The Analysis of Behaviours in Swarm Systems

Adam Erskine

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2016

Abstract

In nature animal species often exist in groups. We talk of insect swarms, flocks of birds, packs of lions, herds of wildebeest etc. These are characterised by individuals interacting by following their own rules, privy only to local information. Robotic swarms or simulations can be used explore such interactions. Mathematical formulations can be constructed that encode similar ideas and allow us to explore the emergent group behaviours. Some behaviours show characteristics reminiscent of the phenomena of criticality. A bird flock may show near instantaneous collective shifts in direction: velocity changes that appear to correlated over distances much larger individual separations.

Here we examine swarm systems inspired by flocks of birds and the role played by criticality. The first system, Particle Swarm Optimisation (PSO), is shown to behave optimally when operating close to criticality. The presence of a critical point in the algorithm's operation is shown to derive from the swarm's properties as a random dynamical system. Empirical results demonstrate that the optimality lies on or near this point.

A modified PSO algorithm is presented which uses measures of the swarm's diversity as a feedback signal to adjust the behaviour of the swarm. This achieves a statistically balanced mixture of exploration and exploitation behaviours in the resultant swarm. The problems of stagnation and parameter tuning often encountered in PSO are automatically avoided.

The second system, Swarm Chemistry, consists of heterogeneous particles combined with kinetic update rules. It is known that, depending upon the parametric configuration, numerous structures visually reminiscent of biological forms are found in this system. The parameter set discovered here results in a cell-division-like behaviour (in the sense of prokaryotic fission). Extensions to the swarm system produces a swarm that shows repeated cell division. As such, this model demonstrates a behaviour of interest to theories regarding the origin of life.

Acknowledgements

Many thanks are due to the following people. To my supervisor Michael Herrmann for his guidance and suggestions. To my fellow student Thomas Joyce for his discussions and insights regarding particle swarm optimisation. Finally to my wife, Patricia, and son, Thomas for their continued patience and support.

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/K503034/1].

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. A version of the CriPS algorithm work, discussed in chapter 4, was presented at the 2015 European Conference on Artificial Life (Erskine and Herrmann, 2015a). The cell division like behaviour located within the Swarm Chemistry model, discussed in chapter 5, was originally presented at The European Conference on Artificial Life (ECAL) 2013 (Erskine and Herrmann, 2013). This work was extended and published in a special edition of the journal Artificial Life (Erskine and Herrmann, 2015b). The theory, implementation, experimental work, and analysis were done by me, and the papers were written by me, but were influenced through detailed discussions with my co-author and supervisor Michael Herrmann.

(*author*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Swarms are common in nature. Wherever we look we see accumulations of individual animals into larger groups. Honeybees, ants and termites form colonies tens of thousands strong. In the air birds flock by the thousand. Underwater, but in similar numbers, we find shoals of fish. On land great herds of wildebeest range across plains. Such structures can remain robust over great spatial and time scales. They frequently exhibit group behaviours that are hard to explain by examining the actions of the individual. The response of a swarm to external stimuli can appear near instantaneous. Analogous properties arise in other swarm systems such as the action of the neural interconnections in the brain. Again, we can describe this as a system of multiple individual elements, with many interactions. We can attempt to explore the mechanisms employed within a brain, by studying the structure of the interconnections, and the electrical and chemical activities. However, the properties and behaviours of the brain as a whole remain hard to understand from this neuron to neuron activity.

Bird flocks can often appear ‘as of one mind’, showing near instantaneous collective shifts in direction. Velocity changes appear to be correlated over much larger distances than it seems possible that the individual birds can be communicating. The loss of length and time scales, the ability to switch behaviours, are somewhat reminiscent of the properties of systems that exhibit what has been called ‘criticality’. We will discuss in more detail what criticality means in chapter 2. For the purpose of this work we define a system at criticality as being poised on an edge between two behaviours.

