance in another 44. The majority of the improved results came from the more complex atomic configurations, starting from the 8-atom problem – this is especially interesting when the comparatively good performance of PSO-DRS in relation to SPSO on these same problems is taken into account.

Significance results for rule 1b-adapted swarms can be seen in table 6.8, but the performance plots shown in figures 6.13(a) and 6.13(b) provide more information on the effects of the adaptations. Adaptation of the *K*-parameter gives equivalent, nearly identical performance to the fixed-parameter PSO-DRS,

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
LJ_2	*	*	*	*	*	*	*
LJ_3	*	*	+	*	+	+	+
LJ_4	*	*	*	*	*	*	*
LJ_5	+	*	+	*	*	+	+
LJ_6	–	*	*	–	*	*	*
LJ_7	*	*	+	*	*	+	*
LJ_8	+	*	+	+	+	+	+
LJ_9	+	*	+	+	+	+	+
LJ_{10}	+	*	+	+	+	+	+
LJ_{11}	+	*	+	+	+	+	+
LJ_{12}	+	*	+	+	+	+	+
LJ_{13}	+	*	+	+	+	+	+
LJ_{14}	+	*	+	+	+	+	+
LJ_{15}	+	*	+	+	+	+	+
LJ_{16}	+	*	+	+	+	+	+
LJ_{17}	+	*	+	+	+	+	+
LJ_{18}	+	*	+	+	+	+	+
LJ_{19}	+	*	+	+	+	+	+
LJ_{20}	+	*	+	+	+	+	+
LJ_{26}	+	*	+	+	+	+	+
LJ_{38}	+	–	+	+	+	+	+

Table 6.8: Significance of results for *rule 1b-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* on Lennard-Jones problems where + = better, * = equivalent, – = worse

which fixes K at 2, in all but the most complex problem configuration. As rule 1b encourages fast convergence by decreases the value of the parameter under adaptation every time the swarm improves its best found position, this result, alongside the poor result for increased values of K under rule 1a, confirms that the lowest value of K=2 will give the best possible performance for all values of K on the Lennard-Jones problems.

It is notable that while the swarms adapting either NP or ϕ using rule 1b performed very well, when those two parameters were adapted together performance was even further improved, as can be seen in the results for NP+ϕ and NP+K+ϕ. Both of these configurations showed performance that was the best of any rule / parameter(s) combination, indicating that pushing for the fast convergence associated with low values for the adaptive parameters is the best found strategy for the atomic structuring problems when optimized with a PSO-DRS algorithm.

Results for the rule 2-adapted swarm were mixed, and are shown in table 6.9 and performance plots in figures 6.14(a) and 6.14(b).

The swarms that placed NP or K individually under adaptation both performed significantly worse than the fixed-parameter swarm, but the performance of the swarm using rule 2 to adapt ϕ was significantly improved. When the adaptations were applied to multiple parameters, the results showed that the adaptation of NP+K resulted in performance that was almost always significantly worse than the fixed-parameter swarm, but each of the combinations of parameters that involved ϕ (NP+ϕ, K+ϕ, and NP+K+ϕ) was significantly improved. In particular, the NP+ϕ adaptation gave results that were equivalent to the best seen in the best rule 1b-adapted swarms.

	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
LJ_2	*	*	*	*	*	*	*
LJ_3	–	–	*	*	+	+	+
LJ_4	–	–	*	*	*	*	*
LJ_5	–	–	+	+	*	+	*
LJ_6	–	–	*	–	*	*	*
LJ_7	–	–	+	–	*	+	+
LJ_8	–	–	+	–	+	+	+
LJ_9	–	–	+	–	+	+	+
LJ_{10}	–	–	+	–	+	+	+
LJ_{11}	–	–	+	–	+	+	+
LJ_{12}	–	–	+	–	+	+	+
LJ_{13}	–	–	+	–	+	+	+
LJ_{14}	–	–	+	–	+	+	+
LJ_{15}	–	–	+	–	+	+	+
LJ_{16}	–	–	+	–	+	+	+
LJ_{17}	–	–	+	–	+	+	+
LJ_{18}	–	–	+	–	+	+	+
LJ_{19}	–	–	+	–	+	+	+
LJ_{20}	–	–	+	–	+	+	+
LJ_{26}	–	–	+	–	+	+	+
LJ_{38}	–	–	+	–	+	+	+

Table 6.9: Significance of results for *rule 2-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* on Lennard-Jones problems where + = better, * = equivalent, – = worse

Rule 3-adapted swarms were also mixed in performance, although not in the same way as those adapted by rule 2. While the *K*-adapted swarm again performed poorly, the swarm that adapted *NP* this time was significantly improved over the fixed-parameter configuration. The ϕ adaptation again gave very good performance.

As before, the results of the single-point adaptations can be seen in the multi-point results, both of which are shown in figures 6.15(a) and 6.15(a). This time the *NP*+ϕ swarm was significantly improved, but every other configuration demonstrated reduced performance. As the *NP*+ϕ combination of parameters is the only one not including *K* in the adaptation, this could be an indication that poor or unnecessary adjustments to *K* are influencing the behaviour of the swarm, negating the improvements that would otherwise come with the adaptation of *NP* and *K*.

6.9 Discussion

6.9.1 Behaviour

The effects of the various rules on the rate of convergence for the swarm over time can be seen in Figs 6.16(a),6.16(b), and 6.16(c). The associated adjustments to the parameters under adaptation follow the predicted patterns for an average optimization: rule 1a drops to the lowest/slowest possible rate of convergence, rule 1b increases to the highest/fastest rate, and rules 2 and 3 operate somewhere in between these two extremes.

Plots of the value for rule 3 adapting ϕ or *NP* show the convergence rate at iteration 1 as substantially higher than for the other rules - this is due to the update schedule for rule 3 being dynamic, rather than a

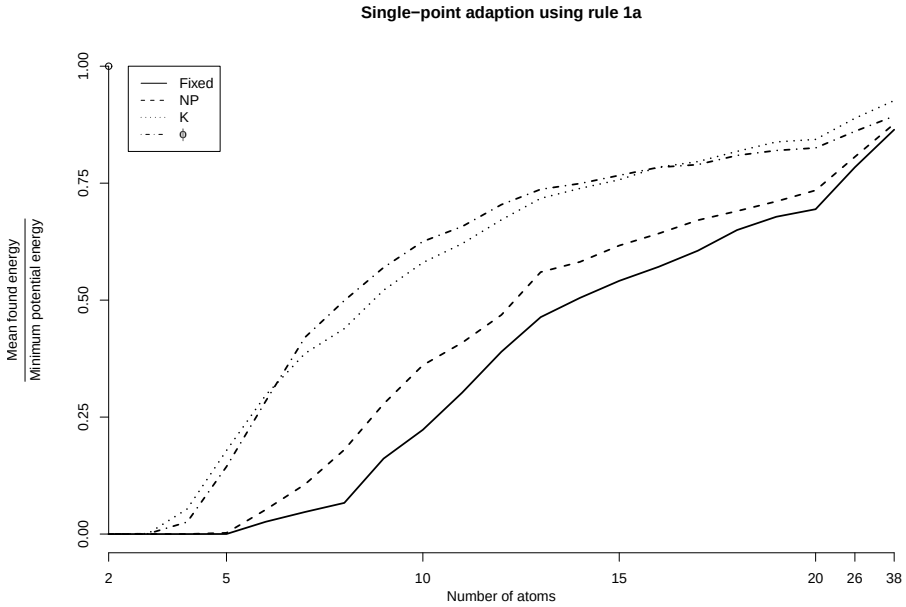
	NP	**K**	ϕ	**NP+K**	**NP+ϕ**	**K+ϕ**	**NP+K+ϕ**
LJ_2	*	*	*	*	*	*	*
LJ_3	+	–	*	–	*	–	–
LJ_4	*	–	*	–	*	–	–
LJ_5	+	–	+	–	+	–	–
LJ_6	*	–	*	–	*	–	–
LJ_7	+	–	*	–	+	–	–
LJ_8	+	–	+	–	+	–	–
LJ_9	+	–	+	–	+	–	–
LJ_{10}	+	–	+	–	+	–	–
LJ_{11}	+	–	+	–	+	–	–
LJ_{12}	+	–	+	–	+	–	–
LJ_{13}	+	–	+	–	+	–	–
LJ_{14}	+	–	+	–	+	–	–
LJ_{15}	+	–	+	–	+	–	–
LJ_{16}	+	–	+	–	+	–	–
LJ_{17}	+	–	+	–	+	–	–
LJ_{18}	+	–	+	–	+	–	–
LJ_{19}	+	–	+	–	+	–	–
LJ_{20}	+	–	+	–	+	–	–
LJ_{26}	+	–	+	–	+	–	–
LJ_{38}	+	–	+	–	+	–	–

Table 6.10: Significance of results for *rule 3-adapted PSO-DRS* vs *fixed-parameter PSO-DRS* on Lennard-Jones problems where + = better, * = equivalent, – = worse

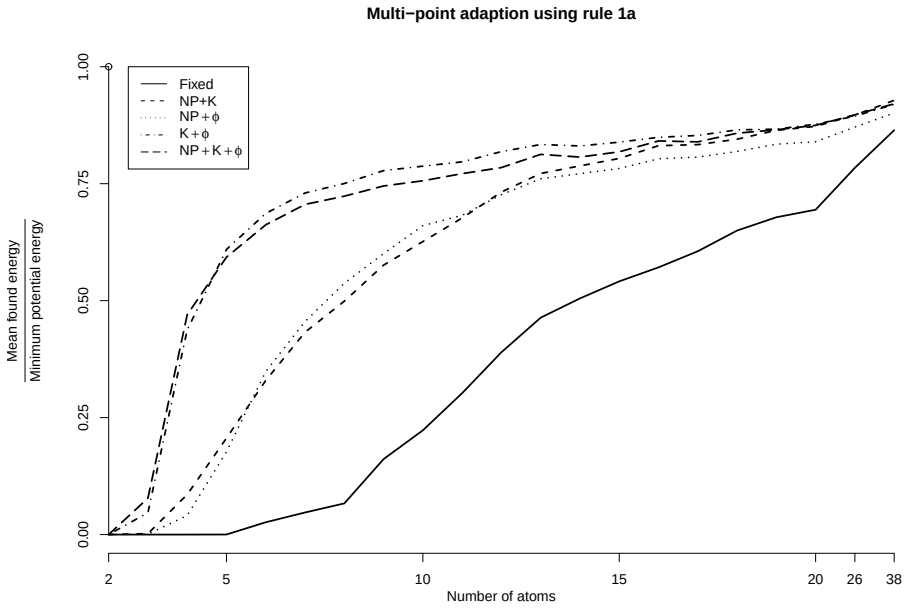
fixed update at the end of each iteration. By the time iteration 1 is complete and the convergence rate is recorded, rule 3 will have updated the value of the parameter under adaptation multiple times, resulting in the higher values seen in the plots at iteration 1.

That the rate of convergence under rule 3 reduces after the initial exploration phase and comes to match that of the rule 2-adapted swarms, especially in the case when ϕ is under adaptation, shows both the similarity and the differences between the two rules. Both appear to slowly raise the adapted parameter throughout the lifetime of the optimization after approximately 100 updates, increasing the rate of convergence, but while this is also the behaviour for rule 2 at the start of the process, rule 3 quickly raises the adapted parameter within the first few updates, then reduces it over time until a point where the increase begins again. Given the superior performance of rule 3 over rule 2 on most of the benchmark problems, this strategy seems to have merit.

In terms of behaviour, this quick increase to convergence speed, followed by a decrease, then a slow increase does not fit as well with the concept of the phased approach to optimization as a simple slow increase. At the very start of the optimization it seems appropriate for the convergence speed to be kept low to prevent premature convergence, slowly increased as minima are discovered, and increased further as a single optima is settled on in order to facilitate fast exploitation. By quickly increasing the rate of convergence right at the start of the optimization, rule 3 encourages the swarm to begin discovering minima immediately, without the initial phase of spreading out to explore the search space. As this behaviour had a positive effect on the performance of the swarm, it could suggest that exploration is less important to the algorithm than quickly selecting and exploiting the first few minima that are discovered.



(a) Single-point



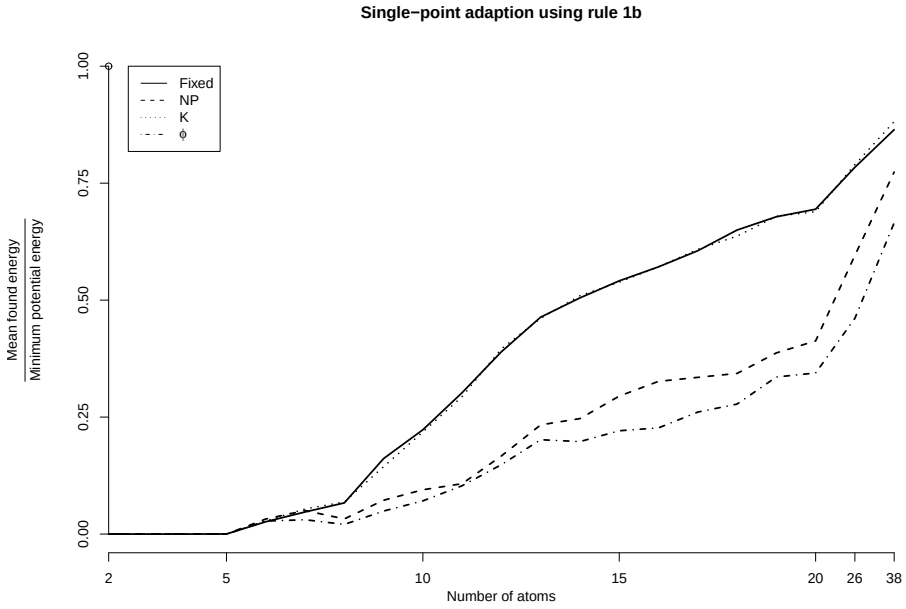
(b) Multi-point

Figure 6.12: Performance of rule 1a-adapted swarms on the Lennard-Jones benchmark

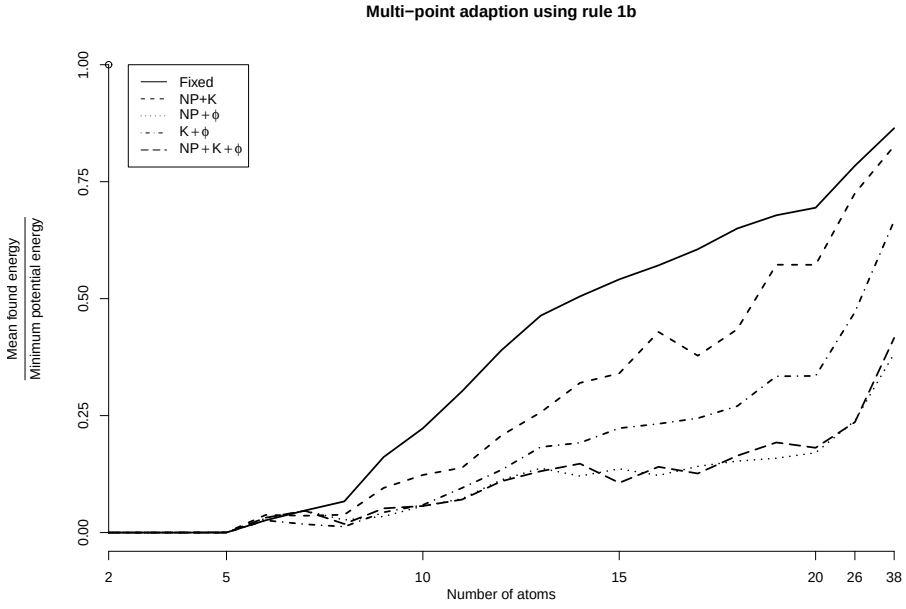
6.9.2 Performance

The performance of the swarms under adaptation ranged quite a bit, depending on the rule being used and the parameter, or parameters, under adaptation.

Rules 1a and 1b, with their extreme adjustments to the adaptable parameters, showed generally poor performance, though 1b obtained both significantly better means and superior single-run results than the fixed-parameter PSO-DRS on the Lennard-Jones problems in most cases. This was counter-balanced



(a) Single-point

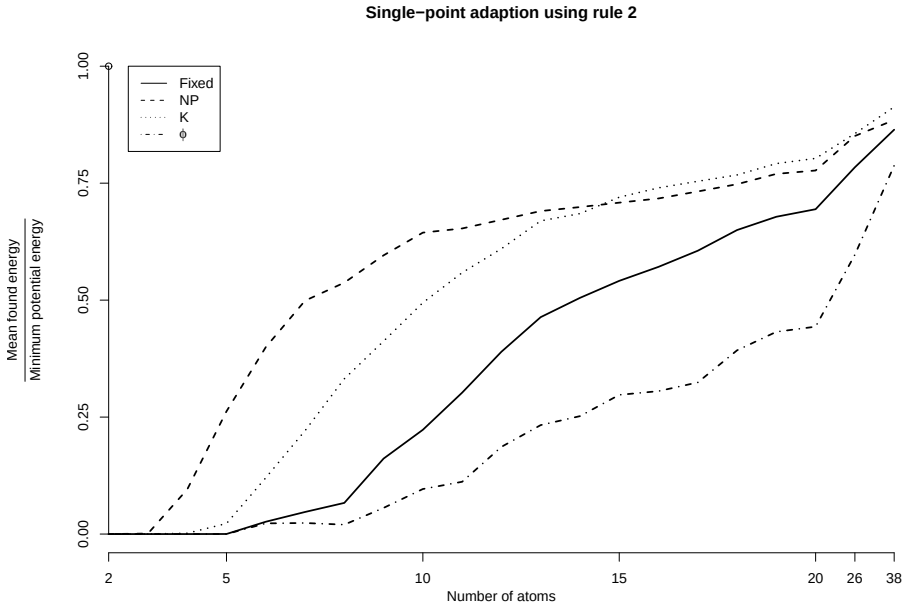


(b) Multi-point

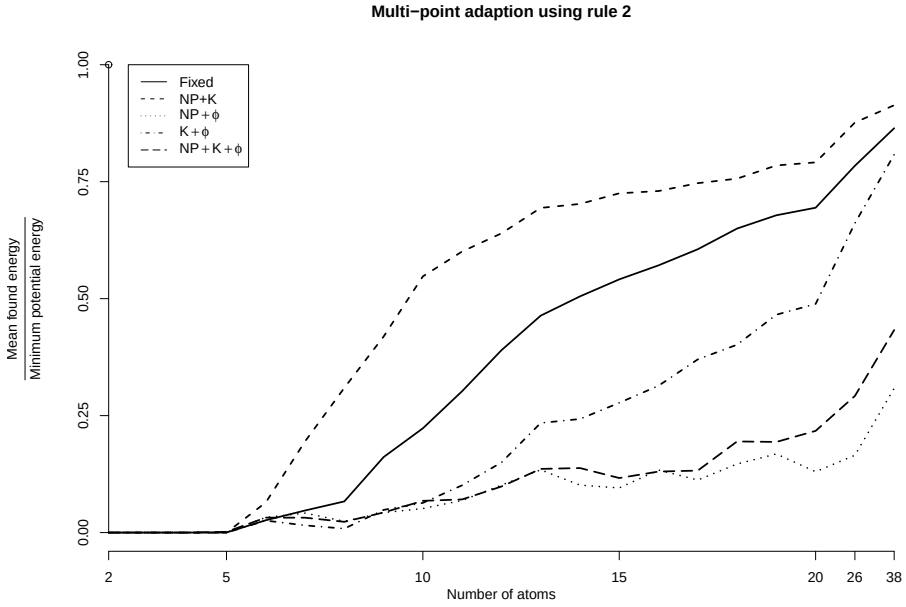
Figure 6.13: Performance of rule 1b-adapted swarms on the Lennard-Jones benchmark

by its generally poor comparative performance on the complex multimodals f_4–f_9. Given the observed behaviour of this rule in rapidly increasing the rate of convergence, this poor performance on problems with a large number of minima is to be expected.

The excellent performance of 1b on the Lennard-Jones problems was contrasted by the extremely poor performance of rule 1a on the same set. This relationship held to some extent on the unimodal problems, with 1b showing equal means and equal-to-superior best results on the majority of the prob-



(a) Single-point

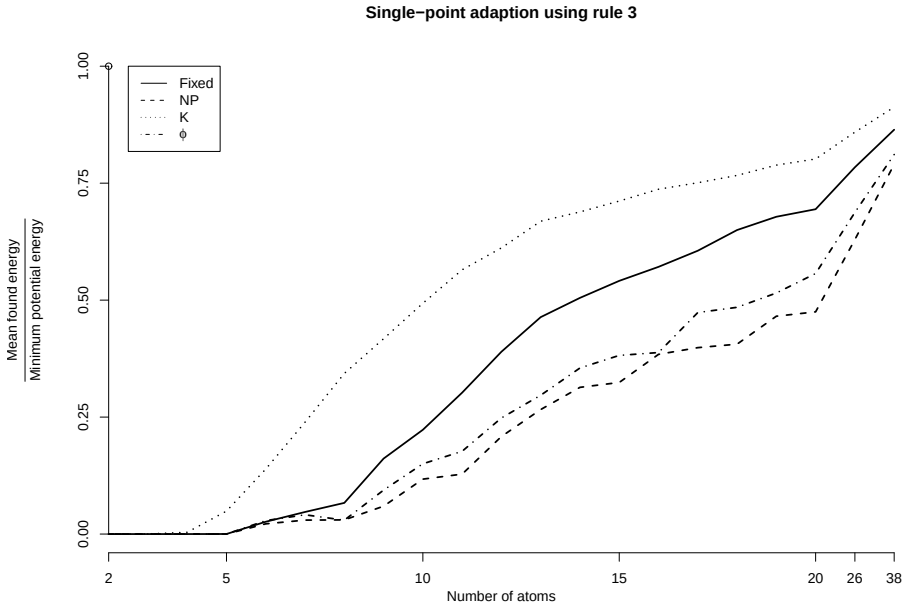


(b) Multi-point

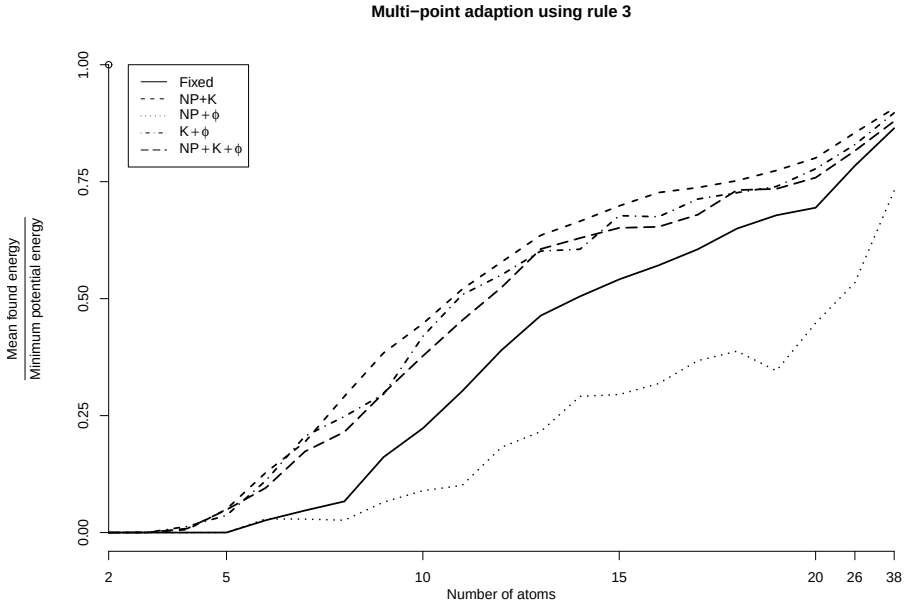
Figure 6.14: Performance of rule 2-adapted swarms on the Lennard-Jones benchmark

lems, while 1a was inferior to the fixed-parameter swarm for nearly all of the combinations of adapted parameters.