This thesis uses swarm systems inspired by flocking behaviours and suggest that criticality is a property that enables swarms to operate in a beneficial manner. In terms of solutions offered by the Particle Swarm Optimisation algorithm this is a question of

whether the swarm behaves in an optimal manner if it can be made to behave critically. With the Swarm Chemistry flocking system we shall show that an important discovered dynamic behaviour exists at a narrow range of parameter values that poises the system between two somewhat more static behaviours.

In particular the research questions that we seek answers to are:

1. Under what conditions does the particle swarm optimisation (PSO) algorithm exhibit criticality?

2. Does the PSO swarm acting critically allow it to act optimally?

3. Can this knowledge be utilised in a practical manner?

4. Can we show that other swarm systems also exhibit criticality?

1.1 Contributions

In answer to research question 1 chapter 3 shows that for PSO there exists a locus of the algorithm's parameter values that result in critical behaviour. On this locus the PSO swarm is poised between divergent and convergent states. The derivation of this is arrived at by rewriting the PSO update rules in matrix form. As the swarm evolves its dynamics are determined by considering the infinite product of this stochastic matrix.

This predicted locus of stability differs from the previous approaches. Empirical evidence is presented to support the correctness of the random dynamical system approach.

Empirically we look to see where in parameter space optimal results occur for many problem functions. We show, in answer to research question 2 that as the number of iterations executed increases the location of optimal solutions approaches our predicted critical parameter curve.

To answer research question 3 we observe that real world problems tend to present as a black box function. The ability of the PSO swarm to locate solutions is dependent upon the nature of the problem space, how long we will run the algorithm for and where in the parameter space we choose the controlling parameter values. Given knowledge of the first two items we can (at least in theory) intelligently choose how sub-critical we *need* the swarm to be to obtain optimal results. In practice this can still be hard. Thus a novel algorithm, CriPS, is proposed in chapter 4. This is shown to require no specific setting of parameter values and avoids stagnation. This is achieved by using a

measure of the swarm's diversity within the problem space. Changes in this measure are used as a feedback signal to modify the PSO parameters.

CriPS is shown to outperform other simple PSO variants. In comparison with other metaheuristics this new algorithm performed more modestly. It is shown that for at least one of the problem function sets CriPS outperforms all the comparator algorithms. The general performance may reflect the limitations of PSO as an algorithm rather than this new approach. The algorithm's update rules are shown to result in controlling PSO's parameters such they are varied so as to take the system's behaviour from sub- to super- critical (i.e. cross the locus of stability).

Finally we answer research question 4. In chapter 5 we show that critical behaviour in swarm systems is not restricted to PSO. A repeating cell-division-like behaviour is demonstrated within the system called swarm chemistry. This emerges from the repeated low level kinetic interactions between the particles in a heterogeneous swarm. This division behaviour exists in a small but finite volume of the swarm chemistry parameter space. It appears to sit on an edge of the parameter space between no division behaviours and markedly less dynamic division behaviours. And yet the behaviour was robust to population changes and even the addition of more species of particle.

Chapter 2

Introduction to swarms

As background to these studies, we discuss a number of mechanisms seen in swarms and swarm systems that lead to interesting behaviours. To avoid excessive repetition we may use terms such as swarm, colony, community, collective etc. to describe aggregations. In referring to the individuals we similarly employ many names dependent upon the scope of the discussion. For biological swarms it is easiest to use the name of the animal itself themselves: bees, ants, wildebeest, birds, fish etc. For abstract swarms we will call them particles, individuals, or agents. It is hoped that in this manner repetition may be avoided in the discussions without introducing confusion.

Pertinent to the studies will be the mechanisms employed to generate complex behaviours from the interactions among the members of a swarm. We start by looking at some of the concepts relevant to this. We will then outline the structure that this thesis will follow and the motivations that lie behind it. The final section of this introduction will set out the nature of the swarm systems that we are studying.