Given the superior performance of the exploitation-focused rule 1b on problems with a single, possibly very shallow global optimum, it is reasonable to expect the observed opposite effect on multimodal problems. Despite its emphasis on slow convergence, rule 1a also performed rather poorly on these problems due to a lack of ability to settle down and optimize a single point after discovery. Rules 2 and



(a) Single-point

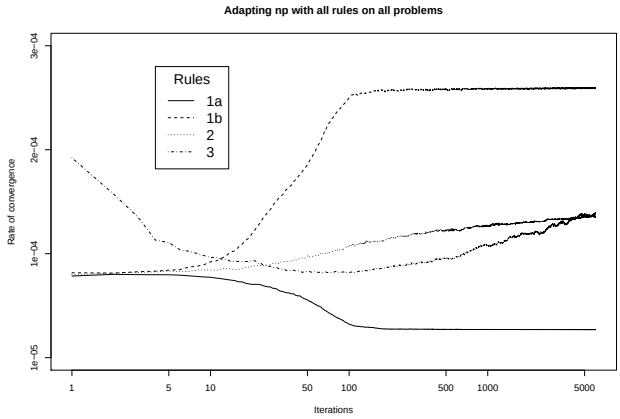


(b) Multi-point

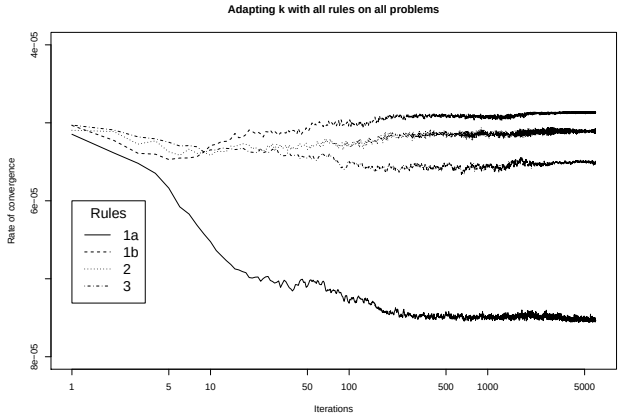
Figure 6.15: Performance of rule 3-adapted swarms on the Lennard-Jones benchmark

3 made an effort to obtain a compromise in behaviour, and returned acceptable behaviour on the entire suite of problems.

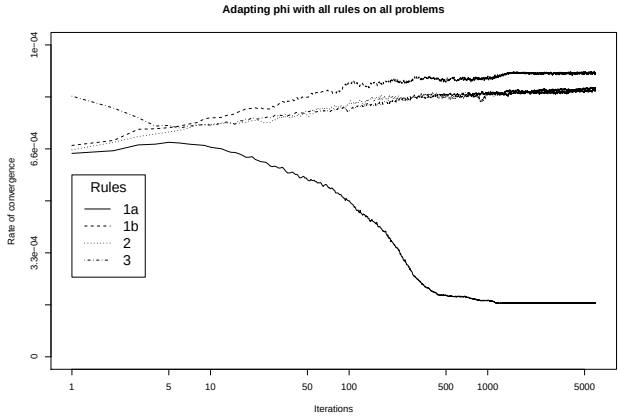
Despite their similar general behaviour described in the previous section, the performance of rule 3 was superior to rule 2 on all of the unimodal and multimodal problem, and very slightly inferior on the Lennard-Jones problems. Rule 3 also gave very good performance results in comparison to the fixed-parameter swarm, particularly on the *NP*+ϕ-only adapted swarm, where mean performance was



(a) *NP*



(b) *K*



(c) ϕ

Figure 6.16: Convergence rate over time for single-point adaptation

statistically equivalent all but one problem – f_4, on which the fixed parameter swarm excels. Similar comparative performance was seen for nearly all of the other rule 3-adapted swarms.

In terms of sheer performance, several of the configurations of rules and adaptive parameters produced equivalent or superior performance to the fixed-parameter PSO-DRS algorithm across the entire benchmark. The swarms using rule 1b to adapt the number of particles performed significantly better on

68% of Lennard-Jones trials, and equivalent on another 30%. The rule 2-adapted *NP+*ϕ and *NP+K+*ϕ swarms were significantly better than PSO-DRS on all but a few Lennard-Jones problems, and worse on three of the standard problems, both multi- and uni-modal.

Rule 3 produced an equivalently-performing configuration, and an improved configuration: the adapted *NP* swarm was better than PSO-DRS on eighteen of the 21 Lennard-Jones and worse on a single multimodal and a single unimodal problem, and results for the *NP+*ϕ-adapted swarm were improved for seventeen Lennard-Jones problems, and only worse on f_4.

Full results are shown in tables 6.11 – 6.15 for the swarm adapting all three parameters using rule 2, and for the swarm adapting *NP+*ϕ using rule 3. The former is interesting because while it is significantly worse than the fixed-parameter PSO-DRS on a few of the multimodal problems, its performance on the Lennard-Jones class of problems is exceptional. The latter, rule 3-adapted swarm is included in these full results because it is the best performer across the entire benchmark.

		SPSO	**DRS**	**DRS**	**DRS**
	Adapting	*Fixed*	*Fixed*	***NP+K+*ϕ*: R2***	***NP+*ϕ*: R3***
f_1	Success rate	**100%**	**100%**	96%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	9.9E-15±9.4E-15	**0.0±0.0**
	Worst	**0.0**	**0.0**	4.7E-13	**0.0**
	FEvals±StE	109253±360	**76748±847**	174703±15602	102079±9413
f_2	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	8.66E-8	3.61E-5	1.16E-6	**5.64E-9**
	Mean±StE	**2.39E-6±4.86E-7**	3.36E-3±1.21E-3	2.71E-3±2.04E-3	4.02±1.67
	Worst	**2.27E-5**	5.14E-2	9.5E-2	74.0
	FEvals±StE	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	1.32E-4	**2.72E-5**	6.36E-3	1.30E-3
	Mean±StE	**2.81±0.55**	8.48±1.18	11.35±1.57	15.23±3.29
	Worst	**14.36**	33.1	66.58	80.3
	FEvals±StE	-	-	-	-

Table 6.11: Results for **best-performing** adaptive rules vs SPSO and PSO-DRS on unimodal problems

Although these two configurations produced the best performance, both specifically and generally, it was common to see configurations that showed much improved performance on several problems. For example, both the rule 2- and rule 3-adapted swarms produced equivalent or better performance in more than half of the possible configurations on the Lennard-Jones problems, and the rule 1b-adapted swarms were equivalent or superior in all but one case. Looking strictly at best single-run cases, rule 1b-adapted swarms did very well on the f_2 unimodal, and the rule 2-adapted swarms were excellent in the majority of configurations on the highly-complex f_5. The rule 3-adapted swarms were even better in terms of individual runs, particularly when adapting the *NP* parameter, where the single best run was equivalent or superior to PSO-DRS in every case.

While performance is not the primary goal of this investigation, it is encouraging to see that improved results can be obtained even with the straightforward rules and adaptations proposed and implemented here. Given how these rules were able to be designed to produce specific behaviours and corre-

	Adapting	**SPSO** *Fixed*	**DRS** *Fixed*	**DRS** ***NP+K+ϕ: R2***	**DRS** ***NP+ϕ: R3***
f_4	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	2961	**829**	2606	948
	Mean±StE	3264±21	**1576±39**	3238±45	1952±72
	Worst	3614	**2013**	3553	3079
	FEvals±StE	-	-	-	-
f_5	Success rate	0%	0%	0%	**32%**
	Best	98.50	2.98	6.04E-14	**0.0**
	Mean±StE	149.02±3.48	**9.19±0.64**	20.68±2.42	11.72±2.22
	Worst	198.0	**21.89**	73.63	56.7
	FEvals±StE	-	-	-	**407830±12921**
f_6	Success rate	20%	**100%**	0%	12%
	Best	**0.0**	**0.0**	4.68E-9	**0.0**
	Mean±StE	14.68±1.16	**0.0±0.0**	1.32E-6±7.99E-7	4.17E-2±2.94E-2
	Worst	19.75	**0.0**	3.39E-5	1.16
	FEvals±StE	239923±39688	**226790±9485**	-	281326±17253
f_7	Success rate	**98%**	94%	20%	78%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**1.48E-4±1.48E-4**	4.44E-4±2.51E-4	3.55E-3±1.12E-3	2.37E-3±7.07E-4
	Worst	**7.4E-3**	**7.4E-3**	3.2E-2	2.22E-2
	FEvals±StE	124726±4922	**70276±704**	317417±18013	130082±11133
f_8	Success rate	**100%**	96%	98%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	4.15E-3±2.9E-3	5.1E-14±5.1E-14	**0.0±0.0**
	Worst	**0.0**	1.04E-1	2.5E-12	**0.0**
	FEvals±StE	128167±1107	**95725±630**	129594±13401	111774±9765
f_9	Success rate	**100%**	98%	94%	86%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	2.2E-4±2.2E-4	1.03E-10±9.45E-11	1.74E-3±6.44E-4
	Worst	**0.0**	1.1E-2	4.71E-9	2.1E-2
	FEvals±StE	118098±440	**91635±454**	184279±12718	138086±12368

Table 6.12: Results for **best-performing** adaptive rules vs SPSO and PSO-DRS on complex multimodal problems

sponding performance, using the same guidelines established here opens up PSO-DRS for performance-focused exploration and rule development.

6.9.3 Adaptive Framework

The PSO-DRS form of particle swarm optimization has been demonstrated in previous chapters to show excellent performance on many problems, and clear improvements to the standard PSO algorithm. Further improvements beyond what has been achieved on the selected benchmark are difficult to obtain, given that PSO-DRS is reliably able to find the optimal point in 10 of the original 14 problems, and very good points on the remaining 4. Despite this difficulty, introducing adaptive techniques via the framework shown in figure 6.17 were shown to produce significantly improved performance in many cases, alongside expected and explainable behaviour patterns.

This is the desired result of this examination, showing that PSO-DRS-driven optimization can be easily tuned to produce superior performance, without altering any of the fundamental properties of the algorithm. Nothing in the initialization or processing components of the algorithm was changed for the adaptive framework, only the values of individual parameters, yet these adjustments were able to produce very different behaviour and performance.

Despite the excellent performance of the base PSO-DRS algorithm on the assigned benchmark, this

	Adapting	**SPSO** *Fixed*	**DRS** *Fixed*	**DRS** ***NP+K+ϕ: R2***	**DRS** ***NP+ϕ: R3***
f_{10}	Success rate	**100%**	98%	**100%**	98%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	0.0163±0.0163	**0.0±0.0**	0.0163±0.0163
	Worst	**0.0**	0.816	**0.0**	0.816
	FEvals±StE	13528±228	**5622±122**	21097±4609	6743±279
f_{11}	Success rate	**100%**	98%	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	1.62±1.62	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	81.0	**0.0**	**0.0**
	FEvals±StE	9313±81	**6198±134**	14424±7875	6546±211
f_{12}	Success rate	86%	**98%**	76%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.708±0.251	0.149±0.149	1.59±0.412	0.751±0.297
	Worst	5.10	7.47	7.47	7.47
	FEvals±StE	21751±3860	**16839±1415**	22225±7581	19012±1539
f_{13}	Success rate	88%	**98%**	**98%**	96%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.823±0.323	**0.134±0.134**	**0.134±0.134**	0.267±0.187
	Worst	7.64	**6.68**	6.74	**6.68**
	FEvals±StE	28871±12526	31861±2111	15454±823	**21322±1063**
f_{14}	Success rate	90%	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.759±0.326	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	8.11	**0.0**	**0.0**	**0.0**
	FEvals±StE	17690±861	25119±1795	**14762±996**	24129±2343

Table 6.13: Results for **best-performing** adaptive rules vs SPSO and PSO-DRS on simple multimodal problems

performance cannot necessarily be expected for other applicable types of problem. While continuing improvements to the base algorithm could be researched and applied, as has been the case with most variations to major optimization algorithms, each of these improvements would likewise be constrained by the same stipulation. What is more useful is a separate system for adjusting the algorithm according to the features of the problem or problems under optimization, independent of the functioning of the algorithm itself.

The Lennard-Jones problem, in its various configurations, provides a suite of functions on which a fixed-parameter PSO-DRS shows relatively poor performance. By determining adaptive rules and configurations that are able to maintain the excellent PSO-DRS performance on the other 14 optimization problems as well as attain superior results on Lennard-Jones, it is demonstrated that PSO-DRS, itself an extension and step beyond PSO, can be extended for general applicability to problems beyond the standard range of optimization benchmarks.

What is important here are not the specific rules that were used to obtain good results, but the framework of adaptability that has been established. Adjustment of the points of adaptation, *NP*, *K*, and ϕ, is shown to have a measurable effect on the ability of the swarm to converge to an optimum point. The rules used here are examples of behaviours that can be implanted into the swarm to effect a dynamic, responsive convergence rate throughout the entire process of optimization.

This combination of concept and application represents a reasoned approach to further development of the PSO-DRS algorithm. Rather than simply presenting a new variation to the basic fixed-parameter

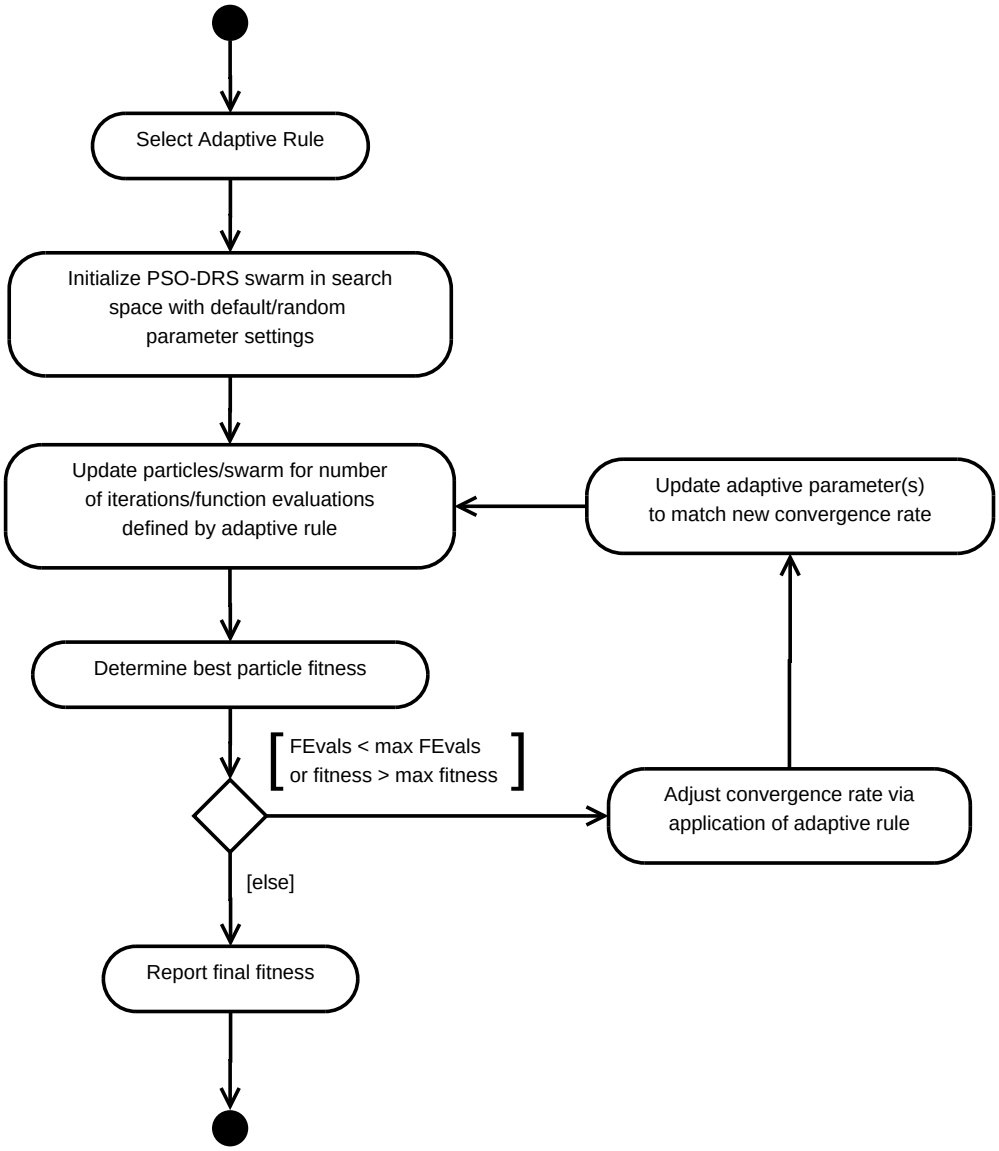


Figure 6.17: Adaptive PSO-DRS framework

algorithm without examining the possible methods of implementing changes, defining the points of adaptation and demonstrating the effects of a spectrum of adaptive rules allowed for a better understanding of the resulting behaviour. The effects of these behaviours could then be seen in the application of the new swarm configurations in the performance results: slow-converging rule 1a-adapted swarms were less effective than fast-converging rule 1b-adapted swarms, while rule 2- and rule 3- adapted swarms compromised between the two and showed the best performance overall.

The improved ability of many of the adapted swarms using these example rules on both the standard and the extended problem space is evidence that this method of adaptation is part of a fruitful area of research. Using PSO-DRS as a base, we can apply conditional adaptations using this framework to build optimizers as needed that are suited to specific circumstances and practicalities at will. This valuable

avenue of development broadens the range of PSO research to embrace diverse applications while retaining all of the core features of the algorithm that have made it the popular, outstanding optimizer that it is today.

6.10 Conclusions

This chapter has focused on the development of an framework for adaptively adjusting the parameters, and hence the behaviour, of the PSO-DRS algorithm. These adaptations are determined by pre-defined rules that are designed to adjust the parameters based on conditions taken from reported performance at points of the optimization process that vary from simply whether or not performance is improving, to how quickly the improvement is taking place, if at all.

As a first step, the adaptable parameters of the swarm algorithm were defined. These consisted of the *swarm-level* settings of *NP* and *K*, and the *particle-level* setting ϕ. As these three parameters control the composition of the swarm, the interaction between individual particles, and the movement of those particles, their values are integral to the functioning of the algorithm.

The performance of the adapted form of PSO-DRS was compared to that of SPSO and the fixed-parameter form of PSO-DRS. Two approaches were taken: single-parameter adaptation, and multiple-parameter adaptation. On the benchmark previously defined in chapter 3 and used throughout the work, the adaptive system was able in several configurations to obtain results equivalent to those of the best fixed-parameter PSO-DRS algorithm without the need for pre-selecting one, two, or all three of the parameters.

Another, potentially more significant advantage of this framework lies in its extension of the range of problems that PSO-DRS can optimize effectively. The Lennard-Jones problem was introduced, and it was shown how the adaptive framework brought about great improvements to PSO-DRS performance across all configurations of the problem.

Through examination of the behaviour and performance of the adapted PSO-DRS swarm algorithm, we can conclude that the framework developed here improves both the performance and the applicability of PSO-DRS.

	Adapting	**Random Search**	**SPSO** *Fixed*	**DRS** *Fixed*	**DRS** ***NP+K+ϕ: R2***	**DRS** ***NP+ϕ: R3***
LJ_2	Success rate	0%	**100%**	**100%**	**100%**	**100%**
	Best	1.06E-7	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.118±0.023	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	0.591	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals±StE	-	8979±245	7601±753	8807±932	**7196±745**
LJ_3	Success rate	0%	**100%**	22%	98%	80%
	Best	0.436	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±0.059	**0.0±0.0**	3.2E-6±1.47E-6	1.4E-7±1.4E-7	1.5E-6±8.5E-7
	Worst	2.22	**0.0**	7.2e-5	6.98E-6	3.78E-5
	FEvals±StE	-	**16208±239**	417959±36851	64379±11222	101060±18613
LJ_4	Success rate	0%	**100%**	20%	98%	92%
	Best	2.23	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	4.05±0.074	**0.0±0.0**	2.7E-5±1.5E-5	5.9E-6±5.9E-6	1.8E-5±1.8E-5
	Worst	4.97	**0.0**	7.2E-4	2.97E-4	8.9E-4
	FEvals±StE	-	**35990±1312**	361428±27471	86547±12957	119381±12254
LJ_5	Success rate	0%	**100%**	6%	96%	84%
	Best	9.1	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	9.1±5.3E-7	**0.0±0.0**	1.2E-3±3.9E-4	1.5E-2±1.3E-2	8.4E-6±5.1E-6
	Worst	9.1	**0.0**	1.4E-2	6.3E-1	2.25E-4
	FEvals±StE	-	**58204±1921**	522557±57857	83811±15056	187843±13722
LJ_6	Success rate	0%	2%	0%	**10%**	2%
	Best	7.22	**0.0**	3.2E-4	**0.0**	**0.0**
	Mean±StE	9.44±0.102	0.4±8.2E-3	**0.335±0.023**	0.406±0.034	0.371±0.02
	Worst	10.73	**0.409**	0.53	1.61	0.66
	FEvals±StE	-	387432±0	-	**69145±11436**	168239±0
LJ_7	Success rate	0%	40%	0%	**42%**	8%
	Best	10.41	**0.0**	2.3E-3	**0.0**	**0.0**
	Mean±StE	12.67±0.103	0.541±0.098	0.779±0.091	0.525±0.067	**0.476±0.079**
	Worst	13.69	4.2	2.99	**1.52**	1.97
	FEvals±StE	-	152420±19186	-	**115489±27942**	346136±72266
LJ_8	Success rate	0%	**78%**	0%	50%	12%
	Best	19.82	**0.0**	6.4E-3	**0.0**	**0.0**
	Mean±StE	19.82±5.9E-7	**0.118±0.044**	1.32±0.133	0.456±0.128	0.522±0.137
	Worst	19.82	**1.04**	4.06	5.87	5.23
	FEvals±StE	-	240278±18703	-	**107250±10164**	333654±75485
LJ_9	Success rate	0%	**28%**	0%	12%	4%
	Best	16.54	**0.0**	1.15	**0.0**	**0.0**
	Mean±StE	18.96±0.136	**0.865±0.116**	3.89±0.2	1.04±0.075	1.57±0.287
	Worst	20.41	4.35	7.92	**2.03**	8.73
	FEvals±StE	-	239197±36321	-	166648±79646	316426±49634
LJ_{10}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	19.63	1.1E-4	0.143	**1.1E-7**	**1.1E-7**
	Mean±StE	22.69±0.17	1.96±0.22	6.33±0.32	**1.93±0.425**	2.54±0.354
	Worst	25.65	**6.39**	11.33	14.98	11.76
	FEvals±StE	-	-	-	-	-
LJ_{11}	Success rate	0%	**10%**	0%	6%	0%
	Best	22.95	**0.0**	5.32	**0.0**	0.002
	Mean±StE	26.44±0.197	2.67±0.35	9.9±0.3	**2.32±0.34**	3.31±0.32
	Worst	29.02	11.15	13.93	15.83	**8.63**
	FEvals±StE	-	274779±48157	-	299412±43412	-

Table 6.14: Results for **best-performing** adaptive rules vs SPSO and PSO-DRS on Lennard-Jones problems 2–11

	Adapting	**Random Search**	**SPSO** *Fixed*	**DRS** *Fixed*	**DRS** ***NP+K+ϕ: R2***	**DRS** ***NP+ϕ: R3***
LJ_{12}	Success rate	0%	**0%**	**0%**	**0%**	**0%**
	Best	26.05	1.72	10.78	**4.4E-7**	1.73
	Mean±StE	31.8±0.323	5.17±0.72	14.8±0.28	**3.74±0.642**	6.92±0.78
	Worst	37.41	36.97	**19.36**	26.0	21.05
	FEvals±StE	-	-	-	-	-
LJ_{13}	Success rate	0%	0%	0%	**4%**	0%
	Best	33.83	**3.71**	13.73	**0.0**	3.89
	Mean±StE	38.52±0.411	10.96±1.46	20.55±0.36	**6.03±0.603**	9.57±0.8
	Worst	44.33	43.33	**24.28**	27.34	26.08
	FEvals±StE	-	-	-	326334±62880	-
LJ_{14}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	35.08	2.88	18.8	**2.17E-7**	4.01
	Mean±StE	42.03±0.567	19.32±2.55	24.15±0.31	**6.6±0.61**	13.93±1.15
	Worst	47.85	46.85	29.8	**28.14**	29.98
	FEvals±StE	-	-	-	-	-
LJ_{15}	Success rate	0%	0%	0%	**2%**	0%
	Best	39.93	3.09	23.14	**0.0**	1.91
	Mean±StE	47.86±0.589	34.36±2.75	28.32±0.32	**6.1±0.491**	15.44±1.28
	Worst	52.32	51.32	32.09	**17.1**	34.03
	FEvals±StE	-	-	-	179019±0	-
LJ_{16}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	45.20	0.912	25.8	**1.7E-6**	2.5
	Mean±StE	53.6±0.581	39.08±2.88	32.44±0.35	**7.41±0.542**	18.09±1.52
	Worst	56.82	55.82	37.62	**20.56**	40.81
	FEvals±StE	-	-	-	-	-
LJ_{17}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	42.02	4.02	30.52	**0.011**	4.13
	Mean±StE	59.81±0.53	49.69±2.8	37.13±0.41	**8.12±0.972**	22.53±1.63
	Worst	61.32	60.32	**42.2**	43.25	44.94
	FEvals±StE	-	-	-	-	-
LJ_{18}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	56.71	6.57	38.82	**0.246**	1.68
	Mean±StE	65.78±0.329	58.45±2.54	43.23±0.26	**12.96±1.36**	25.79±1.88
	Worst	66.53	65.53	**47.29**	50.58	50.57
	FEvals±StE	-	-	-	-	-
LJ_{19}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	61.93	19.05	44.26	3.55	**3.25**
	Mean±StE	72.03±0.292	69.43±1.05	49.28±0.34	**14.08±0.94**	25.12±1.99
	Worst	72.66	71.66	53.42	**31.78**	53.13
	FEvals±StE	-	-	-	-	-
LJ_{20}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	77.16	8.23	45.13	**3.23**	9.4
	Mean±StE	77.18±2.99E-4	72.8±1.6	53.59±0.41	**16.78±1.32**	34.55±2.2
	Worst	77.18	76.18	**58.92**	61.49	60.17
	FEvals±StE	-	-	-	-	-
LJ_{26}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	108.3	93.11	**80.37**	13.04	
	Mean±StE	108.3±2.99E-4	103.02±0.5	84.92±0.29	**31.63±1.89**	57.86±2.61
	Worst	108.3	107.32	**88.98**	91.88	90.99
	FEvals±StE	-	-	-	-	-
LJ_{38}	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	173.7	141.6	145.2	**39.1**	81.25
	Mean±StE	173.9±0.004	162.5±0.97	150.3±0.26	**75.3±2.76**	127.06±3.29
	Worst	173.9	172.9	153.8	**152.95**	154.31
	FEvals±StE	-	-	-	-	-

Table 6.15: Results for **best-performing** adaptive rules vs SPSO and PSO-DRS on Lennard-Jones problems 12–20, 26, and 38

Chapter 7

Conclusions

7.1 Future Work

As with any research, many potential directions for future work beyond the scope presented here became apparent during the course of this thesis. A summary of these possibilities are summarized in this section.

Updates to the Standard PSO

While the advances to the original PSO algorithm[1, 2] necessitated the definition of a new standard form of the algorithm, as described in chapter 3, there should of course be no illusions that this standard is any more permanent than the original. New advances are constantly proposed and published that extend and improve PSO – the more generic of these can eventually become part of the canon, and contribute to the formulation of an updated standard.

These advances could take many forms, from improved settings for the parameters χ, c_1, and c_2, to a more analytical approach to determining the optimal number of particles in the swarm. The standard defined here drew from major, well-known publications and empirical results in its method of combining advances into a cohesive algorithm, but there should be no doubt that even these major steps forward can be overtaken by research building on their work. When those new advances appear, an updated standard should be expected to take them into account in pushing forward the canonical form of PSO for representation in the optimization community.

Full Sampling Distribution of PSO-DR

While the first two moments, corresponding to the mean and the variance, of the general PSO-DR sampling distribution were derived here in chapter 5, the method introduced by Poli[76, 77, 78] and used in this work contains no restrictions on determining higher moments. Accordingly, Poli determined the first four moments of a PSO equivalent to the standard, adding skew and kurtosis, and suggested that the method could potentially be used to determine the full sampling distribution.

Given the identical nature of the stability regions for the mean and variance of PSO-DR, it would be particularly interesting to see whether this extends to the higher moments. Using Poli's method makes

it relatively straightforward to extend the derivation to encompass further moments, and if the noted similarities hold for all higher moments, the full sampling distribution of PSO-DR could be exactly obtained.

Improved and Specialized Adaptive Rules

The four adaptive rules used in chapter 6 were used both as a prototypical system of adaptation, and as evidence of the applicability of the framework for optimizing previously unsuitable problem spaces, in this case the Lennard-Jones set. These rules are given as examples, not as integral parts of the adaptive PSO-DRS algorithm. Nothing prevents their replacement with alternate rules intended for any desired direction of research, be it into performance, behaviour, or some other aspect of optimization.

One of the major goals of this thesis has been to introduce an adaptive system that can be used in many contexts without necessitating any alterations to the core PSO/PSO-DR/PSO-DRS functionality. The nature of the adaptive rules allows them to be constructed and applied to the algorithm independently of the rest of the optimization process. With this ability, the development of new adaptive rules is the obvious next step.

7.2 Summary

The future for development of the PSO algorithm is bright and wide open; the concept and field are still relatively new, leaving many areas as yet unexplored. New publications appear with nearly every conference and journal on evolutionary algorithms and swarm intelligence. A cursory Internet search reveals articles and tutorials aimed at all levels of users, lecture notes for undergraduate algorithms courses, and thousands of publications all referring back to the original proposal. This interest stems from both the excellent performance over many types of problems, and the attractive familiarity of the concepts of swarms. Even the layman can be quickly educated on the functioning of a swarm optimizer by means of real-world biological comparisons.

In section 1.3, the chief objective of this thesis was established as:

> . . . to present the concepts and implementations involved in simplifying the established PSO algorithm, allowing for the introduction of an adaptive aspect that is broadly applicable and unreliant on alterations to the core algorithm.

This goal has been achieved by means of a process of iteratively building on the original PSO formulation[1, 2], as outlined in this summary.

While many interesting and valuable advances have been made in the field, much of the work that has been done has consisted of minor variations to the functionality of the algorithm. This is important research, expanding the literature and deepening the pool of research, but what is truly valuable is the less common study that examines the fundamental components of the concept and builds on them. Alternative

topologies, the effects of inertia weight and constriction on preventing divergence, mathematical stability analyses, and the like were all such advances, taking the original work and stepping forward with it into new formulations that must now for the sake of modernity be applied to any use of the algorithm. Each of these significantly improves the functioning of the swarm, and by collecting the most important components into a single new standard formulation, we not only obtain much-improved performance, we also ensure that future research will be working from the same single origin, which is vital for the development of new, similarly fundamental advances.

Simplification of the optimization process is one such advance. A standard formulation is a beginning, not the final word on PSO. Questions remain about the functionality and behaviour of the algorithm, some of which are made more difficult to answer by the complex interactions between components at both the levels of individual particles and the swarm as a whole. Multiplicative stochasticity leads to velocity bursts, which appear unlikely to benefit the search for improved solutions, even hindering it by means of wasted time and processing that could be better applied elsewhere. The previous velocity component, already suspected of being vestigial from the time that PSO was used to simulate graphically-appealing swarm simulations, can be removed. Under appropriate circumstances, the cognitive component of the update equations is unnecessary and can be removed, leading to a more social, better performing optimizer.

The modifications made here to PSO in the development of the PSO-DRS algorithm have not been concerned with bolting on new functionality or behaviour, but with actively removing and simplifying components of the algorithm down to a state where the same functionality and applicability remain, but those properties that have led to both conceptual and practical complexity have been stripped away. Performance remains equivalent to the standard PSO formulation, and even significantly improved in multiple instances, specifically the very difficult highly multimodal problems f_4–f_6 on which the standard algorithm often struggles.

With its first-order update equation, basic additive stochasticity, and single point of attraction, PSO-DRS is immediately accessible for a thorough mathematical analysis, which was used to show the proof of stability for appropriate selection of ϕ. This single tunable parameter caters to adaptability, obviating the need to balance multiple variables in order to find combinations that fall within the region of stability.

Such adaptation is made possible by the introduction of a conceptual framework that allows for quick development and testing of systems with changeable, adaptive behaviour. The clearly-defined points of adaptation at both the swarm and particle levels, along with pluggable rules or sets of rules for adjusting these parameters based on desired conditions, is both uncomplicated enough to be reproducible by any interested researcher, and open-ended enough to allow for nearly any sort of behaviour to be made a fundamental part of the optimization process.

Rather than unalterable, rigidly applied adaptations, this model encourages exploration of the ef-

fects of different approaches to adaptability. The benefits of this are seen by way of the significantly improved performance of various types of adapted PSO-DRS algorithms on a benchmark outside the typical range of global optimization testing, the Lennard-Jones atomic clustering problems. While the fixed-parameter PSO-DRS swarm is able to show excellent performance on the standard benchmark, results are very poor for all but the simplest atomic configurations. By tuning the parameters adaptively, however, we were able to obtain far improved performance on the majority of these problems without the need for a specially-constructed static formulation. The generic rules used to good effect here are only a small, demonstrative sample of the possible sets of conditions for tuning swarm and particle parameters, possibilities limited only by the range of applications.

Just as with the definition of a standard formulation of the PSO algorithm, what has been developed with this framework for adaptation is just a starting point for research into the ways that the concept can be extended and applied to many and various types of problems. By basing the adaptations of parameters on the rate of convergence, the effects of altering these settings are understandable and, just as importantly, predictable. The pluggable nature of the conditions for adaptation and their effects on the swarm opens up a new and separate area for research outside of the underlying PSO-DRS algorithm. Rather than fundamentally altering the algorithm and increasing the complexity or narrowing the applicability, rules can now be developed specifically for certain types of problems and introduced into the behaviour of the swarm while retaining the defining characteristics of PSO.

All of the concepts and features explored in this work have led to something that is not a reimagining of PSO, but a refinement – a new conceptual and practical framework that embraces the ground-breaking past research that has brought the field to its present state of popularity and prominence, while moving it forward on the path to a future as an adaptive system of optimization. This direction is one that can and will be explored for the future development of particle swarm optimization.

Appendix A

Results tables

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_1	Success rate	**100%**	0%	**100%**	**100%**
	Best	**0.0**	1.21E-13	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	1.29E-12±2.04E-13	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	6.91E-12	**0.0**	**0.0**
	FEvals±StE	76748±847	-	136301±2686	145548±8414
f_2	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	**3.61E-5**	105.6	9.06	601.3
	Mean±StE	**3.36E-3±1.21E-3**	302.6±15.74	85.51±10.64	2445±120.7
	Worst	**5.14E-2**	539.2	311.7	4514
	FEvals±StE	-	-	-	-
f_3	Success rate	0%	0%	0%	0%
	Best	2.72E-5	7.64	4.32E-4	9.82
	Mean±StE	8.48±1.18	22.72±1.32	4.44±0.51	25.67±1.99
	Worst	33.1	69.3	11.21	105.8
	FEvals±StE	-	-	-	-

Table A.1: Results for single-point adaptive DRS using rule 1a on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_4	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	**829**	1692	2369	2053
	Mean±StE	**1576±39**	2004±17	3300±184	2975±82
	Worst	**2013**	2257	7063	4207
	FEvals±StE	-	-	-	-
f_5	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	2.98	**5.49**	8.95	88.79
	Mean±StE	**9.19±0.64**	19.5±0.82	17.21±0.5	117.6±2.07
	Worst	**21.89**	30.88	27.86	152.3
	FEvals±StE	-	-	-	-
f_6	Success rate	**100%**	0%	**100%**	88%
	Best	**0.0**	6.65E-7	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	4.69E-6±4.52E-7	**0.0±0.0**	1.09E-7±1.08E-7
	Worst	**0.0**	1.36E-5	**0.0**	5.4E-6
	FEvals±StE	226790±9485	-	**190987±1121**	505540±5523
f_7	Success rate	**94%**	0%	90%	58%
	Best	**0.0**	2.56E-11	**0.0**	**0.0**
	Mean±StE	4.44E-4±2.51E-4	**3.87E-10±7.03E-11**	7.4E-4±3.17E-4	2.99E-4±2.07E-4
	Worst	7.4E-3	**2.34E-9**	7.4E-3	7.4E-3
	FEvals±StE	70276±704	-	**56260±212**	276203±27278
f_8	Success rate	96%	0%	**100%**	**100%**
	Best	**0.0**	4.13E-9	**0.0**	**0.0**
	Mean±StE	4.15E-3±2.9E-3	6.45E-8±2.01E-8	**0.0±0.0**	**0.0±0.0**
	Worst	0.104	9.79E-7	**0.0**	**0.0**
	FEvals±StE	95725±630	-	168130±2494	304209±4092
f_9	Success rate	98%	0%	92%	**100%**
	Best	**0.0**	2.47E-8	**0.0**	**0.0**
	Mean±StE	2.2E-4±2.2E-4	1.76E-7±3.08E-8	8.79E-4±4.26E-4	**0.0±0.0**
	Worst	1.1E-2	1.01E-6	1.1E-2	**0.0**
	FEvals±StE	**91635±454**	-	164769±2055	297647±5891

Table A.2: Results for single-point adaptive DRS using rule 1a on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_{10}	Success rate	98%	**100%**	82%	94%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.0163±0.0163	**0.0±0.0**	0.15±0.045	0.049±0.028
	Worst	0.816	**0.0**	0.816	0.816
	FEvals±StE	**5622±122**	8698±509	13445±1552	58580±11946
f_{11}	Success rate	98%	**100%**	92%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	**0.0±0.0**	6.48±3.14	**0.0±0.0**
	Worst	81.0	**0.0**	81.0	**0.0**
	FEvals±StE	6198±134	16388±477	**6100±456**	7478±334
f_{12}	Success rate	98%	**100%**	60%	66%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.149±0.149	**0.0±0.0**	2.77±0.492	2.35±0.475
	Worst	7.47	**0.0**	7.52	7.47
	FEvals±StE	**16839±1415**	112444±14197	30448±10859	22067±7548
f_{13}	Success rate	98%	**100%**	62%	90%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.134±0.134	**0.0±0.0**	2.42±0.45	0.506±0.246
	Worst	6.68	**0.0**	7.65	6.68
	FEvals±StE	31861±2111	88428±3971	**29395±13917**	69831±11918
f_{14}	Success rate	**100%**	**100%**	80%	98%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	1.31±0.378	4.26E-16±4.26E-16
	Worst	**0.0**	**0.0**	7.79	2.13E-14
	FEvals±StE	25119±1795	78361±3099	**12524±720**	63323±11173

Table A.3: Results for single-point adaptive DRS using rule 1a on simple multimodal problems

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP**	**PSO-DRS** **K**	**PSO-DRS** **PHI**
LJ_2	Success	**100%**	6%	0%	4%
	Best	**0.0**	**0.0**	1.08E-13	**0.0**
	Mean±StE	**0.0±0.0**	2.51E-11±6.52E-12	4.60E-10±1.88E-10	1.90E-10±4.85E-11
	Worst	**0.0**	2.34E-10	7.85E-09	1.69E-09
	Fevals±StE	**7601±753**	379856±125095	-	258896±98697
LJ_3	Success	**22%**	0%	0%	0%
	Best	**0.0**	4.35E-09	6.76E-05	7.40E-05
	Mean±StE	**3.20E-06±1.47E-06**	5.37E-06±8.77E-07	0.003±3.13E-04	0.002±1.95E-04
	Worst	7.25E-05	**2.29E-05**	0.010	0.007
	Fevals±StE	**417959±36851**	-	-	-
LJ_4	Success	**20%**	0%	2%	0%
	Best	**0.0**	5.66E-06	**0.0**	0.001
	Mean±StE	**2.70E-05±1.46E-05**	3.21E-04±6.18E-05	0.322±0.027	0.154±0.016
	Worst	**7.17E-04**	0.002	0.678	0.536
	Fevals±StE	**361428±27471**	-	453500±0	-
LJ_5	Success	**6%**	0%	0%	0%
	Best	**0.0**	1.60E-04	0.950	0.315
	Mean±StE	**0.001±3.88E-04**	0.023±0.005	1.63±0.048	1.31±0.069
	Worst	**0.014**	0.174	2.46	2.32
	Fevals±StE	**522557±57857**	-	-	-
LJ_6	Success	0%	0%	0%	0%
	Best	**3.22E-04**	0.334	2.75	1.33
	Mean±StE	**0.335±0.023**	0.663±0.037	3.78±0.068	3.61±0.104
	Worst	**0.530**	1.44	4.63	4.94
	Fevals±StE	-	-	-	-
LJ_7	Success	0%	0%	0%	0%
	Best	**0.002**	0.606	4.53	4.82
	Mean±StE	**0.779±0.091**	1.76±0.076	6.37±0.084	6.94±0.124
	Worst	**2.99**	3.47	7.30	8.37
	Fevals±StE	-	-	-	-
LJ_8	Success	0%	0%	0%	0%
	Best	**0.006**	1.38	7.00	8.00
	Mean±StE	**1.32±0.133**	3.57±0.165	8.70±0.091	9.89±0.098
	Worst	**4.06**	6.58	9.99	11.30
	Fevals±StE	-	-	-	-
LJ_9	Success	0%	0%	0%	0%
	Best	**1.15**	3.69	11.24	12.28
	Mean±StE	**3.89±0.203**	6.71±0.182	12.57±0.092	13.74±0.120
	Worst	**7.92**	9.48	14.07	15.65
	Fevals±StE	-	-	-	-
LJ_{10}	Success	0%	0%	0%	0%
	Best	**0.143**	6.83	13.80	15.41
	Mean±StE	**6.33±0.319**	10.26±0.205	16.49±0.110	17.77±0.127
	Worst	**11.33**	13.25	18.13	19.47
	Fevals±StE	-	-	-	-
LJ_{11}	Success	0%	0%	0%	0%
	Best	**5.32**	9.18	19.04	18.20
	Mean±StE	**9.90±0.300**	13.40±0.212	20.34±0.108	21.54±0.130
	Worst	**13.93**	16.30	21.93	22.75
	Fevals±StE	-	-	-	-