This thesis looks at swarm systems and behaviours that emerge from their dynamics. Algorithms derived from the flocking of birds have found use in fields such as computer graphics and function optimisation. It is in these areas that we will discuss new understandings and novel behaviours. Our interest is therefore in the remarkable capabilities of swarms, and the potential applications that arise from determining the role of individual interactions in the development of system-wide characteristics.

2.1 Swarms: emergence

Emergence[1] is the non-obvious high level characteristics of a system that arises from low level interactions of the constituent parts.

Honeybees live in colonies with tens of thousands of individuals. They coordinate dozens of activities that are required for the colony to successfully meet the challenges of life. What is remarkable is this is achieved without any central control being imposed. Instead a range of mechanisms are employed which mediate the interactions between the individual bees. Among these we see the use of various chemical signals, direct bee to bee communication, and quorum sensing. Similar approaches can be seen in other insect swarms, flocks of birds, herds of mammals, and amongst the coordination of slimemolds or bacterial colonies.

Emergence as a phenomenon is also apparent in other systems that consist of many interacting units. The immune system in many organisms provides an adaptive mechanism to protect against pathogens. Its ability to respond to a wide range of (potentially evolving) infections in a flexible and robust manner can be studied as a set of emergent behaviours. Similarly consciousness is suggested as an emergent property that arises from the mass of interactions between neurons within a brain. The phenotypes of species are themselves the emergent form of the genetic interactions during embryo development.

2.2 Swarms: self-organisation

The emergent behaviours have no master controller, capable of oversight of all the component parts at all times. Instead they arise through local interactions and local knowledge. As such they are said to be *self-organised* arising from interactions between individuals amplifying fluctuations in the behaviours of the colony. Such feedback can be both robust and highly reactive to the group's environment. The amplification is a result of feedback between the driving signal and the individual behaviour. Thus honeybees foraging for food may explore a multitude of sources (of potentially variable quality). When good ones are found, they will advertise this to their hive mates. New recruits will fly to exploit the sites advertised. This is a positive feedback mechanism. On exhaustion of the food supply, no further recruitment to it

[1] There is a Scots phrase: 'Mony a mickle maks a muckle'. It means that many small amounts accumulate to make a large amount. This seems a succinct description of emergence.

occurs, and foragers return to exploration (negative feedback).

2.3 Swarm systems

We can also explore beyond the biological swarm. Other scientific disciplines describe aggregate systems which yield properties that provide insight into emergent behaviours arising from individual interactions. These often touch on biological themes, but in a more abstract manner.

@Ball (2011b) describes a range of well know systems that may be considered as swarm systems including Cellular automata, Turing reaction-diffusion dynamics, and Belousov Zhabotinsky patterns. They show interesting emergent behaviours. Cellular automata (CA) are a set of models consisting of a regular grid of cells. Each cell may exist in one of a finite number of states. In the simplest case, on and off states. The model progresses by applying update rules for each element on the grid. These tend to be of the form where a cell's new state is in some way dependent upon state of both itself and of cells within a certain local neighbourhood. The famous two dimensional, two state system "Conway's game of life" consists of 4 simple rules but results in a vast array of emergent structures and behaviours. With different numbers of states and rule sets, cellular automata can show both trivial and complex behaviours. Continuous spatial automata can be specified, using differential equations to describe state evolutions. Alan Turing's reaction-diffusion textures are of this form and may be used to model the appearance of animal spots and stripes. Similarly the famous Belousov Zhabotinsky chemical reactions that give rise of oscillating patterns of chemicals can be generated by cellular automata.

The systems we explore may be described as inspired by the motions of bird flocks. However they are of an abstracted form, constructed from sets of discrete particles with continuous position and velocity states. Updates consist of iterative rules, somewhat akin to cellular automata.

2.4 Thesis structure

We have briefly covered some interesting features of swarms. In the next part of this chapter we review a range of mechanisms employed in biological swarms and how these confer benefits on the particular species employing them. In the process

of discussing these biological methods we detail how they have been abstracted and re-purposed to solve problems, or used within embodied robots.