Table A.4: Results for single-point adaptive DRS using rule 1a on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP**	**PSO-DRS K**	**PSO-DRS PHI**
LJ_{12}	Success	0%	0%	0%	0%
	Best	**10.78**	10.98	23.18	23.62
	Mean±StE	**14.80±0.282**	17.78±0.307	25.50±0.133	26.73±0.137
	Worst	**19.36**	21.24	27.26	28.79
	Fevals±StE	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%
	Best	**13.73**	20.35	28.40	30.38
	Mean±StE	**20.55±0.366**	24.82±0.297	31.81±0.157	32.66±0.138
	Worst	**24.28**	28.11	34.76	34.22
	Fevals±StE	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%
	Best	**18.80**	22.16	32.97	32.86
	Mean±StE	**24.15±0.315**	27.83±0.231	35.35±0.163	35.85±0.181
	Worst	**29.80**	31.57	38.82	37.99
	Fevals±StE	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%
	Best	**23.14**	28.94	37.07	36.90
	Mean±StE	**28.32±0.322**	32.27±0.246	39.62±0.182	40.11±0.157
	Worst	**32.09**	35.60	43.22	41.85
	Fevals±StE	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%
	Best	**25.80**	31.66	41.63	41.96
	Mean±StE	**32.45±0.353**	36.49±0.260	44.56±0.162	44.53±0.116
	Worst	**37.62**	39.80	47.35	46.35
	Fevals±StE	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%
	Best	**30.52**	38.06	42.02	42.02
	Mean±StE	**37.13±0.408**	41.13±0.239	48.81±0.209	48.42±0.236
	Worst	**42.20**	44.76	51.82	51.14
	Fevals±StE	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%
	Best	38.82	**36.76**	51.05	51.26
	Mean±StE	**43.23±0.261**	45.93±0.371	54.42±0.169	53.84±0.155
	Worst	**47.29**	49.63	57.22	56.27
	Fevals±StE	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%
	Best	**44.26**	47.54	57.39	55.56
	Mean±StE	**49.28±0.336**	51.66±0.284	60.90±0.173	59.57±0.167
	Worst	**53.42**	55.37	63.33	61.76
	Fevals±StE	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%
	Best	**45.13**	52.94	61.79	60.54
	Mean±StE	**53.59±0.409**	56.70±0.214	65.10±0.217	63.70±0.185
	Worst	**58.92**	59.41	68.31	66.33
	Fevals±StE	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%
	Best	**80.37**	80.84	89.55	87.79
	Mean±StE	**84.92±0.285**	87.33±0.306	96.25±0.277	93.23±0.221
	Worst	**88.98**	90.69	99.04	95.58
	Fevals±StE	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%
	Best	145.2	146.6	153.0	**143.7**
	Mean±StE	**150.3±0.265**	152.5±0.314	161.2±0.323	155.4±0.349
	Worst	**153.8**	156.3	164.7	158.9
	Fevals±StE	-	-	-	-

Table A.5: Results for single-point adaptive DRS using rule 1a on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_1	Success rate	**100%**	**100%**	0%	10%	0%
	Best	**0.0**	**0.0**	7.08E-9	**0.0**	1.36E-5
	Mean±StE	**0.0±0.0**	**0.0±0.0**	7.41E-7±1.51E-7	27.53±12.95	67.95±29.86
	Worst	**0.0**	**0.0**	4.84E-6	592.9	1028
	FEvals±StE	76748±847	164468±1602	-	**52258±1842**	-
f_2	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**3.61E-5**	1.82	1885	29130	24374
	Mean±StE	**3.36E-3±1.21E-3**	14.44±1.61	5708±221	46183±1476	45266±1392
	Worst	**5.14E-2**	68.92	9961	68635	65526
	FEvals±StE	-	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**2.72E-5**	0.426	25.22	239	132
	Mean±StE	**8.48±1.18**	47.76±4.67	28.85±0.395	1.58E6±8.09E5	4.78E6±1.96E6
	Worst	**33.1**	89.56	38.59	2.7E7	6.38E7
	FEvals±StE	-	-	-	-	-

Table A.6: Results for multi-point adaptive DRS using rule 1a on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_4	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**829**	2961	2396	8900	8717
	Mean±StE	**1576±39**	4848±227	2917±51	9711±48	9456±43
	Worst	**2013**	7286	3687	10505	10018
	FEvals±StE	-	-	-	-	-
f_5	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**2.98**	87.94	118.1	227.0	220.7
	Mean±StE	**9.19±0.64**	138.3±2.83	138.1±1.37	343.4±4.62	341.1±4.1
	Worst	**21.89**	190.2	156.1	398.3	383.3
	FEvals±StE	-	-	-	-	-
f_6	Success rate	**100%**	0%	0%	0%	0%
	Best	**0.0**	7.97E-10	4.35E-5	15.55	18.09
	Mean±StE	**0.0±0.0**	3.68E-9±3.14E-10	3.03E-4±3.56E-5	19.0±0.14	19.42±0.05
	Worst	**0.0**	9.93E-9	1.2E-3	20.4	20.05
	FEvals±StE	**226790±9485**	-	-	-	-
f_7	Success rate	94%	**98%**	0%	0%	0%
	Best	**0.0**	**0.0**	5.43E-6	1.41E-3	2.82E-3
	Mean±StE	4.44E-4±2.51E-4	**1.48E-4±1.48E-4**	1.36E-3±3.98E-4	4.62±2.75	8.52±4.04
	Worst	**7.4E-3**	**7.4E-3**	1.79E-2	135.1	154.6
	FEvals±StE	**70276±704**	333355±2128	-	-	-
f_8	Success rate	**96%**	0%	0%	0%	0%
	Best	**0.0**	9.76E-11	3.33E-7	21.61	6.3E3
	Mean±StE	4.15E-3±2.9E-3	**2.99E-6±1.36E-6**	1.87E-4±6.13E-5	6.19E7±7.65E6	7.59E7±6.16E6
	Worst	1.04E-1	**5.71E-5**	3.02E-3	1.73E8	1.81E8
	FEvals±StE	**95725±630**	-	-	-	-
f_9	Success rate	**98%**	0%	0%	0%	0%
	Best	**0.0**	4.24E-10	2.1E-5	33.56	1.54E4
	Mean±StE	2.2E-4±2.2E-4	**2.33E-5±1.89E-5**	1.4E-3±3.4E-4	1.45E8±1.81E7	1.91E8±1.71E7
	Worst	0.011	**9.35E-4**	1.3E-2	6.14E8	5.1E8
	FEvals±StE	**91635±454**	-	-	-	-

Table A.7: Results for multi-point adaptive DRS using rule 1a on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_{10}	Success rate	**98%**	88%	**98%**	34%	20%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.6E-2±1.6E-2	9.8E-3±3.8E-2	**3.2E-16±3.2E-16**	2.4E-1±7.6E-2	1.1E-13±2.9E-14
	Worst	0.816	0.816	**1.58E-14**	3.14	1.08E-12
	FEvals±StE	**5622±122**	21748±1705	123369±17826	198276±23250	275697±26338
f_{11}	Success rate	98%	96%	**100%**	88%	98%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	2.30±1.74	**0.0±0.0**	8.42±3.47	1.62±1.62
	Worst	81.0	81.0	**0.0**	81.0	81.0
	FEvals±StE	**6198±134**	18123±1257	12992±695	12833±5482	25968±2268
f_{12}	Success rate	98%	64%	86%	58%	62%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.149±0.149	2.45±0.48	**2.1E-7±2.1E-7**	2.80±0.48	1.70±0.421
	Worst	7.47	7.52	**1.05E-5**	7.47	7.52
	FEvals±StE	16839±1415	62637±14066	104590±20282	**13148±1301**	100888±19675
f_{13}	Success rate	98%	76%	**100%**	60%	72%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.134±0.134	1.19±0.34	**0.0±0.0**	2.42±0.43	1.13±0.286
	Worst	6.68	7.84	**0.0**	7.64	7.64
	FEvals±StE	**31861±2111**	78885±11785	89895±7795	47855±17148	86769±6290
f_{14}	Success rate	**100%**	86%	**100%**	58%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	0.656±0.25	**0.0±0.0**	2.62±0.45	0.645±0.279
	Worst	**0.0**	8.11	**0.0**	8.11	7.81
	FEvals±StE	25119±1795	56843±2476	93287±6645	**17441±396**	101840±9686

Table A.8: Results for multi-point adaptive DRS using rule 1a on simple multimodal problems

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_2	Success	**100%**	0%	0%	0%	0%
	Best	**0.0**	1.70E-13	9.83E-14	3.81E-11	1.01E-12
	Mean±StE	**0.0±0.0**	3.9E-10±7.5E-11	4.4E-10±1.5E-10	2.1E-07±8.0E-08	4.0E-07±1.3E-07
	Worst	**0.0**	2.28E-09	6.68E-09	2.92E-06	4.56E-06
	Fevals±StE	**7601±753**	-	-	-	-
LJ_3	Success	**22%**	0%	0%	0%	0%
	Best	**0.0**	2.68E-04	1.67E-04	0.024	0.024
	Mean±StE	**3.2E-06±1.5E-06**	0.006±6.4E-04	0.003±3.0E-04	0.140±0.019	0.234±0.025
	Worst	**7.25E-05**	0.022	0.012	0.851	0.774
	Fevals±StE	**417959±36851**	-	-	-	-
LJ_4	Success	**20%**	0%	0%	0%	0%
	Best	**0.0**	0.137	0.039	1.47	1.25
	Mean±StE	**2.7E-05±1.5E-05**	0.514±0.025	0.245±0.017	2.62±0.067	2.82±0.055
	Worst	**7.17E-04**	0.942	0.608	3.35	3.41
	Fevals±StE	**361428±27471**	-	-	-	-
LJ_5	Success	**6%**	0%	0%	0%	0%
	Best	**0.0**	1.10	0.413	4.40	3.92
	Mean±StE	**0.001±3.9E-04**	1.88±0.054	1.61±0.060	5.55±0.077	5.40±0.073
	Worst	**0.014**	3.01	2.67	6.38	6.17
	Fevals±StE	**522557±57857**	-	-	-	-
LJ_6	Success	0%	0%	0%	0%	0%
	Best	**3.22E-04**	1.89	2.69	7.22	6.64
	Mean±StE	**0.335±0.023**	4.19±0.075	4.43±0.092	8.73±0.076	8.42±0.088
	Worst	**0.530**	5.08	5.75	9.57	9.57
	Fevals±StE	-	-	-	-	-
LJ_7	Success	0%	0%	0%	0%	0%
	Best	**0.002**	5.67	6.11	10.12	10.29
	Mean±StE	**0.779±0.091**	7.14±0.077	7.52±0.091	12.05±0.093	11.65±0.083
	Worst	**2.99**	8.12	8.83	12.96	12.93
	Fevals±StE	-	-	-	-	-
LJ_8	Success	0%	0%	0%	0%	0%
	Best	**0.006**	8.35	9.10	12.29	12.29
	Mean±StE	**1.32±0.133**	9.89±0.097	10.65±0.078	14.87±0.096	14.34±0.106
	Worst	**4.06**	11.38	11.72	16.18	15.41
	Fevals±StE	-	-	-	-	-
LJ_9	Success	0%	0%	0%	0%	0%
	Best	**1.15**	11.80	12.34	16.54	16.07
	Mean±StE	**3.89±0.203**	13.89±0.113	14.48±0.109	18.76±0.118	17.97±0.111
	Worst	**7.92**	15.73	16.01	20.33	19.81
	Fevals±StE	-	-	-	-	-
LJ_{10}	Success	0%	0%	0%	0%	0%
	Best	**0.143**	15.65	17.35	19.63	18.80
	Mean±StE	**6.33±0.319**	17.80±0.111	18.77±0.088	22.38±0.146	21.49±0.145
	Worst	**11.33**	19.76	20.02	24.34	23.05
	Fevals±StE	-	-	-	-	-
LJ_{11}	Success	0%	0%	0%	0%	0%
	Best	**5.32**	19.81	20.39	22.95	21.06
	Mean±StE	**9.90±0.300**	22.17±0.142	22.36±0.129	26.10±0.163	25.28±0.169
	Worst	**13.93**	24.26	24.14	27.83	27.37
	Fevals±StE	-	-	-	-	-

Table A.9: Results for multi-point adaptive DRS using rule 1a on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_{12}	Success	0%	0%	0%	0%	0%
	Best	**10.78**	24.42	24.87	26.05	26.05
	Mean±StE	**14.80±0.282**	27.80±0.138	27.59±0.116	31.08±0.199	29.78±0.175
	Worst	**19.36**	29.54	29.00	33.43	32.06
	Fevals±StE	-	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%	0%
	Best	**13.73**	32.66	29.83	33.83	33.83
	Mean±StE	**20.55±0.366**	34.22±0.107	33.71±0.147	36.96±0.181	36.02±0.163
	Worst	**24.28**	36.26	35.64	38.93	38.40
	Fevals±StE	-	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%	0%
	Best	**18.80**	35.08	34.43	35.08	35.08
	Mean±StE	**24.15±0.315**	37.70±0.167	36.91±0.139	39.74±0.242	38.61±0.225
	Worst	**29.80**	40.01	39.68	42.30	41.34
	Fevals±StE	-	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%	0%
	Best	**23.14**	39.29	35.46	39.93	39.29
	Mean±StE	**28.32±0.322**	42.06±0.192	40.95±0.182	43.89±0.238	42.81±0.249
	Worst	**32.09**	45.44	43.04	46.77	45.94
	Fevals±StE	-	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%	0%
	Best	**25.80**	42.31	42.31	44.61	42.31
	Mean±StE	**32.45±0.353**	47.23±0.211	45.63±0.139	48.24±0.216	47.80±0.255
	Worst	**37.62**	49.59	47.09	50.78	50.45
	Fevals±StE	-	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%	0%
	Best	**30.52**	42.02	42.02	42.02	42.02
	Mean±StE	**37.13±0.408**	51.11±0.307	49.48±0.234	52.31±0.295	51.47±0.349
	Worst	**42.20**	53.63	51.91	55.70	55.70
	Fevals±StE	-	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%	0%
	Best	**38.82**	49.48	49.48	52.49	49.48
	Mean±StE	**43.23±0.261**	56.23±0.257	54.48±0.180	57.55±0.240	57.08±0.302
	Worst	**47.29**	59.26	56.73	59.79	60.25
	Fevals±StE	-	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%	0%
	Best	**44.26**	59.43	57.25	57.60	59.42
	Mean±StE	**49.28±0.336**	62.78±0.192	60.63±0.166	62.96±0.231	62.87±0.226
	Worst	**53.42**	65.54	62.72	65.59	65.87
	Fevals±StE	-	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%	0%
	Best	**45.13**	61.57	61.57	61.57	61.57
	Mean±StE	**53.59±0.409**	67.34±0.216	64.81±0.168	67.70±0.243	67.51±0.270
	Worst	**58.92**	69.60	66.52	70.66	70.70
	Fevals±StE	-	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%	0%
	Best	**80.37**	93.63	89.74	94.12	93.91
	Mean±StE	**84.92±0.285**	97.12±0.228	94.33±0.234	96.87±0.199	97.18±0.217
	Worst	**88.98**	100.6	97.17	99.73	100.2
	Fevals±StE	-	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%	0%
	Best	**145.2**	155.7	147.1	154.1	152.2
	Mean±StE	**150.3±0.265**	161.4±0.324	156.7±0.379	159.9±0.315	160.1±0.420
	Worst	**153.8**	164.7	163.0	163.9	164.5
	Fevals±StE	-	-	-	-	-

Table A.10: Results for multi-point adaptive DRS using rule 1a on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_1	Success rate	**100%**	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals±StE	76748±847	98769±872	74862±180	**57177±182**
f_2	Success rate	0%	0%	0%	0%
	Best	3.61E-5	2.19E-7	2.33	1.06E-6
	Mean±StE	3.36E-3±1.21E-3	1.85E-5±1.01E-5	19.27±3.08	6.0E-4±2.14E-4
	Worst	5.14E-2	5.1E-4	127.3	8.32E-3
	FEvals±StE	-	-	-	-
f_3	Success rate	0%	0%	0%	0%
	Best	2.72E-5	2.8E-4	2.8E-2	1.15E-4
	Mean±StE	8.48±1.18	6.29±1.54	13.99±1.80	15.14±3.37
	Worst	33.1	69.7	70.43	79.55
	FEvals±StE	-	-	-	-

Table A.11: Results for single-point adaptive DRS using rule 1b on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_4	Success rate	0%	0%	0%	0%
	Best	829	2015	2369	3553
	Mean±StE	1576±39	3274±50	2942±33	3553±1.17E-13
	Worst	2013	3731	3435	3553
	FEvals±StE	-	-	-	-
f_5	Success rate	0%	0%	0%	0%
	Best	2.98	0.995	0.995	36.81
	Mean±StE	9.19±0.64	18.47±2.15	10.05±0.69	68.95±2.14
	Worst	21.89	73.63	24.87	102.5
	FEvals±StE	-	-	-	-
f_6	Success rate	**100%**	32%	94%	0%
	Best	**0.0**	**0.0**	**0.0**	0.931
	Mean±StE	**0.0±0.0**	0.383±0.383	8.08E-16±4.61E-16	16.29±0.41
	Worst	**0.0**	19.15	1.47E-14	18.50
	FEvals±StE	226790±9485	270183±10178	**215929±9757**	-
f_7	Success rate	94%	74%	**98%**	92%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	4.44E-4±2.51E-4	2.27E-3±5.67E-4	**1.48E-4±1.48E-4**	9.36E-4±4.61E-4
	Worst	**7.4E-3**	1.48E-2	**7.4E-3**	1.48E-2
	FEvals±StE	70276±704	106736±2908	88251±1352	**61719±383**
f_8	Success rate	**96%**	94%	94%	78%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.15E-3±2.9E-3**	6.22E-3±3.52E-3	6.22E-3±3.52E-3	4.77E-2±1.71E-2
	Worst	**0.104**	**0.104**	**0.104**	0.518
	FEvals±StE	95725±630	85459±2295	92602±730	**62120±456**
f_9	Success rate	98%	**100%**	92%	86%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	2.2E-4±2.2E-4	**0.0±0.0**	8.79E-4±4.26E-4	1.54E-3±5.45E-4
	Worst	1.1E-2	**0.0**	1.1E-2	1.1E-2
	FEvals±StE	91635±454	93815±1703	91166±358	**63097±266**

Table A.12: Results for single-point adaptive DRS using rule 1b on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_{10}	Success rate	98%	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.0163±0.0163	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	0.816	**0.0**	**0.0**	**0.0**
	FEvals±StE	5622±122	7374±352	7481±228	**4923±108**
f_{11}	Success rate	98%	84%	76%	78%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	12.65±4.15	34.56±17.06	17.82±4.79
	Worst	81.0	81.0	837.0	81.0
	FEvals±StE	6198±134	**4598±119**	5772±221	7676±218
f_{12}	Success rate	**98%**	86%	66%	**98%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.149±0.149**	0.537±0.235	2.39±0.48	**0.149±0.149**
	Worst	**7.47**	**7.47**	**7.47**	**7.47**
	FEvals±StE	16839±1415	13776±1053	**10986±708**	14290±694
f_{13}	Success rate	**98%**	96%	66%	84%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.134±0.134	0.175±0.125	2.27±0.45	**1.07±0.35**
	Worst	6.68	**5.27**	6.68	6.68
	FEvals±StE	31861±2111	**11516±291**	12876±615	14933±536
f_{14}	Success rate	**100%**	90%	76%	70%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	0.55±0.24	1.61±0.41	2.0±0.44
	Worst	**0.0**	6.7	6.7	7.67
	FEvals±StE	25119±1795	12108±453	14438±731	13516±543

Table A.13: Results for single-point adaptive DRS using rule 1b on simple multimodal problems

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP**	**PSO-DRS** **K**	**PSO-DRS** **PHI**
LJ_2	Success	**100%**	18%	4%	92%
	Best	0.0	0.0	0.0	0.0
	Mean±StE	**0.0±0.0**	5.90E-11±2.76E-11	1.00E-11±2.70E-12	2.34E-15±1.34E-15
	Worst	**0.0**	1.13E-09	1.22E-10	5.86E-14
	Fevals±StE	**7601±753**	75613±23599	378874±106717	161194±18415
LJ_3	Success	22%	2%	0%	**80%**
	Best	**0.0**	**0.0**	4.02E-09	**0.0**
	Mean±StE	3.20E-06±1.47E-06	1.37E-06±6.62E-07	3.53E-06±8.52E-07	**8.20E-14±3.36E-14**
	Worst	7.25E-05	2.59E-05	3.20E-05	**1.26E-12**
	Fevals±StE	417959±36851	**139282±0**	-	218902±21150
LJ_4	Success	20%	48%	0%	**92%**
	Best	**0.0**	**0.0**	4.39E-10	**0.0**
	Mean±StE	2.70E-05±1.46E-05	1.01E-04±7.11E-05	2.11E-05±1.11E-05	**3.61E-13±3.23E-13**
	Worst	7.17E-04	0.003	5.23E-04	**1.62E-11**
	Fevals±StE	361428±27471	**160318±16121**	-	291474±19172
LJ_5	Success	6%	84%	6%	**100%**
	Best	0.0	0.0	0.0	0.0
	Mean±StE	0.001±3.88E-04	5.86E-06±3.66E-06	5.97E-04±2.43E-04	**0.0±0.0**
	Worst	0.014	1.75E-04	0.012	**0.0**
	Fevals±StE	522557±57857	**86519±8671**	427684±52735	213245±14554
LJ_6	Success	0%	0%	0%	**8%**
	Best	3.22E-04	0.005	0.007	**0.0**
	Mean±StE	**0.335±0.023**	0.409±0.026	0.358±0.021	0.352±0.020
	Worst	0.530	1.52	0.685	**0.437**
	Fevals±StE	-	-	-	**323430±75045**
LJ_7	Success	0%	**14%**	0%	0%
	Best	0.002	**0.0**	7.09E-04	5.29E-08
	Mean±StE	0.779±0.091	0.829±0.078	0.886±0.085	**0.506±0.059**
	Worst	2.99	3.03	2.37	**1.09**
	Fevals±StE	-	**111735±29835**	-	-
LJ_8	Success	0%	**38%**	0%	0%
	Best	0.006	**0.0**	0.010	6.30E-06
	Mean±StE	1.32±0.133	0.641±0.153	1.36±0.135	**0.408±0.076**
	Worst	4.06	5.98	3.59	**1.90**
	Fevals±StE	-	**150535±19444**	-	-
LJ_9	Success	0%	**22%**	0%	0%
	Best	1.15	**0.0**	0.420	7.93E-04
	Mean±StE	3.89±0.203	1.74±0.259	3.47±0.212	**1.19±0.082**
	Worst	7.92	6.74	6.56	**2.27**
	Fevals±StE	-	**152720±26443**	-	-
LJ_{10}	Success	0%	0%	0%	0%
	Best	0.143	**1.07E-07**	2.22	0.003
	Mean±StE	6.33±0.319	2.69±0.362	6.20±0.260	**2.02±0.144**
	Worst	11.33	10.99	10.53	**4.94**
	Fevals±StE	-	-	-	-
LJ_{11}	Success	0%	0%	0%	0%
	Best	5.32	**0.851**	5.12	0.863
	Mean±StE	9.90±0.300	3.53±0.482	9.63±0.300	**3.38±0.216**
	Worst	13.93	13.56	15.16	**9.49**
	Fevals±StE	-	-	-	-

Table A.14: Results for single-point adaptive DRS using rule 1b on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP**	**PSO-DRS** **K**	**PSO-DRS** **PHI**
LJ_{12}	Success	0%	0%	0%	0%
	Best	10.78	**4.38E-07**	8.95	2.47
	Mean±StE	14.80±0.282	6.34±0.699	15.03±0.300	**5.63±0.244**
	Worst	19.36	18.17	19.43	**9.52**
	Fevals±StE	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%
	Best	13.73	2.88	11.57	**1.62**
	Mean±StE	20.55±0.366	10.35±0.849	20.44±0.431	**8.94±0.329**
	Worst	24.28	25.61	25.53	**14.05**
	Fevals±StE	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%
	Best	18.80	**2.03**	19.13	4.57
	Mean±StE	24.15±0.315	11.81±1.09	24.38±0.334	**9.46±0.362**
	Worst	29.80	28.82	29.44	**16.32**
	Fevals±StE	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%
	Best	23.14	**2.01**	23.73	6.56
	Mean±StE	28.32±0.322	15.43±1.33	28.17±0.308	**11.55±0.380**
	Worst	32.09	36.83	32.59	**17.55**
	Fevals±StE	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%
	Best	25.80	**1.00**	25.42	6.83
	Mean±StE	32.45±0.353	18.54±1.49	32.45±0.334	**12.89±0.407**
	Worst	37.62	41.19	36.73	**20.79**
	Fevals±StE	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%
	Best	30.52	**0.964**	27.85	7.89
	Mean±StE	37.13±0.408	20.54±1.83	37.32±0.430	**15.99±0.514**
	Worst	42.20	43.52	43.14	**25.33**
	Fevals±StE	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%
	Best	38.82	**2.69**	36.20	10.95
	Mean±StE	43.23±0.261	22.84±2.11	42.36±0.372	**18.48±0.560**
	Worst	47.29	50.63	46.76	**27.51**
	Fevals±StE	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%
	Best	44.26	**2.79**	44.60	16.28
	Mean±StE	49.28±0.336	28.13±2.07	49.34±0.318	**24.39±0.564**
	Worst	53.42	52.31	54.33	**34.76**
	Fevals±StE	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%
	Best	45.13	**4.90**	47.02	14.33
	Mean±StE	53.59±0.409	31.86±2.29	53.17±0.308	**26.57±0.712**
	Worst	58.92	55.68	57.96	**34.56**
	Fevals±StE	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%
	Best	80.37	**17.98**	76.05	36.21
	Mean±StE	84.92±0.285	64.52±2.85	85.53±0.351	**49.95±0.821**
	Worst	88.98	92.60	90.94	**62.33**
	Fevals±StE	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%
	Best	145.2	**77.65**	149.7	101.8
	Mean±StE	150.3±0.265	134.6±3.00	153.3±0.279	**115.7±0.944**
	Worst	153.8	156.6	158.0	**129.6**
	Fevals±StE	-	-	-	-

Table A.15: Results for single-point adaptive DRS using rule 1b on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_1	Success rate	**100%**	**100%**	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals±StE	76748±847	48398±456	64405±1242	**43309±365**	46943±780
f_2	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	3.61E-5	**2.65E-9**	3.01E-7	1.47E-5	3.64E-8
	Mean±StE	3.36E-3±1.21E-3	**4.54E-6±1.97E-6**	122.2±91.3	5.24E-2±2.77E-2	541.2±297.5
	Worst	5.14E-2	**9.33E-5**	4172	1.35	11568
	FEvals±StE	-	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**2.72E-5**	4.11E-3	8.47E-4	3.84E-4	9.02E-2
	Mean±StE	8.48±1.18	**5.28±0.81**	25.17±4.29	11.67±2.72	31.39±5.46
	Worst	33.1	**18.25**	94.51	78.2	136.4
	FEvals±StE	-	-	-	-	-

Table A.16: Results for multi-point adaptive DRS using rule 1b on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_4	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**829**	3079	3553	3435	3435
	Mean±StE	**1576±39**	3370±23	3553±3.6E-12	3548±3	3554±4
	Worst	**2013**	3731	3553	3553	3731
	FEvals±StE	-	-	-	-	-
f_5	Success rate	0%	0%	0%	0%	0%
	Best	2.98	32.83	18.9	19.9	40.79
	Mean±StE	9.19±0.64	64.82±2.79	64.86±2.96	71.96±3.69	78.37±2.78
	Worst	21.89	110.4	105.5	135.3	132.3
	FEvals±StE	-	-	-	-	-
f_6	Success rate	**100%**	26%	0%	0%	0%
	Best	**0.0**	**0.0**	5.37E-14	1.47E-14	1.57E-13
	Mean±StE	**0.0±0.0**	0.42±0.40	3.18±0.96	14.72±0.79	11.69±1.25
	Worst	**0.0**	19.75	19.4	18.36	19.59
	FEvals±StE	226790±9485	174832±11884	-	-	-
f_7	Success rate	**94%**	52%	48%	82%	46%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.4E-4±2.5E-4**	8.1E-3±2.1E-3	9.97E-3±2.0E-3	7.9E-3±4.8E-3	1.0E-2±2.5E-3
	Worst	**7.4E-3**	8.03E-2	7.54E-2	2.24E-1	7.79E-2
	FEvals±StE	70276±704	49234±1616	72782±1967	**46579±1200**	51874±2014
f_8	Success rate	**96%**	68%	92%	72%	56%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.2E-3±2.9E-3**	7.5E-2±2.7E-2	8.3E-3±4.0E-3	1.3E-1±4.9E-2	2.1E-1±1.2E-1
	Worst	**0.104**	1.25	**0.104**	1.98	5.64
	FEvals±StE	95725±630	**46873±1519**	63386±1888	62750±4508	63022±2335
f_9	Success rate	**98%**	84%	94%	72%	80%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**2.2E-4±2.2E-4**	2.4E-3±1.0E-3	4.4E-4±3.1E-4	4.3E-2±3.3E-2	2.2E-3±6.3E-4
	Worst	**1.1E-2**	4.39E-2	**1.1E-2**	1.60	**1.1E-2**
	FEvals±StE	91635±454	**48775±1581**	67996±1607	52283±1661	55694±1456

Table A.17: Results for multi-point adaptive DRS using rule 1b on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_{10}	Success rate	98%	**100%**	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.63E-2±1.63E-2	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	0.816	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals±StE	5622±122	9704±847	5695±241	**4459±101**	4725±140
f_{11}	Success rate	**98%**	82%	82%	78%	66%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**1.62±1.62**	14.48±4.42	14.58±4.45	32.94±17.05	27.54±5.48
	Worst	**81.0**	**81.0**	**81.0**	837.0	**81.0**
	FEvals±StE	6198±134	**4541±198**	7450±406	5877±139	5643±343
f_{12}	Success rate	**98%**	88%	92%	86%	84%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.149±0.149**	0.818±0.32	0.303±0.171	1.05±0.37	0.848±0.331
	Worst	7.47	7.47	**5.05**	7.47	7.47
	FEvals±StE	16839±1415	10505±381	14756±783	11378±449	**10375±417**
f_{13}	Success rate	**98%**	86%	96%	72%	80%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.134±0.134**	0.935±0.331	0.142±0.111	1.89±0.43	1.22±0.37
	Worst	6.68	6.68	**5.3**	7.65	7.65
	FEvals±StE	31861±2111	**9137±365**	14831±932	11714±339	11190±413
f_{14}	Success rate	**100%**	92%	96%	72%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	0.441±0.229	0.305±0.213	1.82±0.42	0.612±0.273
	Worst	**0.0**	6.7	7.67	6.7	7.67
	FEvals±StE	25119±1795	9315±314	12916±722	10816±345	11509±712

Table A.18: Results for multi-point adaptive DRS using rule 1b on simple multimodal problems

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP+K**	**PSO-DRS** **NP+PHI**	**PSO-DRS** **K+PHI**	**PSO-DRS** **NP+K+PHI**
LJ_2	Success	**100%**	8%	78%	90%	70%
	Best	0.0	0.0	0.0	0.0	0.0
	Mean±StE	**0.0±0.0**	8.0E-11±2.1E-11	3.7E-13±2.9E-13	6.1E-15±4.2E-15	8.0E-14±3.3E-14
	Worst	**0.0**	6.61E-10	1.45E-11	2.03E-13	1.06E-12
	Fevals±StE	**7601±753**	162372±47817	85563±13205	167330±16501	143939±15044
LJ_3	Success	22%	0%	**92%**	90%	86%
	Best	**0.0**	1.02E-14	**0.0**	**0.0**	**0.0**
	Mean±StE	3.2E-06±1.5E-06	1.0E-05±4.7E-06	**2.7E-15±1.3E-15**	9.0E-14±8.8E-14	9.9E-10±9.9E-10
	Worst	7.25E-05	2.09E-04	**4.13E-14**	4.41E-12	4.95E-08
	Fevals±StE	417959±36851	-	**55502±8825**	241670±23457	58081±8605
LJ_4	Success	20%	28%	**96%**	90%	94%
	Best	0.0	0.0	0.0	0.0	0.0
	Mean±StE	2.7E-05±1.5E-05	0.001±0.001	2.1E-12±2.0E-12	**4.9E-15±2.7E-15**	1.9E-09±1.9E-09
	Worst	7.17E-04	0.060	1.00E-10	**1.21E-13**	9.71E-08
	Fevals±StE	361428±27471	177189±24090	71232±8639	272866±18523	**65143±8733**
LJ_5	Success	6%	74%	98%	**100%**	94%
	Best	0.0	0.0	0.0	0.0	0.0
	Mean±StE	0.001±3.9E-04	0.005±0.005	0.002±0.002	**0.0±0.0**	6.1E-06±5.3E-06
	Worst	0.014	0.225	0.081	**0.0**	2.64E-04
	Fevals±StE	522557±57857	105536±10037	**50576±7053**	158424±9952	53420±8216
LJ_6	Success	0%	2%	**6%**	4%	**6%**
	Best	3.22E-04	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.335±0.023	0.480±0.051	0.389±0.020	**0.330±0.023**	0.405±0.032
	Worst	**0.530**	2.70	1.03	0.411	1.78
	Fevals±StE	-	**30070±0**	55801±17632	233656±78918	27374±4464
LJ_7	Success	0%	**24%**	20%	8%	**24%**
	Best	0.002	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.779±0.091	0.595±0.062	0.771±0.057	**0.294±0.056**	0.770±0.084
	Worst	2.99	1.44	1.52	**0.973**	3.39
	Fevals±StE	-	134595±28193	116504±27462	421957±56960	**92825±28013**
LJ_8	Success	0%	34%	48%	4%	**56%**
	Best	0.006	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.32±0.133	0.757±0.186	0.535±0.157	**0.250±0.060**	0.365±0.080
	Worst	4.06	5.59	7.30	**1.70**	2.36
	Fevals±StE	-	155767±19863	**99680±13218**	549749±5006	100390±11272
LJ_9	Success	0%	10%	**24%**	0%	**24%**
	Best	1.15	**0.0**	**0.0**	1.50E-06	**0.0**
	Mean±StE	3.89±0.203	2.30±0.358	**0.830±0.077**	1.05±0.098	1.25±0.215
	Worst	7.92	9.86	**2.08**	2.42	8.44
	Fevals±StE	-	221000±44862	103584±15486	-	**72078±18688**
LJ_{10}	Success	0%	0%	0%	0%	0%
	Best	0.143	**1.07E-07**	**1.07E-07**	2.63E-04	**1.07E-07**
	Mean±StE	6.33±0.319	3.50±0.374	**1.61±0.111**	1.68±0.123	**1.61±0.122**
	Worst	11.33	12.81	**3.18**	3.68	5.69
	Fevals±StE	-	-	-	-	-
LJ_{11}	Success	0%	4%	**14%**	0%	4%
	Best	5.32	**0.0**	**0.0**	0.573	**0.0**
	Mean±StE	9.90±0.300	4.55±0.542	2.37±0.369	3.13±0.234	**2.31±0.237**
	Worst	13.93	14.21	12.68	**7.70**	9.08
	Fevals±StE	-	201151±85547	168507±46917	-	**147480±32335**

Table A.19: Results for multi-point adaptive DRS using rule 1b on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_{12}	Success	0%	0%	0%	0%	0%
	Best	10.78	**1.62**	1.66	1.73	**1.62**
	Mean±StE	14.80±0.282	7.88±0.745	4.28±0.446	5.09±0.235	**4.17±0.449**
	Worst	19.36	19.98	13.17	**9.96**	17.79
	Fevals±StE	-	-	-	-	-
LJ_{13}	Success	0%	0%	**2%**	0%	0%
	Best	13.73	2.88	**0.0**	4.52	2.88
	Mean±StE	20.55±0.366	11.39±0.924	6.10±0.436	8.11±0.240	**5.81±0.349**
	Worst	24.28	27.84	19.15	**11.28**	15.11
	Fevals±StE	-	-	**244245±0**	-	-
LJ_{14}	Success	0%	0%	0%	0%	0%
	Best	18.80	2.79	**2.17E-07**	3.29	**2.17E-07**
	Mean±StE	24.15±0.315	15.31±1.00	**5.78±0.696**	9.18±0.408	7.04±0.646
	Worst	29.80	28.55	20.94	**16.56**	21.93
	Fevals±StE	-	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%	0%
	Best	23.14	1.90	1.90	5.14	**0.935**
	Mean±StE	28.32±0.322	17.80±1.48	7.13±0.695	11.67±0.458	**5.56±0.491**
	Worst	32.09	36.15	24.60	20.23	**18.75**
	Fevals±StE	-	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%	0%
	Best	25.80	1.91	5.38E-04	5.56	**2.20E-07**
	Mean±StE	32.45±0.353	24.36±1.58	**6.94±0.769**	13.21±0.536	7.98±0.971
	Worst	37.62	42.37	**19.33**	23.38	26.55
	Fevals±StE	-	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%	0%
	Best	30.52	3.01	**0.870**	6.52	**0.870**
	Mean±StE	37.13±0.408	23.18±1.70	8.68±1.04	14.99±0.490	**7.74±0.755**
	Worst	42.20	44.18	29.86	**21.86**	23.23
	Fevals±StE	-	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%	**2%**
	Best	38.82	4.20	0.246	9.27	**0.0**
	Mean±StE	43.23±0.261	28.89±1.78	**10.16±1.04**	17.97±0.553	10.92±1.29
	Worst	47.29	48.96	29.05	**27.92**	33.29
	Fevals±StE	-	-	-	-	**289177±0**
LJ_{19}	Success	0%	0%	0%	0%	0%
	Best	44.26	3.51	**2.36**	11.04	3.30
	Mean±StE	49.28±0.336	41.60±1.81	**11.55±1.05**	24.27±0.742	13.99±1.20
	Worst	53.42	58.29	**32.18**	35.14	35.36
	Fevals±StE	-	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%	0%
	Best	45.13	7.09	**2.35**	16.30	2.42
	Mean±StE	53.59±0.409	44.17±1.80	**13.18±1.24**	25.84±0.808	13.99±1.43
	Worst	58.92	60.05	**37.65**	42.51	40.61
	Fevals±StE	-	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%	0%
	Best	80.37	29.79	**3.02**	37.82	6.10
	Mean±StE	84.92±0.285	78.53±1.73	25.90±2.03	51.03±0.799	**25.55±2.00**
	Worst	88.98	91.81	67.43	63.05	**61.74**
	Fevals±StE	-	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%	0%
	Best	145.2	61.81	**16.32**	91.89	35.71
	Mean±StE	150.3±0.265	143.7±2.91	**66.54±4.11**	115.9±1.08	72.36±3.57
	Worst	153.8	159.3	**127.0**	131.6	131.3
	Fevals±StE	-	-	-	-	-

Table A.20: Results for multi-point adaptive DRS using rule 1b on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_1	Success rate	**100%**	0%	0%	**100%**
	Best	**0.0**	7.55E-4	15430	**0.0**
	Mean±StE	**0.0±0.0**	1.39E-2±1.95E-3	30431±885	**0.0±0.0**
	Worst	**0.0**	6.86E-2	47033	**0.0**
	FEvals±StE	76748±847	-	-	**58776±235**
f_2	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	3.61E-5	155.2	1.06E-3	**3.26E-5**
	Mean±StE	3.36E-3±1.21E-3	13551±1447	1.71E-1±4.3E-2	**6.69E-4±1.06E-4**
	Worst	5.14E-2	26596	1.89	**3.24E-3**
	FEvals±StE	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	**2.72E-5**	72.6	1.6E-2	4.48E-4
	Mean±StE	**8.48±1.18**	68756±21455	21.1±3.26	13.2±3.12
	Worst	**33.1**	709646	79.1	80.0
	FEvals±StE	-	-	-	-

Table A.21: Results for single-point adaptive DRS using rule 2 on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_4	Success rate	0%	0%	0%	0%
	Best	829	3524	1658	2013
	Mean±StE	1576±39	5491±72	2081±37	3217±58
	Worst	2013	6290	2611	3553
	FEvals±StE	-	-	-	-
f_5	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	**2.98**	56.8	1.99	9.95
	Mean±StE	**9.19±0.64**	182.2±7.76	9.09±0.57	41.4±2.74
	Worst	**21.89**	233	20.89	92.5
	FEvals±StE	-	-	-	-
f_6	Success rate	**100%**	0%	**100%**	16%
	Best	**0.0**	0.99	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	12.18±0.82	**0.0±0.0**	1.57±0.58
	Worst	**0.0**	18.09	**0.0**	14.92
	FEvals±StE	226790±9485	-	**149384±1066**	249562±15697
f_7	Success rate	**94%**	0%	**94%**	90%
	Best	**0.0**	0.42	**0.0**	**0.0**
	Mean±StE	**4.44E-4±2.51E-4**	1.37±0.11	**4.44E-4±2.51E-4**	7.89E-4±3.41E-4
	Worst	**7.4E-3**	3.68	**7.4E-3**	9.86E-3
	FEvals±StE	70276±704	-	**61856±891**	62852±446
f_8	Success rate	96%	0%	**100%**	82%
	Best	**0.0**	1.51E-2	**0.0**	**0.0**
	Mean±StE	4.15E-3±2.9E-3	7.4E5±1.3E5	**0.0±0.0**	5.6E-2±2.04E-2
	Worst	0.104	4.76E6	**0.0**	0.62
	FEvals±StE	95725±630	-	105991±969	**63904±536**
f_9	Success rate	98%	0%	**100%**	80%
	Best	**0.0**	1.13	**0.0**	**0.0**
	Mean±StE	2.2E-4±2.2E-4	3.36E6±6.2E5	**0.0±0.0**	2.66E-2±2.46E-2
	Worst	0.011	2.35E7	**0.0**	1.23
	FEvals±StE	91635±454	-	103086±675	**64358±288**

Table A.22: Results for single-point adaptive DRS using rule 2 on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_{10}	Success rate	**98%**	**98%**	40%	92%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**1.63E-2±1.63E-2**	**1.63E-2±1.63E-2**	2.35±0.8	6.53E-2±3.16E-2
	Worst	**0.816**	**0.816**	30.56	**0.816**
	FEvals±StE	5622±122	14990±1016	**5346±226**	6398±338
f_{11}	Success rate	98%	96%	**100%**	90%
	Best	**0.0**	**0.0**	**0.0**	
	Mean±StE	1.62±1.62	6.9E-15±6.6E-15	**0.0±0.0**	8.1±3.47
	Worst	81.0	3.3E-13	**0.0**	81.0
	FEvals±StE	6198±134	17884±4209	6314±222	5919±135
f_{12}	Success rate	**98%**	6%	68%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.149±0.149**	0.515±0.247	2.39±0.498	0.751±0.297
	Worst	**7.47**	**7.47**	7.52	**7.47**
	FEvals±StE	16839±1415	251606±27340	15485±1506	**13439±533**
f_{13}	Success rate	**98%**	0%	94%	90%
	Best	**0.0**	4.4E-10	**0.0**	**0.0**
	Mean±StE	0.134±0.134	**2.7E-7±8.88E-8**	0.401±0.227	0.64±0.275
	Worst	6.68	**3.67E-6**	6.68	6.68
	FEvals±StE	31861±2111	-	45727±9013	**15379±463**
f_{14}	Success rate	**100%**	0%	98%	78%
	Best	**0.0**	2.8E-10	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	4.55E-7±2.18E-7	0.133±0.133	1.47±0.396
	Worst	**0.0**	1.02E-5	6.67	7.67
	FEvals±StE	25119±1795	-	35566±3691	14470±421

Table A.23: Results for single-point adaptive DRS using rule 2 on simple multimodal problems