As the systems that studied in this thesis are based on flocking behaviours, we will look at that particular behaviour in its own section, paying particular attention to issues most pertinent to the subsequent chapters. We continue by discussing the properties of more abstract swarm systems, inspired by, but perhaps not wholly derived from nature. Here we are thinking of systems such as sand-pile models that yield insights into areas as disparate as earthquakes and neural information processing. This provides an introduction to the role of criticality in swarm systems which is returned to in later chapters.

We look in Chapter 3 to derive a proper explanation of how the Particle Swarm Optimization(PSO) algorithm functions. PSO is one of a range of metaheuristic algorithms commonly used to find good solutions to optimisation type problems. A theoretical treatment of the system as a random dynamical system is presented. The swarm behaves critically for a locus of parameter values that define a curve in parameter space. We then compare the predictions made from this approach with empirical results, demonstrating that the optimal performance is achieved where the swarm behaves critically (or where practical constraints apply, sub-critically). However, if allowed to behave sub-critically, then stagnation will eventually occur.

Chapter 4 presents modifications to PSO so that the new algorithm is able to tune itself to the problem being solved by means of online adjustment of its parameters derived from a feedback signal taken from measurements of the swarm. The aim is to maintain the critical/sub-critical performance of the previous chapter but with a mechanism that ensures stagnation is avoided automatically and no parameters need to be set. This Critical PSO's (CriPS) performance is experimentally tested and compared with a range of comparator metaheuristics.

In Chapter 5 a different flocking system is examined. Swarm Chemistry, a heterogeneous extension of Craig Reynold's Boids algorithm generates convincing flocking motions from the iterated application of three simple rules (Reynolds, 1987). Originally conceived as a homogeneous collection of particles, it has in recent years been modified to allow its constituents to vary the strength of each rule. Complex behaviours in this system are found. Here a novel behaviour that presents as a repeated cell-division-like dynamic is examined. This exists in a finite but narrow area of the system's parameter space and shares the 'on the edge' feel of the earlier PSO critical locus of parameter values.

Finally we summarise our findings and discuss the potential for future work both in the realms of metaheuristics and Swarm Chemistry.

2.5 Thesis motivations

In swarm systems, patterns and behaviours may arise that are not obvious from the multiple simple interactions occurring between the individuals in the swarm. These are said to be emergent properties of the system. We describe this decentralized, self-organised emergence of useful behaviours as swarm intelligence. A continuing problem with the exploitation of swarm intelligence is in designing the low level interactions so that desirable high level behaviours emerge. One approach is to take inspiration from nature. Biological swarms, for example, ants and honeybees exhibit multiple behaviours that are necessary for the success of the colony. Each of these have had tens of behaviours described and listed (Chittka and Niven, 2009; Wilson, 1984). Observation and experiment allows us to unpick the relationships between low level interactions and the coordination of these colony-wide behaviours. These understandings provide useful inspirations for algorithms or swarm robotic systems.

Ants and honeybees species are much studied and exhibit many complex behaviours that are robustly performed, effectively without central control. This is by no means unusual, the use of swarm intelligence to solve the challenges of existence is widespread. Other examples include: the synchronisation of firefly flashing in order to provide a large display for the purposes of attracting mates; bird flocking and herding of wildebeest to avoid predation; quorum sensing in colonies of bacteria to allow a group response to changes in the chemical environment.

Biology provides many inspirations. Physics and Maths may yield yet more. The Scottish biologist and mathematician Sir D'Arcy Wentworth Thompson (1860-1948) was a great exponent of the role that physical processes may play in the morphological development of creatures and their artefacts. The processes at all scales of life appear complex. Yet uncovering the rules that underpin these processes may shed light on the forms that life can take. For example, Thompson saw that the forms that soap bubbles took as their surface energies pulled on each other until equilibria were found bore resemblances to biological forms. Thomson suggested a number of examples from nature that may gain their structure from bubble like growth mechanisms. These included the structure of honeycomb, the cone cells in a fly's eye, and the defensive inorganic structures employed by certain sponges (Thompson, 1917). Time has

demonstrated that his beliefs were not always correct. The structure of honeycomb: its hexagonal packing and shape of end caps, are not formed from bubble interactions (bees just build them, perhaps using rule-based methods) (Ball, 2011b). However the packing of the four cones per ommatidia of a fly's compound eye may be due a mechanism akin to the bubble-like squeezing together of the cells as they grow. Ball also documents work that notes that the spicule structures of sponges appears to form via a mechanism whereby a bubble array is created and then the inorganic compounds are allowed to permeate the interstices of the bubble matrix. The creature is leveraging the free structure from what Ball refers to as a fossilised foam.

Examples of this interaction between species and the physics of the environment abound. *Velella valella* is a small pleuston[2] Cnidarian species somewhat like a jellyfish. They float as flat, blue, jelly-like plates on the ocean surface. Beneath the surface small tentacles hang waiting to catch prey. Above water they have a small vertical fixed sail-like structure, angled to catch the wind, yielding their common name, "by-the-wind sailor". This provides the animal with its motive force. The species exists in two forms, with differing chirality: the sail curved left or right. The action of the wind upon each subgroup pushes them in different directions. As a result playwright Nick Darke described the species as "sorted by the wind" (Darke, 2004). The interplay of the wind and the animal's physical structure results in a simple behaviour that (presumably) is beneficial for the species.

A further example of a species exploiting physics for its gain is seen in the mechanism by which the spores of mushrooms may be released when ripe. The Buller's drop mechanism (Money, 1998), consists of a small drop of liquid which sits near the portion of the mushroom where the ripening spore sits. As it ripens sugars in the cell wall act as a point where water vapour can condense. As this occurs the newly condensed drop and the extant "Buller's drop" coalesce. The resultant sudden change in center of mass provides the force that expels the spore. Here the species exploits the both the condensation point of water and forces available from a sudden shift of mass.

The boon of self-organization is that the emergent forms are, in some sense, available for free. Structure emerges from interactions without the need for it to be explicitly coded. An understanding of these rules and their application allow us the possibility of re-using this free structure in robotic systems. Allowing swarms of robots to self-organise into useful structures, to self-repair or to reproduce structures without the need for explicit command and control is beneficial. Explicit control is often hard

[2] A species living on the surface of water.

to achieve, particularly if it is required to be robust to changing environments. This is important as swarms built with central control suffer from problems as the number of individuals grow. Group fitness tends to fall off once numbers increase over some threshold. Jimenez (2005) cited by Kernbach et al. (2009) shows that "increasing the number of robots that work on one problem in parallel, does not lead to linear or sub-linear improvement in the collective performance.". Avoiding the need for central control would appear to offer a solution for this. Şahin (2005) describes how these emergent behaviours are flexible, scalable, and robust. These characteristics are appealing to swarm robotics. The problem becomes how to select which simple individual actions are required to achieve a given high level behaviour. We can use the biological examples as our inspiration. This can be approached in two ways. Either, we abstract from nature to develop algorithms that are useful, or we can simulate nature more closely and hope that our results are both useful and provide biological validation.

Modular robotics is an active area of research and systems have been looked at that involve re-configurable lattices, chains or swarms. Sets of robots act together to bridge gaps, or modify their arrangement to form useful structures. Lattices and chains exist as connected robots that modify the angles between their component parts to achieve this. Whereas modular swarms exist as a set of independent units able to rearrange themselves. The Swarm Chemistry system we study later, whilst explored in simulation form, is of this latter category. It is hoped that Swarm Chemistry provides a model that allows us to look at the automatic creation of morphological artifacts. One of its appeals is that it is a simple model that provides insights into the mechanisms at play in the self-organization of structure. A recent striking example of swarm morphogenesis is Harvard's Kilobot project (Rubenstein et al., 2014). This uses a swarm of 1024 simple robots to form two-dimensional shapes robustly in a decentralised manner using only local information, a defined starting position origin, and neighbour to neighbour communications.