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP**	**PSO-DRS** **K**	**PSO-DRS** **PHI**
LJ_2	Success	**100%**	0%	4%	38%
	Best	**0.0**	1.47E-14	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	1.65E-10±4.16E-11	1.41E-10±5.18E-11	1.43E-11±1.08E-11
	Worst	**0.0**	1.45E-09	2.31E-09	5.23E-10
	Fevals±StE	**7601±753**	-	364636±179536	265298±35500
LJ_3	Success	22%	0%	0%	**38%**
	Best	**0.0**	4.34E-05	5.47E-06	**0.0**
	Mean±StE	3.20E-06±1.47E-06	0.003±3.33E-04	4.74E-04±8.52E-05	**3.05E-07±2.10E-07**
	Worst	7.25E-05	0.010	0.003	**9.28E-06**
	Fevals±StE	417959±36851	-	-	**269175±28221**
LJ_4	Success	20%	0%	0%	**50%**
	Best	**0.0**	0.016	2.84E-05	**0.0**
	Mean±StE	2.70E-05±1.46E-05	0.572±0.035	0.010±0.003	**4.04E-08±4.04E-08**
	Worst	7.17E-04	1.15	0.101	**2.02E-06**
	Fevals±StE	361428±27471	-	-	**376894±26791**
LJ_5	Success	6%	0%	0%	**94%**
	Best	**0.0**	1.19	6.56E-04	**0.0**
	Mean±StE	0.001±3.88E-04	2.39±0.058	0.205±0.031	**4.23E-06±2.94E-06**
	Worst	0.014	3.02	0.923	**1.11E-04**
	Fevals±StE	522557±57857	-	-	**229575±16543**
LJ_6	Success	0%	0%	0%	**10%**
	Best	3.22E-04	3.83	0.293	**0.0**
	Mean±StE	0.335±0.023	5.08±0.067	1.53±0.103	**0.291±0.026**
	Worst	0.530	5.71	3.20	**0.415**
	Fevals±StE	-	-	-	**321144±56840**
LJ_7	Success	0%	0%	0%	**4%**
	Best	0.002	6.50	0.732	**0.0**
	Mean±StE	0.779±0.091	8.23±0.082	3.65±0.195	**0.390±0.062**
	Worst	2.99	9.12	6.62	**1.08**
	Fevals±StE	-	-	-	**483838±55369**
LJ_8	Success	0%	0%	0%	0%
	Best	0.006	7.42	3.81	**2.83E-05**
	Mean±StE	1.32±0.133	10.65±0.113	6.58±0.151	**0.398±0.074**
	Worst	4.06	11.82	9.11	**2.30**
	Fevals±StE	-	-	-	-
LJ_9	Success	0%	0%	0%	0%
	Best	1.15	10.83	6.69	**9.63E-07**
	Mean±StE	3.89±0.203	14.37±0.137	9.94±0.196	**1.35±0.226**
	Worst	**7.92**	15.18	13.24	8.25
	Fevals±StE	-	-	-	-
LJ_{10}	Success	0%	0%	0%	0%
	Best	0.143	15.74	11.12	**0.007**
	Mean±StE	6.33±0.319	18.31±0.097	14.05±0.184	**2.74±0.287**
	Worst	11.33	19.51	16.63	**11.09**
	Fevals±StE	-	-	-	-
LJ_{11}	Success	0%	0%	0%	0%
	Best	5.32	15.91	14.98	**0.708**
	Mean±StE	9.90±0.300	21.40±0.166	18.31±0.177	**3.66±0.269**
	Worst	13.93	22.95	20.76	**10.96**
	Fevals±StE	-	-	-	-

Table A.24: Results for single-point adaptive DRS using rule 2 on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP**	**PSO-DRS K**	**PSO-DRS PHI**
LJ_{12}	Success	0%	0%	0%	0%
	Best	10.78	20.48	18.88	**2.02**
	Mean±StE	14.80±0.282	25.50±0.187	23.16±0.216	**7.08±0.606**
	Worst	**19.36**	27.32	26.91	19.56
	Fevals±StE	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%
	Best	13.73	25.13	26.47	**5.08**
	Mean±StE	20.55±0.366	30.60±0.249	29.67±0.186	**10.32±0.642**
	Worst	**24.28**	33.11	33.24	25.41
	Fevals±StE	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%
	Best	18.80	28.89	30.33	**4.69**
	Mean±StE	24.15±0.315	33.44±0.215	32.77±0.178	**12.06±0.860**
	Worst	**29.80**	35.84	35.39	30.01
	Fevals±StE	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%
	Best	23.14	33.24	34.23	**4.86**
	Mean±StE	28.32±0.322	37.06±0.192	37.67±0.176	**15.57±1.14**
	Worst	**32.09**	39.38	41.01	33.63
	Fevals±StE	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%
	Best	25.80	35.56	38.66	**6.67**
	Mean±StE	32.45±0.353	40.75±0.222	42.04±0.183	**17.36±1.08**
	Worst	**37.62**	43.25	44.74	39.76
	Fevals±StE	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%
	Best	30.52	40.90	42.02	**6.67**
	Mean±StE	37.13±0.408	44.90±0.199	46.24±0.215	**19.87±1.03**
	Worst	42.20	47.84	49.45	**41.05**
	Fevals±StE	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%
	Best	38.82	46.13	47.95	**11.16**
	Mean±StE	43.23±0.261	49.75±0.219	51.07±0.195	**26.13±1.47**
	Worst	**47.29**	53.62	53.68	47.41
	Fevals±StE	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%
	Best	44.26	48.58	53.66	**18.45**
	Mean±StE	49.28±0.336	55.94±0.292	57.54±0.208	**31.41±1.44**
	Worst	**53.42**	60.54	59.61	55.83
	Fevals±StE	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%
	Best	45.13	54.18	57.10	**14.93**
	Mean±StE	53.59±0.409	59.97±0.369	61.97±0.213	**34.21±1.55**
	Worst	58.92	65.05	64.48	**58.43**
	Fevals±StE	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%
	Best	80.37	89.15	89.55	**47.25**
	Mean±StE	84.92±0.285	92.16±0.198	92.72±0.193	**64.67±1.81**
	Worst	**88.98**	94.37	95.82	90.69
	Fevals±StE	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%
	Best	145.2	146.9	154.3	**119.3**
	Mean±StE	150.3±0.265	153.9±0.326	158.8±0.339	**137.0±1.78**
	Worst	**153.8**	157.4	165.0	156.1
	Fevals±StE	-	-	-	-

Table A.25: Results for single-point adaptive DRS using rule 2 on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_1	Success rate	**100%**	0%	0%	**100%**	96%
	Best	**0.0**	3.62E-13	1.83E-9	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	2.4E-12±2.5E-13	7.7E-7±6.9E-7	**0.0±0.0**	9.9E-15±9.4E-15
	Worst	**0.0**	8.94E-12	3.46E-5	**0.0**	4.7E-13
	FEvals±StE	76748±847	-	-	46435±420	174703±15602
f_2	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	3.61E-5	1.02E-2	1.18E-1	6.22E-6	**1.16E-6**
	Mean±StE	3.4E-3±1.2E-3	7.1E-2±7.1E-3	5.9E-1±3.5E-2	5.2E-2±3.0E-2	2.7E-3±2.0E-3
	Worst	**5.14E-2**	2.3E-1	1.27	1.47	9.5E-2
	FEvals±StE	-	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**2.72E-5**	9.35E-3	9.36E-2	1.16E-4	6.36E-3
	Mean±StE	**8.48±1.18**	9.47±1.66	13.78±2.81	8.76±2.11	11.35±1.57
	Worst	**33.1**	77.04	73.25	74.74	66.58
	FEvals±StE	-	-	-	-	-

Table A.26: Results for multi-point adaptive DRS using rule 2 on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_4	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**829**	3322	2865	2250	2606
	Mean±StE	**1576±39**	4585±129	3298±28	3238±45	3345±27
	Worst	**2013**	5967	3553	3553	3553
	FEvals±StE	-	-	-	-	-
f_5	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	2.98	1.001	4.78E-6	8.95	**6.04E-14**
	Mean±StE	9.19±0.64	7.18±0.705	**6.27±0.799**	48.77±3.84	20.68±2.42
	Worst	**21.89**	24.88	22.88	139.29	73.63
	FEvals±StE	-	-	-	-	-
f_6	Success rate	**100%**	0%	0%	58%	0%
	Best	**0.0**	4.59E-3	8.37E-5	**0.0**	4.68E-9
	Mean±StE	**0.0±0.0**	7.8E-3±2.6E-4	1.4E-4±5.2E-6	8.2E-1±2.8E-1	1.3E-6±8.0E-7
	Worst	**0.0**	1.24E-2	2.54E-4	10.91	3.39E-5
	FEvals±StE	226790±9485	-	-	151427±13226	-
f_7	Success rate	**94%**	0%	0%	66%	20%
	Best	**0.0**	2.05E-6	2.38E-7	**0.0**	**0.0**
	Mean±StE	4.4E-4±2.5E-4	2.2E-3±6.2E-4	**4.0E-4±2.2E-4**	4.0E-3±9.5E-4	3.6E-3±1.1E-3
	Worst	**7.4E-3**	1.73E-2	**7.4E-3**	2.7E-2	3.2E-2
	FEvals±StE	70276±704	-	-	**48151±523**	317417±18013
f_8	Success rate	96%	0%	0%	68%	**98%**
	Best	**0.0**	6.52E-7	1.62E-10	**0.0**	**0.0**
	Mean±StE	4.2E-3±2.9E-3	3.4E-6±2.9E-7	8.9E-10±8.3E-11	1.1E-1±3.6E-2	**5.1E-14±5.1E-14**
	Worst	0.104	1.06E-5	3.19E-9	1.36	**2.5E-12**
	FEvals±StE	95725±630	-	-	**55908±903**	129594±13401
f_9	Success rate	**98%**	0%	0%	74%	94%
	Best	**0.0**	1.65E-5	1.31E-9	**0.0**	**0.0**
	Mean±StE	2.2E-4±2.2E-4	5.7E-5±4.0E-6	9.3E-5±7.7E-5	9.2E-2±5.0E-2	**1.0E-10±9.5E-11**
	Worst	1.1E-2	1.24E-4	3.73E-3	1.60	**4.71E-9**
	FEvals±StE	91635±454	-	-	**55753±844**	184279±12718

Table A.27: Results for multi-point adaptive DRS using rule 2 on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_{10}	Success rate	98%	92%	**100%**	82%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.63E-2±1.63E-2	6.53E-2±3.16E-2	**0.0±0.0**	1.47E-1±4.48E-2	**0.0±0.0**
	Worst	0.816	0.816	**0.0**	0.816	**0.0**
	FEvals±StE	**5622±122**	10101±811	14317±5608	6802±307	21097±4609
f_{11}	Success rate	98%	**100%**	**100%**	92%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	**0.0±0.0**	**0.0±0.0**	4.86±2.75	**0.0±0.0**
	Worst	81.0	**0.0**	**0.0**	81.0	**0.0**
	FEvals±StE	6198±134	8051±354	6200±258	**5211±130**	14424±7875
f_{12}	Success rate	98%	68%	96%	72%	76%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.149±0.149	1.38±0.38	0.299±0.209	1.996±0.462	1.59±0.412
	Worst	7.47	7.47	7.47	7.52	7.47
	FEvals±StE	16839±1415	24365±1889	17657±1413	10341±487	22225±7581
f_{13}	Success rate	98%	18%	**100%**	84%	98%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.134±0.134	0.128±0.128	**0.0±0.0**	1.01±0.33	0.134±0.134
	Worst	6.68	6.40	**0.0**	6.68	6.74
	FEvals±StE	31861±2111	120340±17304	16543±914	10755±451	15454±823
f_{14}	Success rate	**100%**	14%	**100%**	92%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	9.38E-12±1.97E-12	**0.0±0.0**	5.36E-1±2.6E-1	**0.0±0.0**
	Worst	**0.0**	7.77E-11	**0.0**	6.70	**0.0**
	FEvals±StE	25119±1795	109764±12486	16902±944	**12602±676**	14762±996

Table A.28: Results for multi-point adaptive DRS using rule 2 on simple multimodal problems

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_2	Success	**100%**	2%	96%	20%	**100%**
	Best	0.0	0.0	0.0	0.0	0.0
	Mean±StE	**0.0±0.0**	1.1E-10±2.5E-11	1.5E-14±1.5E-14	1.5E-12±6.0E-13	**0.0±0.0**
	Worst	**0.0**	9.08E-10	7.27E-13	2.06E-11	**0.0**
	Fevals±StE	**7601±753**	282074±0	191298±16706	292648±57023	8807±932
LJ_3	Success	22%	0%	0%	2%	**98%**
	Best	**0.0**	1.02E-09	2.93E-14	**0.0**	**0.0**
	Mean±StE	3.2E-06±1.5E-06	2.6E-06±8.1E-07	**4.5E-11±1.5E-11**	2.0E-08±1.8E-08	1.4E-07±1.4E-07
	Worst	7.25E-05	3.74E-05	**7.14E-10**	9.02E-07	6.98E-06
	Fevals±StE	417959±36851	-	-	235624±0	**64379±11222**
LJ_4	Success	20%	0%	0%	30%	**98%**
	Best	**0.0**	1.05E-07	5.10E-11	**0.0**	**0.0**
	Mean±StE	2.7E-05±1.5E-05	8.2E-06±2.1E-06	1.1E-09±2.5E-10	**8.0E-10±6.5E-10**	5.9E-06±5.9E-06
	Worst	7.17E-04	9.78E-05	**1.09E-08**	3.25E-08	2.97E-04
	Fevals±StE	361428±27471	-	-	401547±30970	**86547±12957**
LJ_5	Success	6%	2%	**98%**	94%	96%
	Best	0.0	0.0	0.0	0.0	0.0
	Mean±StE	0.001±3.9E-04	2.4E-05±3.7E-06	0.003±0.003	**4.4E-08±2.6E-08**	0.015±0.013
	Worst	0.014	1.31E-04	0.171	**9.68E-07**	0.632
	Fevals±StE	522557±57857	95067±0	95101±12227	272572±16629	**83811±15056**
LJ_6	Success	0%	0%	6%	6%	**10%**
	Best	3.22E-04	3.95E-05	**0.0**	**0.0**	**0.0**
	Mean±StE	0.335±0.023	0.817±0.169	0.411±0.031	**0.326±0.023**	0.406±0.034
	Worst	0.530	4.81	1.78	**0.410**	1.61
	Fevals±StE	-	-	99821±26007	452427±37538	**69145±11436**
LJ_7	Success	0%	0%	26%	0%	**42%**
	Best	0.002	2.23E-05	**0.0**	1.03E-07	**0.0**
	Mean±StE	0.779±0.091	3.22±0.231	0.688±0.074	**0.254±0.054**	0.525±0.067
	Worst	2.99	5.80	2.79	**0.973**	1.52
	Fevals±StE	-	-	165260±19538	-	**115489±27942**
LJ_8	Success	0%	0%	42%	0%	**50%**
	Best	0.006	0.005	**0.0**	5.38E-07	**0.0**
	Mean±StE	1.32±0.133	6.13±0.337	0.478±0.158	**0.167±0.047**	0.456±0.128
	Worst	4.06	10.13	7.57	**1.72**	5.87
	Fevals±StE	-	-	214654±24648	-	**107249±10164**
LJ_9	Success	0%	0%	**22%**	0%	12%
	Best	1.15	4.62	**0.0**	1.95E-04	**0.0**
	Mean±StE	3.89±0.203	10.10±0.351	**1.03±0.131**	1.17±0.096	1.04±0.075
	Worst	7.92	14.94	5.66	2.75	**2.03**
	Fevals±StE	-	-	**159382±19727**	-	166648±79646
LJ_{10}	Success	0%	0%	0%	0%	0%
	Best	0.143	11.43	1.36E-07	0.030	**1.07E-07**
	Mean±StE	6.33±0.319	15.56±0.248	**1.46±0.142**	1.80±0.134	1.93±0.425
	Worst	11.33	18.72	6.34	**4.24**	14.98
	Fevals±StE	-	-	-	-	-
LJ_{11}	Success	0%	0%	2%	0%	**6%**
	Best	5.32	13.34	**0.0**	1.18	**0.0**
	Mean±StE	9.90±0.300	19.68±0.296	**2.25±0.275**	3.31±0.191	2.32±0.340
	Worst	13.93	22.59	11.27	**6.25**	15.83
	Fevals±StE	-	-	510731±0	-	**299412±43412**

Table A.29: Results for multi-point adaptive DRS using rule 2 on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_{12}	Success	0%	0%	0%	0%	0%
	Best	10.78	17.79	1.66	1.06	**4.38E-07**
	Mean±StE	14.80±0.282	24.28±0.295	3.85±0.392	5.69±0.290	**3.74±0.642**
	Worst	19.36	27.56	13.30	**11.46**	26.00
	Fevals±StE	-	-	-	-	-
LJ_{13}	Success	0%	0%	2%	0%	**4%**
	Best	13.73	26.12	**0.0**	4.73	**0.0**
	Mean±StE	20.55±0.366	30.76±0.231	**5.94±0.488**	10.38±0.392	6.03±0.603
	Worst	24.28	32.93	23.43	**17.59**	27.34
	Fevals±StE	-	-	481136±0	-	**326334±62880**
LJ_{14}	Success	0%	0%	0%	0%	0%
	Best	18.80	30.17	8.44E-07	4.37	**2.17E-07**
	Mean±StE	24.15±0.315	33.61±0.208	**4.85±0.417**	11.61±0.376	6.60±0.610
	Worst	29.80	36.34	**14.92**	15.74	28.14
	Fevals±StE	-	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%	**2%**
	Best	23.14	33.91	1.14E-06	6.56	**0.0**
	Mean±StE	28.32±0.322	37.94±0.213	**4.99±0.405**	14.51±0.481	6.10±0.491
	Worst	32.09	40.50	**13.07**	21.96	17.10
	Fevals±StE	-	-	-	-	**179019±0**
LJ_{16}	Success	0%	0%	0%	0%	0%
	Best	25.80	35.56	2.12	9.90	**1.70E-06**
	Mean±StE	32.45±0.353	41.48±0.276	7.59±0.721	17.83±0.542	**7.41±0.542**
	Worst	37.62	44.60	**19.42**	25.70	20.56
	Fevals±StE	-	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%	0%
	Best	30.52	41.93	0.245	14.79	**0.011**
	Mean±StE	37.13±0.408	45.80±0.229	**6.89±0.668**	22.70±0.612	8.12±0.972
	Worst	42.20	48.61	**23.29**	33.19	43.25
	Fevals±StE	-	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%	0%
	Best	38.82	46.25	1.57	15.11	**0.246**
	Mean±StE	43.23±0.261	50.33±0.237	**9.75±1.11**	26.72±0.728	12.96±1.36
	Worst	47.29	53.62	50.17	**34.57**	50.58
	Fevals±StE	-	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%	0%
	Best	44.26	48.58	**2.36**	23.12	3.55
	Mean±StE	49.28±0.336	57.02±0.339	**12.21±1.45**	33.84±0.690	14.08±0.941
	Worst	53.42	61.18	52.71	44.31	**31.78**
	Fevals±StE	-	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%	0%
	Best	45.13	54.18	**2.82**	25.96	3.23
	Mean±StE	53.59±0.409	61.07±0.393	**10.04±0.600**	37.71±0.638	16.78±1.32
	Worst	58.92	67.73	**19.57**	47.22	61.49
	Fevals±StE	-	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%	0%
	Best	80.37	90.53	**4.12**	58.49	13.04
	Mean±StE	84.92±0.285	94.93±0.256	**17.90±1.38**	71.62±0.699	31.63±1.89
	Worst	88.98	97.77	**39.41**	82.80	91.88
	Fevals±StE	-	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%	0%
	Best	145.2	153.4	**13.89**	132.0	39.10
	Mean±StE	150.3±0.265	158.9±0.371	**53.63±3.48**	140.5±0.531	75.30±2.76
	Worst	153.8	163.0	**112.7**	148.7	152.9
	Fevals±StE	-	-	-	-	-

Table A.30: Results for multi-point adaptive DRS using rule 2 on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_1	Success rate	**100%**	**100%**	0%	**100%**
	Best	**0.0**	**0.0**	8197	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	18182±705	**0.0±0.0**
	Worst	**0.0**	**0.0**	35919	**0.0**
	FEvals±StE	76748±847	146136±15472	-	**57332±174**
f_2	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	3.61E-5	**3.08E-6**	9.72E-2	6.05E-5
	Mean±StE	**3.36E-3±1.21E-3**	7.81±2.16	29.97±10.8	3.65E-3±1.4E-3
	Worst	**5.14E-2**	577.6	440.3	6.57E-2
	FEvals±StE	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	2.72E-5	**2.85E-6**	1.71E-4	1.84E-3
	Mean±StE	8.48±1.18	9.32±2.48	24.7±4.63	**8.1±2.05**
	Worst	**33.1**	79.3	127.2	74.3
	FEvals±StE	-		-	

Table A.31: Results for single-point adaptive DRS using rule 3 on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_4	Success rate	**0%**	**0%**	**0%**	**0%**
	Best	829	**473**	1658	1066
	Mean±StE	**1576±39**	2085±115	2436±67	2347±92
	Worst	**2013**	3316	3834	3553
	FEvals±StE	-	-	-	-
f_5	Success rate	0%	**6%**	0%	0%
	Best	2.98	**0.0**	3.98	1.99
	Mean±StE	**9.19±0.64**	10.97±1.42	15.86±1.72	30.53±2.96
	Worst	**21.89**	41.79	80.45	76.61
	FEvals±StE	-	**437421±15196**	-	-
f_6	Success rate	**100%**	70%	**100%**	18%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	1.02E-11±1.02E-11	**0.0±0.0**	1.86E-2±1.86E-2
	Worst	**0.0**	5.09E-10	**0.0**	0.931
	FEvals±StE	226790±9485	285195±12686	**149647±2268**	314949±20182
f_7	Success rate	**94%**	80%	90%	86%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.44E-4±2.51E-4**	1.33E-3±4.06E-4	7.89E-4±3.41E-4	1.43E-3±5.76E-4
	Worst	**7.4E-3**	**7.4E-3**	9.86E-3	2.24E-2
	FEvals±StE	70276±704	142750±14177	**58751±413**	62029±470
f_8	Success rate	96%	**100%**	**100%**	86%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.15E-3±2.9E-3**	**0.0±0.0**	**0.0±0.0**	3.11E-2±1.73E-2
	Worst	0.104	**0.0**	**0.0**	0.83
	FEvals±StE	95725±630	170665±18994	104691±1192	**63043±627**
f_9	Success rate	98%	**100%**	96%	80%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	2.2E-4±2.2E-4	**0.0±0.0**	4.39E-4±3.08E-4	2.2E-3±6.28E-4
	Worst	0.011	**0.0**	0.011	0.011
	FEvals±StE	91635±454	178617±18196	108072±1066	**63832±371**

Table A.32: Results for single-point adaptive DRS using rule 3 on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP***	**PSO-DRS** ***K***	**PSO-DRS** ϕ
f_{10}	Success rate	98%	**100%**	64%	92%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.0163±0.0163	**0.0±0.0**	0.294±0.056	0.0653±0.0316
	Worst	0.816	**0.0**	0.816	0.816
	FEvals±StE	**5622±122**	9235±335	10099±1173	6049±161
f_{11}	Success rate	98%	92%	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	6.48±3.14	**0.0±0.0**	**0.0±0.0**
	Worst	81.0	81.0	**0.0**	**0.0**
	FEvals±StE	6198±134	8308±470	6478±234	**5997±168**
f_{12}	Success rate	98%	**100%**	78%	82%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.149±0.149	**0.0±0.0**	1.55±0.42	1.3±0.398
	Worst	7.47	**0.0**	7.47	7.47
	FEvals±StE	16839±1415	31124±4195	18593±1563	**14159±816**
f_{13}	Success rate	**98%**	94%	94%	90%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.134±0.134**	0.345±0.196	0.401±0.227	0.668±0.286
	Worst	**6.68**	**6.68**	**6.68**	**6.68**
	FEvals±StE	31861±2111	23605±1765	50225±11207	**14609±565**
f_{14}	Success rate	**100%**	98%	98%	82%
	Best	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	0.134±0.134	0.134±0.134	1.15±0.353
	Worst	**0.0**	6.7	6.7	6.7
	FEvals±StE	25119±1795	26557±3766	39623±7323	**16131±626**

Table A.33: Results for single-point adaptive DRS using rule 3 on simple multimodal problems

	Adapting	**PSO-DRS** ***Fixed***	**PSO-DRS** **NP**	**PSO-DRS** **K**	**PSO-DRS** **PHI**
LJ_2	Success	**100%**	18%	2%	26%
	Best	0.0	0.0	0.0	0.0
	Mean±StE	**0.0±0.0**	5.02E-12±1.64E-12	1.44E-10±3.31E-11	5.61E-12±2.34E-12
	Worst	**0.0**	5.92E-11	1.35E-09	8.29E-11
	Fevals±StE	**7601±753**	275870±55688	229440±0	227716±45308
LJ_3	Success	**22%**	2%	0%	10%
	Best	**0.0**	**0.0**	1.21E-06	**0.0**
	Mean±StE	3.20E-06±1.47E-06	**1.24E-07±6.03E-08**	8.43E-04±1.87E-04	1.04E-05±6.01E-06
	Worst	7.25E-05	**2.68E-06**	0.007	2.96E-04
	Fevals±StE	417959±36851	**136418±0**	-	396184±40379
LJ_4	Success	20%	2%	0%	**22%**
	Best	**0.0**	**0.0**	8.44E-06	**0.0**
	Mean±StE	2.70E-05±1.46E-05	**2.61E-06±2.32E-06**	0.023±0.007	7.69E-05±6.86E-05
	Worst	7.17E-04	**1.16E-04**	0.298	0.003
	Fevals±StE	**361428±27471**	453873±0	-	409173±27727
LJ_5	Success	6%	**74%**	0%	**72%**
	Best	**0.0**	**0.0**	3.12E-06	**0.0**
	Mean±StE	0.001±3.88E-04	**4.42E-05±3.12E-05**	0.447±0.086	2.62E-04±1.59E-04
	Worst	0.014	**0.001**	2.59	0.007
	Fevals±StE	522557±57857	320535±21760	-	**279537±22177**
LJ_6	Success	0%	**4%**	0%	2%
	Best	3.22E-04	**0.0**	0.140	**0.0**
	Mean±StE	0.335±0.023	**0.278±0.026**	1.77±0.137	0.365±0.024
	Worst	**0.530**	0.545	4.20	0.922
	Fevals±StE	-	409420±84214	-	**257878±0**
LJ_7	Success	0%	0%	0%	**2%**
	Best	0.002	1.41E-06	1.20	**0.0**
	Mean±StE	0.779±0.091	**0.492±0.066**	3.95±0.184	0.680±0.092
	Worst	2.99	**1.96**	6.28	3.14
	Fevals±StE	-	-	-	**593150±0**
LJ_8	Success	0%	**10%**	0%	0%
	Best	0.006	**0.0**	4.54	2.18E-05
	Mean±StE	1.32±0.133	0.600±0.125	6.80±0.158	**0.588±0.122**
	Worst	4.06	4.90	9.08	**3.61**
	Fevals±StE	-	**426536±14487**	-	-
LJ_9	Success	0%	**2%**	0%	0%
	Best	1.15	**0.0**	2.40	6.85E-06
	Mean±StE	3.89±0.203	**1.44±0.215**	10.07±0.280	2.27±0.282
	Worst	7.92	**6.47**	12.94	7.15
	Fevals±StE	-	**444251±0**	-	-
LJ_{10}	Success	0%	0%	0%	0%
	Best	0.143	0.328	10.67	**0.006**
	Mean±StE	6.33±0.319	**3.34±0.411**	14.02±0.199	4.27±0.491
	Worst	**11.33**	11.62	16.53	11.83
	Fevals±StE	-	-	-	-
LJ_{11}	Success	0%	**2%**	0%	0%
	Best	5.32	**0.0**	15.01	0.152
	Mean±StE	9.90±0.300	**4.20±0.384**	18.51±0.207	5.80±0.615
	Worst	13.93	**10.81**	22.29	16.68
	Fevals±StE	-	**521437±0**	-	-

Table A.34: Results for single-point adaptive DRS using rule 3 on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP**	**PSO-DRS K**	**PSO-DRS PHI**
LJ_{12}	Success	0%	0%	0%	0%
	Best	10.78	**2.17**	18.40	2.81
	Mean±StE	14.80±0.282	**7.91±0.622**	23.22±0.222	9.42±0.787
	Worst	19.36	**18.04**	26.16	21.33
	Fevals±StE	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%
	Best	13.73	**4.01**	25.83	4.98
	Mean±StE	20.55±0.366	**11.80±0.537**	29.64±0.218	13.15±0.955
	Worst	24.28	**20.78**	31.98	27.68
	Fevals±StE	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%
	Best	18.80	**4.01**	28.53	6.72
	Mean±StE	24.15±0.315	**15.00±0.839**	32.94±0.217	16.98±1.15
	Worst	29.80	**27.21**	35.78	32.16
	Fevals±StE	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%
	Best	23.14	**4.19**	32.27	8.08
	Mean±StE	28.32±0.322	**16.95±0.818**	37.24±0.208	19.98±1.16
	Worst	32.09	**28.37**	39.98	34.73
	Fevals±StE	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%
	Best	25.80	**4.49**	36.63	7.88
	Mean±StE	32.45±0.353	**21.82±1.03**	41.88±0.219	22.04±1.35
	Worst	37.62	**33.94**	44.87	38.42
	Fevals±StE	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%
	Best	30.52	**7.58**	41.83	12.87
	Mean±StE	37.13±0.408	**24.43±1.06**	46.04±0.228	29.03±1.60
	Worst	42.20	**38.78**	49.41	45.09
	Fevals±StE	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%
	Best	38.82	**13.67**	46.54	15.72
	Mean±StE	43.23±0.261	**27.00±0.932**	51.00±0.249	32.24±1.60
	Worst	47.29	**43.92**	54.53	49.61
	Fevals±StE	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%
	Best	44.26	**17.84**	52.91	18.49
	Mean±StE	49.28±0.336	**33.83±1.15**	57.30±0.228	37.44±1.52
	Worst	53.42	**49.17**	60.59	55.48
	Fevals±StE	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%
	Best	45.13	**19.45**	56.44	21.54
	Mean±StE	53.59±0.409	**36.66±1.16**	61.87±0.228	42.98±1.64
	Worst	58.92	**52.36**	64.51	59.10
	Fevals±StE	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%
	Best	80.37	**35.32**	88.66	51.03
	Mean±StE	84.92±0.285	**68.18±1.46**	93.05±0.285	74.46±1.70
	Worst	88.98	**86.80**	96.29	91.56
	Fevals±StE	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%
	Best	145.2	**112.0**	154.9	119.4
	Mean±StE	150.3±0.265	**137.4±1.58**	158.7±0.297	141.1±1.62
	Worst	153.8	**153.1**	162.7	156.0
	Fevals±StE	-	-	-	-

Table A.35: Results for single-point adaptive DRS using rule 3 on Lennard-Jones problems 12–20, 26, and 38

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_1	Success rate	**100%**	**100%**	**100%**	**100%**	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
	Worst	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	FEvals±StE	76748±847	165019±12498	102079±9413	**49377±1506**	71623±4449
f_2	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	3.61E-5	3.79E-8	5.64E-9	1.44E-4	**2.58E-9**
	Mean±StE	**3.36E-3±1.21E-3**	4.6±2.41	4.02±1.67	24.2±14.8	2.4±1.36
	Worst	**5.14E-2**	110.1	74.0	625.2	59.4
	FEvals±StE	-	-	-	-	-
f_3	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**2.72E-5**	3.68E-5	1.30E-3	1.93E-2	4.27E-2
	Mean±StE	**8.48±1.18**	10.7±2.54	15.23±3.29	12.69±3.18	23.11±4.29
	Worst	**33.1**	78.1	80.3	122.5	101.3
	FEvals±StE	-	-	-	-	-

Table A.36: Results for multi-point adaptive DRS using rule 3 on unimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_4	Success rate	**0%**	**0%**	**0%**	**0%**	**0%**
	Best	**829**	1304	948	1303	1421
	Mean±StE	**1576±39**	1827±37	1952±72	2399±74	2451±77
	Worst	**2013**	2606	3079	3435	4905
	FEvals±StE	-	-	-	-	
f_5	Success rate	0%	0%	**32%**	0%	4%
	Best	2.98	2.9E-13	**0.0**	7.96	0.0
	Mean±StE	**9.19±0.64**	10.15±1.69	11.72±2.22	42.76±4.23	33.63±4.45
	Worst	**21.89**	54.72	56.7	155.8	120.3
	FEvals±StE	-	-	**407830±12921**	-	507970±7934
f_6	Success rate	**100%**	54%	12%	54%	54%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	2.3E-2±2.3E-2	4.2E-2±2.9E-2	2.2E-1±7.4E-2	5.9E-2±4.2E-2
	Worst	**0.0**	1.16	1.16	1.90	1.78
	FEvals±StE	226790±9485	243202±15192	281326±17253	**128688±10360**	290879±17450
f_7	Success rate	**94%**	92%	78%	72%	64%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.4E-4±2.5E-4**	6.9E-4±3.4E-4	2.4E-3±7.1E-4	4.6E-3±1.6E-3	4.1E-3±1.5E-3
	Worst	**7.4E-3**	9.86E-3	2.22E-2	7.06E-2	4.89E-2
	FEvals±StE	70276±704	108654±10090	130082±11133	**55843±3232**	76485±3606
f_8	Success rate	96%	**100%**	**100%**	80%	92%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**4.2E-3±2.9E-3**	**0.0±0.0**	**0.0±0.0**	0.104±0.047	0.012±0.006
	Worst	0.104	**0.0**	**0.0**	1.88	0.21
	FEvals±StE	95725±630	133990±12357	111774±9765	**83489±5933**	103202±6399
f_9	Success rate	**98%**	94%	86%	84%	88%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**2.2E-4±2.2E-4**	6.6E-4±3.7E-4	1.7E-3±6.4E-4	1.8E-3±5.8E-4	1.3E-3±5.1E-4
	Worst	**0.011**	**0.011**	0.021	**0.011**	**0.011**
	FEvals±StE	91635±454	136680±13718	138086±12368	**69216±3522**	96477±7311

Table A.37: Results for multi-point adaptive DRS using rule 3 on complex multimodal problems

	Adapting	**PSO-DRS**	**PSO-DRS** ***NP+K***	**PSO-DRS** ***NP+ϕ***	**PSO-DRS** ***K+ϕ***	**PSO-DRS** ***NP+K+ϕ***
f_{10}	Success rate	**98%**	94%	**98%**	84%	78%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0163±0.0163**	0.049±0.028	**0.0163±0.0163**	0.131±0.043	0.18±0.048
	Worst	0.816	0.816	0.816	0.816	0.816
	FEvals±StE	**5622±122**	5873±153	6743±279	13050±2734	9675±1938
f_{11}	Success rate	98%	**100%**	**100%**	98%	**100%**
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	1.62±1.62	**0.0±0.0**	**0.0±0.0**	1.62±1.62	**0.0±0.0**
	Worst	81.0	**0.0**	**0.0**	81.0	**0.0**
	FEvals±StE	6198±134	6685±204	6546±211	**6074±247**	7019±400
f_{12}	Success rate	**98%**	96%	88%	88%	78%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.134±0.134**	0.299±0.209	0.751±0.297	0.8±0.314	1.63±0.439
	Worst	**7.47**	**7.47**	**7.47**	**7.47**	7.52
	FEvals±StE	**16839±1415**	30241±3311	19012±1539	17745±2018	19559±4805
f_{13}	Success rate	98%	**100%**	96%	88%	94%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	0.134±0.134	**0.0±0.0**	0.267±0.187	0.739±0.292	0.401±0.227
	Worst	6.68	**0.0**	6.68	6.68	6.68
	FEvals±StE	31861±2111	38291±5443	**21322±1063**	25582±3465	35413±4748
f_{14}	Success rate	**100%**	**100%**	**100%**	88%	94%
	Best	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	0.804±0.311	0.392±0.222
	Worst	**0.0**	**0.0**	**0.0**	6.7	6.7
	FEvals±StE	25119±1795	34304±2816	24129±2343	**18384±1271**	23639±2081

Table A.38: Results for multi-point adaptive DRS using rule 3 on simple multimodal problems

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_2	Success	**100%**	0%	**100%**	2%	4%
	Best	**0.0**	2.98E-14	**0.0**	**0.0**	**0.0**
	Mean±StE	**0.0±0.0**	2.3E-10±7.7E-11	**0.0±0.0**	9.8E-11±3.6E-11	1.2E-10±4.0E-11
	Worst	**0.0**	3.04E-09	**0.0**	1.63E-09	1.42E-09
	Fevals±StE	7601±753	-	7196±745	197951±0	145102±98337
LJ_3	Success	22%	0%	**80%**	0%	2%
	Best	**0.0**	3.82E-09	**0.0**	6.02E-11	**0.0**
	Mean±StE	3.2E-06±1.5E-06	0.001±2.5E-04	**1.5E-06±8.5E-07**	0.001±3.1E-04	9.4E-04±4.1E-04
	Worst	7.25E-05	0.010	**3.78E-05**	0.011	0.020
	Fevals±StE	417959±36851	-	**101060±18613**	-	205026±0
LJ_4	Success	20%	0%	**92%**	0%	10%
	Best	**0.0**	1.45E-09	**0.0**	1.95E-14	**0.0**
	Mean±StE	2.7E-05±1.5E-05	0.038±0.015	**1.8E-05±1.8E-05**	0.079±0.019	0.052±0.019
	Worst	**7.17E-04**	0.514	8.91E-04	0.622	0.651
	Fevals±StE	361428±27471	-	**119381±12254**	-	312891±49000
LJ_5	Success	6%	0%	**84%**	0%	20%
	Best	**0.0**	1.06E-06	**0.0**	7.78E-07	**0.0**
	Mean±StE	0.001±3.9E-04	0.455±0.090	**8.4E-06±5.1E-06**	0.334±0.072	0.444±0.094
	Worst	0.014	2.43	**2.25E-04**	1.74	2.23
	Fevals±StE	522557±57857	-	**187843±13722**	-	329891±34988
LJ_6	Success	0%	0%	**2%**	0%	0%
	Best	3.22E-04	0.240	**0.0**	0.014	1.94E-06
	Mean±StE	**0.335±0.023**	1.63±0.157	0.371±0.020	1.41±0.163	1.21±0.154
	Worst	**0.530**	4.34	0.659	4.53	3.93
	Fevals±StE	-	-	**168239±0**	-	-
LJ_7	Success	0%	0%	**8%**	0%	2%
	Best	0.002	1.17E-04	**0.0**	6.12E-04	**0.0**
	Mean±StE	0.779±0.091	3.20±0.281	**0.476±0.079**	3.40±0.281	2.86±0.343
	Worst	2.99	6.45	**1.97**	7.13	7.93
	Fevals±StE	-	-	**346136±72266**	-	429834±0
LJ_8	Success	0%	0%	**12%**	0%	2%
	Best	0.006	1.02	**0.0**	0.297	**0.0**
	Mean±StE	1.32±0.133	5.77±0.338	**0.522±0.137**	4.92±0.415	4.26±0.452
	Worst	**4.06**	9.66	5.23	10.12	9.49
	Fevals±StE	-	-	**333654±75485**	-	385249±0
LJ_9	Success	0%	0%	**4%**	0%	0%
	Best	1.15	1.92	**0.0**	0.761	6.23E-06
	Mean±StE	3.89±0.203	9.26±0.453	**1.57±0.287**	7.14±0.548	7.19±0.660
	Worst	**7.92**	14.17	8.73	13.55	13.99
	Fevals±StE	-	-	**316426±49634**	-	-
LJ_{10}	Success	0%	0%	0%	0%	0%
	Best	0.143	2.50	**1**.07E-07	2.56	8.13E-06
	Mean±StE	6.33±0.319	12.69±0.495	**2.54±0.354**	11.92±0.615	10.73±0.700
	Worst	**11.33**	17.12	11.76	17.63	17.50
	Fevals±StE	-	-	-	-	-
LJ_{11}	Success	0%	0%	0%	0%	0%
	Best	5.32	6.34	**0.002**	1.66	1.49
	Mean±StE	9.90±0.300	17.04±0.439	**3.31±0.317**	16.65±0.565	14.86±0.782
	Worst	13.93	21.13	**8.63**	20.87	20.77
	Fevals±StE	-	-	-	-	-

Table A.39: Results for multi-point adaptive DRS using rule 3 on Lennard-Jones problems 2–11

	Adapting	**PSO-DRS *Fixed***	**PSO-DRS NP+K**	**PSO-DRS NP+PHI**	**PSO-DRS K+PHI**	**PSO-DRS NP+K+PHI**
LJ_{12}	Success	0%	0%	0%	0%	0%
	Best	10.78	13.65	**1.72**	4.32	2.40
	Mean±StE	14.80±0.282	21.95±0.463	**6.92±0.785**	20.91±0.794	19.88±0.889
	Worst	**19.36**	26.44	21.05	27.56	26.53
	Fevals±StE	-	-	-	-	-
LJ_{13}	Success	0%	0%	0%	0%	0%
	Best	13.73	16.91	**3.89**	14.07	13.17
	Mean±StE	20.55±0.366	28.17±0.487	**9.57±0.800**	26.67±0.715	26.86±0.542
	Worst	**24.28**	32.95	26.08	33.47	31.73
	Fevals±StE	-	-	-	-	-
LJ_{14}	Success	0%	0%	0%	0%	0%
	Best	18.80	27.09	**4.01**	12.98	7.68
	Mean±StE	24.15±0.315	31.84±0.307	**13.93±1.15**	28.97±0.723	30.12±0.843
	Worst	**29.80**	35.04	29.98	36.29	35.83
	Fevals±StE	-	-	-	-	-
LJ_{15}	Success	0%	0%	0%	0%	0%
	Best	23.14	28.08	**1.91**	22.89	16.31
	Mean±StE	28.32±0.322	36.53±0.340	**15.44±1.28**	35.44±0.552	34.08±0.785
	Worst	**32.09**	40.19	34.03	40.88	41.17
	Fevals±StE	-	-	-	-	-
LJ_{16}	Success	0%	0%	0%	0%	0%
	Best	25.80	35.43	**2.50**	21.32	16.36
	Mean±StE	32.45±0.353	41.29±0.317	**18.09±1.52**	38.35±0.794	37.13±1.00
	Worst	**37.62**	44.63	40.81	45.49	45.58
	Fevals±StE	-	-	-	-	-
LJ_{17}	Success	0%	0%	0%	0%	0%
	Best	30.52	25.50	**4.13**	23.79	8.85
	Mean±StE	37.13±0.408	45.21±0.550	**22.53±1.63**	43.71±0.720	41.66±1.22
	Worst	**42.20**	52.80	44.94	49.55	50.41
	Fevals±StE	-	-	-	-	-
LJ_{18}	Success	0%	0%	0%	0%	0%
	Best	38.82	39.57	**1.68**	23.09	28.29
	Mean±StE	43.23±0.261	50.04±0.515	**25.79±1.88**	48.33±0.934	48.71±0.766
	Worst	**47.29**	54.80	50.57	55.48	55.60
	Fevals±StE	-	-	-	-	-
LJ_{19}	Success	0%	0%	0%	0%	0%
	Best	44.26	48.92	**3.25**	34.07	19.97
	Mean±StE	49.28±0.336	56.23±0.365	**25.12±1.99**	53.74±0.807	53.38±1.17
	Worst	53.42	60.86	**53.13**	61.91	60.78
	Fevals±StE	-	-	-	-	-
LJ_{20}	Success	0%	0%	0%	0%	0%
	Best	45.13	55.86	**9.40**	44.89	24.91
	Mean±StE	53.59±0.409	61.79±0.281	**34.55±2.20**	60.01±0.509	58.55±1.01
	Worst	**58.92**	65.94	60.17	64.77	65.45
	Fevals±StE	-	-	-	-	-
LJ_{26}	Success	0%	0%	0%	0%	0%
	Best	80.37	81.67	**22.03**	74.81	64.96
	Mean±StE	84.92±0.285	92.57±0.392	**57.86±2.61**	89.80±0.660	88.39±1.14
	Worst	**88.98**	96.86	90.99	96.99	96.45
	Fevals±StE	-	-	-	-	-
LJ_{38}	Success	0%	0%	0%	0%	0%
	Best	145.2	149.4	**81.25**	139.4	120.1
	Mean±StE	150.3±0.265	157.7±0.381	**127.1±3.29**	156.0±0.822	152.9±1.09
	Worst	**153.8**	162.0	154.3	163.6	163.5
	Fevals±StE	-	-	-	-	-

Table A.40: Results for multi-point adaptive DRS using rule 3 on Lennard-Jones problems 12–20, 26, and 38

Appendix B

Publications

The following publications were derived from or influenced by this work:

D. Bratton and J. Kennedy., Defining a Standard for Particle Swarm Optimization. In *Proc of the Swarm Intelligence Symposium*, pages 120127, Honolulu, Hawaii, USA, 2007. IEEE.