Biological swarms solve the problems of life e.g. how to efficiently locate and exploit food, avoid predation, find shelter etc. Swarms are thus abstracted and used to create algorithms that are useful. Many have been developed across different domains. A partial list includes solutions to : Travelling salesman problems; network routing, search and optimisation algorithms; information sharing over networks; and job shop scheduling (Karaboga and Akay, 2009; Wikipedia, 2015a,b).

Whether embodied or abstracted we can use swarm inspirations to bootstrap solutions to problem domains that we are interested in.

2.6 On swarms in nature and the nature of swarms

The simplest organisms perhaps have little or no choice in how they navigate through their world. Simple Brownian motion and fortuitous happenstance guide their existence. More complex organisms come replete with sensors to perceive properties in their environment, and the means to interact with the world. Braitenberg (1986) imagined a sequence of vehicles with ever more complex sensorimotor linkages and argued that increasingly complex behaviours in his vehicles could manifest as a result. A simple example would use two bi-lateral light sensors cross connected to motor driven wheels. The resultant robot would turn towards the strongest light source. Walter Grey had earlier built "tortoise" robots that demonstrated these sort of autonomous behaviours in embodied robots. In nature many behaviours can be explained by an organism's response to a stimulus. The term taxis is used to describe this. It may be the result of a signal in the environment or via communication with other organisms. For example it is known that lobsters locate food sources by being guided by turbulent odour plumes in their environment (Moore et al., 1991). Studying such mechanisms allows inspiration to be drawn and used to re-target the mechanisms in, say, a robot controller (Grasso et al., 2000). Our interest is in taxis type response in colonial groups and how they contribute to the colony. Many stimuli may be exploited. Organisms respond to an enormous range of stimuli including: oxygen levels, electric fields, sound and light. Responses may result in a range of movements e.g. toward, away from, maintain a constant angle to etc. We discuss a few examples below.

2.6.1 Chemotaxis

The lobster's ability to locate food sources from the plumes of chemical signals it perceives in its environment is an example of chemotaxis. Chemical signals are also used by many species that exist in groups. Many ant species, for example, use pheromone signalling to aid their swarm's coordination. The ant's chemotaxis is driven by the ants altering their own environment (this is called stigmergy, see section 2.6.6. Ant colony foraging is an emergent behaviour. Individual ants leave their nest and radiate outwards looking for food. An individual ant that finds food, transports it back to its nest. It marks the route back with a chemical signal (pheromone). This acts to recruit more ants to follow the path to the food. Each returning ant reinforces the signal. This ensures that a found food source is exploited by the colony. When the food is exhausted, no more pheromone is laid, the signal decays and ants cease to

visit that site. Pheromone led ant foraging has been used as inspiration for a range of algorithms proposed by M. Dorigo such as Ant Colony Optimisation and Ant Colony System (Bonabeau et al., 1999). Route finding type problems may be cast as graph networks: The nodes are locations to be visited, the edges labelled with the transit costs. As virtual ants travel through network, the quality of the travelled path is evaluated. At each node an ant chooses its next edge based on the relative strength of the pheromones laid along the edges (usually via a stochastic process that preserves a modicum of exploration of apparently poor routes). Successful ants (those who find good paths) get to deposit pheromone along the paths they travelled. Finally signals evaporate after each update. Parameters controlling stochasticity of path selection, evaporation rate etc. control this family of algorithms. The algorithm demonstrates that the biological assumptions are sufficient for useful behaviour to emerge.

Reproductive drivers may also lead to collective behaviours. Cellular slimemolds exist as single celled organisms. When food is short the individual cells (of some species) release a chemical signal. This causes nearby cells to coalesce into a swarm, forming a slug-like composite creature. In this form it rises through the soil to the surface. Some cells form a stalk, others become spores that may be released into the air to spread further afield.