D. Bratton and T. Blackwell., Understanding Particle Swarms through Simplification: A Study of Recombinant PSO. In *Proc. of the GECCO-2007 Workshop on Particle Swarms: The Second Decade*, London, UK, 2007.

D. Bratton and T. Blackwell., A Simplified Recombinant PSO. *J. Artif. Evol. App.*, 2008:14:114:10, January 2008.

R. Poli, D. Bratton, T. Blackwell, and J. Kennedy., Theoretical Derivation, Analysis and Empirical Evaluation of a Simpler Particle Swarm Optimiser. In *IEEE Congress on Evolutionary Computation*, pages 1955-1962, 2007.

T. Blackwell and D. Bratton., Origin of bursts. In *Proc. of the GECCO-2007 Workshop on Particle Swarms: The Second Decade*, London, UK, 2007.

T. Blackwell and D. Bratton., Examination of Particle Tails. *J. Artif. Evol. App.*, 2008:18:118:10, January 2008.

References

[1] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Nagoya, Japan, 1995. IEEE Service Center, Piscataway, NJ.

[2] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc of the IEEE International Conference on Neural Networks*, volume IV, Perth, Australia, 1995. IEEE Service Center, Piscataway, NJ, IV: 1942-1948.

[3] J. Kennedy. The behavior of particles. In *Proc of the 7th Annual Conference on Evolutionary Computing*, volume 1447, pages 581–589, San Diego, CA, USA, 1998. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany.

[4] J. Kennedy. Seminar series on particle swarm optimization, Department of Computer Science, University of Essex, UK, 2006.

[5] Z-H. Zhan, J. Zhang, Y. Li, and H.S-H. Chung. Adaptive Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39(6):1362–1381, 2009.

[6] X. Hu and R. C. Eberhart. Adaptive particle swarm optimization: Detection and response to dynamic systems. In *Proc of the IEEE Congress on Evolutionary Computation*, pages 1666–1670, Honolulu, Hawaii USA, 2002.

[7] W. Zhang, Y. Liu, and M. Clerc. An adaptive PSO algorithm for reactive power optimization. *IEE Conference Publications*, 2003(CP497):302–307, 2003.

[8] A. Rama Mohan Rao and K. Sivasubramanian. Multi-objective optimal design of fuzzy logic controller using a self configurable swarm intelligence algorithm. *Computers & Structures*, 86(23-24):2141 – 2154, 2008.

[9] M. Clerc. Tribes and multiobjective optimization, 2003.

[10] 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, 2002.

[11] M. Clerc. Tribes, a parameter free particle swarm optimizer. In *Proc of the OEP*, Paris, France, 2003.

[12] R. Mendes. *Population Topologies and their Influence in Particle Swarm Performance*. PhD thesis, Escola de Engenharia, Universidade do Minho, 2004.

[13] G.B. Dantzig. Programming of interdependent activities II. Mathematical model. *Econometrica*, 17:200–211, 1949.

[14] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[15] R. B. Wilson. *A simplicial algorithm for concave programming.* PhD thesis, Harvard University, 1963.

[16] D. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Computer Science and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, New York, 1982.

[17] P. M. Morse and H. Feshbach. Asymptotic series: Method of steepest descent. In *Methods of Theoretical Physics*, pages 434–443. McGraw-Hill, New York, New York, 1953.

[18] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964.

[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[20] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[21] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In A Iserles, editor, *Acta Numerica 2004*. Cambridge University Press, 2003.

[22] D. H. Wolpert and W. G. Macready. No Free Lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[23] D. H. Wolpert and W. G. Macready. No Free Lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[24] J.C. Culberson. On the futility of blind search: An algorithmic view of "No Free Lunch". *Evolutionary Computation*, 6(2):109–127, 1998.

[25] C. Schumacher, M. D. Vose, and L. D. Whitley. The No Free Lunch and problem description length. In *Proc of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 565–570, San Francisco, CA, USA, 2001.

[26] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, USA, 1995.

[27] A.S. Fraser. Simulation of genetic systems by automatic digital computers. *Aust. J. Biol. Sci.*, 10, 1957.

[28] A. Fraser and D. Burnell. *Computer Models in Genetics*. New York, McGraw-Hill, 1970.

[29] J.L. Crosby. *Computer Simulation in Genetics*. London, John Wiley & Sons, 1973.

[30] S. Ronald. Robust encodings in genetic algorithms: A survey of encoding issues. In *Proc of the 1997 IEEE International Conference on Evolutionary Computation*, pages 43–48, 1997.

[31] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[32] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[33] K.H. Liang, X. Yao, and C. Newton. Dynamic control of adaptive parameters in evolutionary programming. In *Proc of the Second Asia-Pacific Conference on Simulated Evolution and Learning*. Springer-Verlag, 1998.

[34] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.

[35] X. Yao and Y. Liu. Fast evolutionary programming. In L.J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proc of the 5th Annual Conference on Evolutionary Programming*. MIT Press, 1996.

[36] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis)*. Reprinted by Fromman-Holzboog (1973), 1971.

[37] H-P. Schwefel. *Numerische Optimierung von Computer-Modellen (PhD thesis)*. Reprinted by Birkhuser (1977), 1974.

[38] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, University of California at Berkley, 1995.

[39] J. Vestersträm and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Proc of the 2004 Congress on Evolutionary Computation*, volume 2, pages 1980–1987, 2004.

[40] M. Millonas. *Swarms, phase transitions, and collective intelligence.* Computational Intelligence: A Dynamic System Perspective. IEEE Press, Piscataway, NJ, USA, 1994.

[41] R. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools*. Morgan Kaufmann Publishers, 1996.

[42] J.M. Bishop. Stochastic searching networks. In *Proc. 1st IEE Int. Conf. on Artificial Neural Networks*, pages 329–331. Chapman & Hall, 1989.

[43] S. Hurley and R.M. Whitaker. An agent based approach to site selection for wireless networks. In *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, pages 574–577, New York, NY, USA, 2002. ACM.

[44] H. J. Grech-Cini and G.T. McKee. Locating the mouth region in images of human faces. *Sensor Fusion VI*, 2059(1):458–465, 1993.

[45] N. I. Katevas, N. M. Sgouros, S. G. Tzafestas, G. Papakonstantinou, P. Beattie, J. M. Bishop, P. Tsanakas, and D. Koutsouris. The autonomous mobile robot SENARIO: A sensor aided intelligent navigation system for powered wheelchairs. *Robotics & Automation Magazine, IEEE*, 4(4):60–70+, 1997.

[46] S.J. Nasuto, J.M. Bishop, and S. Lauria. Time Complexity Analysis of the Stochastic Diffusion Search. In *Neural Computation*, pages 260–266, 1998.

[47] S. J. Nasuto and J. M. Bishop. Convergence Analysis of Stochastic Diffusion Search. *Parallel Algorithms Appl.*, 14(2):89–107, 1999.

[48] D.R. Myatt, J.M. Bishop, and S.J. Nasuto. Minimum stable convergence criteria for stochastic diffusion search. *Electronics Letters*, 40(2):112–113, 2004.

[49] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[50] N. Holden and A. Frietas. A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *J. Artif. Evol. App.*, 2008:2:1–2:11, January 2008.

[51] A. Bauer, B. Bullnheimer, R. Hartl, and C. Strauss. Minimizing total tardiness on a single machine using ant colony optimization. *Central European Journal of Operations Research*, 8:125–141, 2000.

[52] A. Shmygelska, R. Aguirre-Hernndez, and H. Hoos. An ant colony optimization algorithm for the 2D HP protein folding problem. In *Proc. 16th Canad. Conf. Artif. Intell*, pages 400–417. Springer-Verlag, 2002.

[53] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.

[54] F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. *The Ubiquity of Chaos*, 1990.

[55] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proc of the 1998 IEEE Conference on Evolutionary Computation*, pages 69–73, Anchorage, AK, 1998.

[56] Y. Shi and R. Eberhart. Empirical study of particle swarm optimization. In *Proc of the Congress on Evolutionary Computation*, pages 1945–1959, Washington, D.C., USA, 1999. IEEE Service Center, Piscataway, NJ, USA.

[57] R. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proc of the 2001 Congress on Evolutionary Computation (CEC'01)*, volume 1, pages 94–100, 2001.

[58] Y. Zheng, L. Ma, L. Zhang, and J. Qian. On the convergence analysis and parameter selection in PSO. In *Proc of the 2003 IEEE International Conference on Machine Learning and Cybernetics*, pages 1802–1807. IEEE Press, Piscataway, NJ, USA, 2003.

[59] M. Clerc. Personal communication on the development of the constriction method, 2006.

[60] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proc of the 2000 Congress on Evolutionary Computation*, volume 1, pages 84–88, 2000.

[61] J. Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proc of the 1999 IEEE Congress on Evolutionary Computation*, pages 1931–1938. IEEE Press, Piscataway, NJ, USA, 1999.

[62] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proc of the 2002 IEEE Congress on Evolutionary Computation*, pages 1671–1676, Honolulu, HA, USA, 2002. IEEE Press, Piscataway, NJ, USA.

[63] D. Watts and S. Strogatz. Collective dynamics of "small world" networks. *Nature*, 393:440–442, 1998.

[64] D. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.

[65] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[66] P. Suganthan. Particle swarm optimizer with neighborhood operator. In *Proc of the 1999 IEEE Congress on Evolutionary Computation*, pages 1958–1962. IEEE Press, Piscataway, NJ, USA, 1999.

[67] M. Clerc. Standard PSO 2006, available at http://www.particleswarm.info/, 2006.

[68] E. Ozcan and C. K. Mohan. Analysis of a simple particle swarm optimization system. *Intelligent Engineering Systems through Artificial Neural Networks*, 8:253–258, 1998.

[69] E. Ozcan and C. K. Mohan. Particle swarm optimization: Surfing the waves. In *Proc of the 1999 Congress on Evolutionary Computation*, pages 1939–1944, Piscataway, NJ, 1999. IEEE Service Center.

[70] T. Blackwell. Particle swarms and population diversity I: Analysis. In *Proc of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 9–13, 2003.

[71] T. Blackwell. Particle swarms and population diversity II: Experiments. In *Proc of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 14–18, 2003.

[72] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, 2001.

[73] I.C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 86:317–325, 2003.

[74] M. Clerc. Stagnation analysis in particle swarm optimisation, or what happens when nothing happens. Technical Report CSM-460, University of Essex, 2006.

[75] V. Kadirkamanathan, K. Selvarajah, and P.J. Fleming. Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, 10:245 – 255, 2006.

[76] R. Poli, D. Bratton, T. Blackwell, and J. Kennedy. Theoretical derivation, analysis and empirical evaluation of a simpler particle swarm optimiser. In *IEEE Congress on Evolutionary Computation*, pages 1955–1962, 2007.

[77] R. Poli and D. Broomhead. Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 134–141, New York, NY, USA, 2007. ACM.

[78] R. Poli. On the moments of the sampling distribution of particle swarm optimisers. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2907–2914, New York, NY, USA, 2007. ACM.

[79] J. Kennedy. Bare bones particle swarms. *IEEE Swarm Intelligence Symposium 2003*, pages 80–87, 2003.

[80] T. Richer and T. Blackwell. The Lévy particle swarm. In *Proc of the 2006 IEEE Congress on Evolutionary Computing*, pages 3150–3157, Vancouver, BC, Canada, 2006. IEEE Press, Piscataway, NJ.

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

[82] J. Pena, A. Upegui, and E. Sanchez. Particle swarm optimization with discrete recombination: An online optimizer for evolvable hardware. In *Proc of the First NASA/ESA conference on adaptive hardware and systems*, pages 163–170, 2006.

[83] C. A. Coello and M. S. Lechuga. MOPSO : A proposal for multiple objective particle swarm. In *Proc of the IEEE World Congress on Computational Intelligence*, volume 2, pages 1051–1056, Hawaii, 2002.

[84] T. Blackwell and P. Bentley. Dynamic search with charged swarms. In *Proc of the Genetic and Evolutionary Computation Conference 2002 (GECCO)*, pages 19–26, 2002.

[85] T. Blackwell and J. Branke. Multi-swarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 2005.

[86] V. Miranda and N. Fonseca. New evolutionary particle swarm algorithm (EPSO) applied to voltage/var control. In *Proc of the 14th Power Systems Computation Conference*, pages 1–6, Seville, Spain, 2002.

[87] J. Kennedy and R. Mendes. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(4):515–519, 2006.

[88] J. Kennedy. Probability and dynamics in the particle swarm. In *Proc of the IEEE Congress on Evolutionary Computation*, pages 340–347. IEEE Press, Piscataway, NJ, USA, 2004.

[89] P. Lévy. *Théorie de lAddition des Variables Aléatoires*. Gauthier-Villars, Paris, France, 1937.

[90] M. Clerc. *Particle Swarm Optimization*. ISTE, 2006.

[91] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Proc of the Fifth Biannual World Automation Congress*, volume 13, pages 265–270, 2002.

[92] K. Yasuda and N. Iwasaki. Adaptive particle swarm optimization via velocity feedback. In A. Abraham, Y. Dote, T. Furuhashi, M. Köppen, A. Ohuchi, and Y. Ohsawa, editors, *Soft Computing as Transdisciplinary Science and Technology*, volume 29 of *Advances in Soft Computing*, pages 423–432. Springer Berlin / Heidelberg, 2005.

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

[94] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[95] X. Hu and R. C. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *Proc of the Sixth World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[96] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[97] J.M. Yang, Y.P. Chen, J.T. Horng, and C.Y. Kao. *Applying Family Competition to Evolution Strategies for Constrained Optimization*, volume 1213 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1997.

[98] A. Osyczka. *Multicriteria optimization for engineering design*. Design Optimization. Academic Press, Inc., New York, NY, USA, 1985.

[99] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proc. of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[100] K. Parsopoulos and M. Vrahatis. Particle swarm optimization in multiobjective problems. In *Proc of the ACM 2002 Symposium on Applied Computing*, pages 603–607, 2002.

[101] K. Parsopoulos and M. Vrahatis. Particle swarm optimizer in noisy and continuously changing environments. *Artificial Intelligence and Soft Computing*, pages 289–294, 2001.

[102] X. Li, T. Blackwell, and J. Branke. Particle swarm with speciation and adaptation in a dynamic environment. In *Proc of the 8th annual Conference on Genetic and evolutionary computation*, pages 51–58, Seattle, WA, USA, 2006.

[103] T. Blackwell and P. Bentley. Don't Push Me! Collision-Avoiding Swarms. In *Proc of the IEEE Congress on Evolutionary Computation 2002*, volume 2, pages 1691–1696, 2002.

[104] T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, 3005:489–500, 2004.

[105] R. Poli. An analysis of publications on PSO applications, Tech Report CSM-469. Technical report, Department of Computer Science, University of Essex, 2007.

[106] J. Salerno. Using the PSO technique to train a recurrent neural model. In *Proc of the 9th IEEE International Conference on Tools with Artificial Intelligence*, pages 45–49, 1997.

[107] R. Eberhart and X. Hu. Human tremor analysis using PSO. In *Proc of the 1999 IEEE Congress on Evolutionary Computation*, pages 1927–1930, Washington D.C., USA, 1999. IEEE Press, Piscataway, NJ, USA.

[108] Y. Lv, S. Li, C. Zhou, W. Guo, and Z. Xu. Two improved algorithms for multiple sequence alignment in a remote diagnose system for colonic cancer in pervasive environment. In *Proc of the First International Symposium on Pervasive Computing and Applications*, pages 145–150, 2006.

[109] S. Selvan, C. Xavier, N. Karssemeijer, J. Sequeira, R. Cherian, and B. Dhala. Parameter estimation in stochastic mammogram model by heuristic optimization techniques. *IEEE Transactions on Information Technology in Biomedicine*, 10:685–695, 2006.

[110] R. Xu, G. Anagnostopoulos, and D. Wunsch. Multiclass cancer classification using semisupervised ellipsoid artmap and particle swarm optimization with gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 65–77, 2007.

[111] R. Eberhart and Y. Shi. Particle swarm optimization: Developments, applications, and resources. In *Proc of the 2001 Congress on Evolutionary Computation (CEC'01)*, volume 1, pages 81–86, 2001.

[112] V. Tandon. *Closing the gap between CAD/CAM and optimized CNC end milling (Master's thesis)*. PhD thesis, Purdue University, 2000.

[113] J. Peng, Y. Chen, and R. Eberhart. Battery pack state of charge estimator design using computational intelligence approaches. In *Proc of the 15th Annual Battery Conference on Applications and Advances*, pages 173–177, 2000.

[114] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Trans. on Power Systems*, 15(4):1232–1239, 2001.

[115] M. Alrashidi and M. El-Hawary. A survey of particle swarm optimization applications in power system operations. *Electric Power Components and Systems*, 34(12):1349–1357, 2006.

[116] H. Bai and B. Zhao. A survey on application of swarm intelligence computation to electric power system. In *Intelligent Control and Automation 2006 WCICA 2006 The Sixth World Congress on*, volume 2, pages 7587–7591, 2006.

[117] N. Jin and Y. Rahmat-Samii. Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations. *IEEE Transactions on Antennas and Propagation*, 55:556–567, 2007.

[118] M. R. AlRashidi and M. E. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13:913–918, August 2009.

[119] R. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In *Proc of the 7th International Conference on Evolutionary Programming*, pages 611–616, 1998.

[120] C. Monson and K. Seppi. Exposing origin-seeking bias in PSO. In *Proc of the 2005 Genetic and Evolutionary Computation Conference*, pages 241–248, 2005.

[121] D. K. Gehlhaar and D. B. Fogel. Tuning evolutionary programming for conformationally flexible molecular docking. In L.J. Fogel, P.J. Angeline, and T. Back, editors, *Proc of the Fifth Annual Conference on Evolutionary Programming*, pages 419–429. MIT Press, Cambridge, MA, 1996.

[122] J. Kennedy and R. Eberhart. Tutorial on particle swarm optimization. IEEE Swarm Intelligence Symposium, 2005.

[123] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[124] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:6570, 1979.

[125] J. Jaccard and C. K. Wan. *LISREL approaches to interaction effects in multiple regression*. Sage Publications, Thousand Oaks, CA, 1996.

[126] C. Lee and X. Yao. Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8:1–13, 2004.

[127] J. A. Hartigan and P. M. Hartigan. The dip test of unimodality. *The Annals of Statistics*, 13(1):70–84, 1985.

[128] M. Maechler. Dip test explorations. Technical report, ETH Zurich, Switzerland, 2009.

[129] T. Blackwell and D. Bratton. Origin of bursts. In *Proc. of the GECCO-2007 Workshop on Particle Swarms: The Second Decade*, London, UK, 2007.

[130] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proc of the Swarm Intelligence Symposium*, pages 120–127, Honolulu, Hawaii, USA, 2007. IEEE.

[131] T. Blackwell and D. Bratton. Examination of particle tails. *J. Artif. Evol. App.*, 2008:18:1–18:10, January 2008.

[132] D. Bratton and T. Blackwell. A simplified recombinant PSO. *J. Artif. Evol. App.*, 2008:14:1–14:10, January 2008.

[133] J.E. Lennard-Jones. On the determination of molecular fields. *Proc. R. Soc. Lond.*, 106:463–477, 1924.

[134] P. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.

[135] M. Hoare. Structure and dynamics of simple microclusters. *Adv. Chem. Phys.*, 40:49–135, 1979.

[136] J.A. Northby. Structure and binding of Lennard-Jones clusters: $13 \leq n \leq 147$. *J. Chem. Phys.*, 87:6166–6177, 1987.

[137] M. Locatelli and F. Schoen. Fast global optimization of difficult Lennard-Jones clusters. *Comput. Optim. Appl.*, 21(1):55–70, 2002.

[138] B. Hartke. Global geometry optimization of atomic and molecular clusters by genetic algorithms. In *GECCO '01: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1284–1291, 2001.

[139] D. Daven, N. Tit, J. Morris, and K. Ho. Structural optimization of Lennard-Jones clusters by a genetic algorithm. *Chemical Physics Letters*, 256(1-2):195 – 200, 1996.

[140] D. Wales and J. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *J. Phys. Chem. A*, 101:5111, 1997.

[141] R. J. W. Hodgson. Particle swarm optimization applied to the atomic cluster optimization problem. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 68–73, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[142] A. Sutton, D. Whitley, M. Lunacek, and A. Howe. PSO and multi-funnel landscapes: How cooperation might limit exploration. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 75–82, New York, NY, USA, 2006. ACM.

[143] S. Call, D. Zubarev, and A. Boldyrev. Global minimum structure searches via particle swarm optimization. *Journal of Computational Chemistry*, 28(7):1177–1186, 2007.

[144] B. Secrest and G. Lamont. Visualizing particle swarm optimization - Gaussian particle swarm optimization. In *Proc of the IEEE Swarm Intelligence Symposium*, pages 198–204. IEEE Press, Piscataway, NJ, USA, 2003.

[145] D. Bratton and T. Blackwell. Understanding particle swarms through simplification: A study of recombinant PSO. In *Proc. of the GECCO-2007 Workshop on Particle Swarms: The Second Decade*, London, UK, 2007.