2.6.2 Thermotaxis

Temperature regulation is important to the survival of a honeybee colony. A hive needs to be able to maintain its temperature well above ambient winter temperatures in order to survive through to spring. When too cold, the bees will cluster, exercise their flight muscles raising their body temperature. If too warm, the bees gather together and flap their wings to promote airflow, or in extreme cases spread water over surfaces and utilise its evaporative cooling. Jones et al. (2004) describes an experiment where the temperature inside a honeybee hive is measured. Individually, honeybees are poikilothermic (unable to regulate their temperature). A honeybee colony, however, is homeothermic. The thermo-regulation thus emerges from their collective actions. Honeybee temperature controls are also known to be used to thwart disease (Starks et al., 2005). Bonabeau et al. (1999) describes bees forming chains by their combs to induce a local increase in temperature to soften the wax to allow easier shaping. Brood development requires an optimal temperature of $\approx 35°C$ (Jones et al., 2004) to ensure full development of their young. If the temperature varies outside this range for any

lengthy time the development of the pupa and larva may be disrupted. Grodzicki and Caputa (2005) detail experimental results looking at temperature preferences among honey bees. Temperature preference varies depending on whether solo bees or groups were tested. Other variations in preference were observed in younger bees (Kengyel et al., 2009; Stabentheiner et al., 2010). The authors conducted a series of experiments exploring the response of groups of honeybees to environments featuring different temperature profiles. It was shown that young bees prefer warm areas, and when placed in a linear thermal gradient will move at random with slight bias toward their ideal temperature. However, a group of bees move randomly but on colliding with other bees will pause before moving on. The duration of the pause was found to increase as a function of temperature. As the bees moved the multiple interactions caused the bees to group in the area of preferred temperature. An algorithm (BEECLUST) based on these experiments has been applied to small groups of robots resulted in similar clustering behaviours in response to light preferences (light acting as a temperature proxy) (Bodi et al., 2009; Kengyel et al., 2009; Kernbach et al., 2013; Schmickl and Hamann, 2011).

2.6.3 Phototaxis

Response to light seems universal. As an energy source it drives plant growth through the day. In animals the evolution of single light sensitive cells to functioning eyes appears to have occurred so frequently that the benefits it gives an organism must be immense. Phototaxis in cockroaches is used to drive them to hide in deeper recesses. Cockroaches also like to remain in groups. This latter characteristic has been exploited to steer cockroaches to more accessible locations. In this interesting experiment (Halloy et al., 2007) robot cockroaches are introduced to groups of real cockroaches. When presented with multiple hiding places with different levels of illumination, the robots can guide the real cockroaches to brighter areas than they would favour on their own.

There are species of South-East Asian fireflies that are able to synchronise their flashing in order to attract mates from further afield. Models describe the process somewhat like the charging and discharging of a capacitor. Thus a firefly *charges* up ready to flash, but if near neighbours flash, then the original fly resets the timer on its charging process. As a result the flashing becomes synchronised. The swarm may be spread across the branches of whole trees. The fireflies at one side of the tree are

too far from those on the other to perceive them directly, but the accumulation of the responses to local information coordinates the whole swarm. The simple algorithm self-organises this high level emergent behaviour. It is assumed that the species' females are now able to see the display from farther afield. Firefly flashing has been utilised as inspiration for the Firefly Algorithm (FA). This is a metaheuristic (akin to PSO) whose candidate solutions (its fireflies) flash to indicate the quality of their current solution. Brighter flashes for better solutions. The fireflies are then attracted to each other by the perceived brightness of the flashes (this being a function of absolute magnitude and distance between the fireflies) (Yang, 2009).

2.6.4 Trophallaxis

Trophallaxis is the word used to described the mouth to mouth exchange of liquid foods. This is seen in many species from vampire bats and birds to bees, wasps and ants. In honeybees it provides a mechanism both for the feeding of members of the colony and as a means to communicate hive wide information to the individual colony members. Crailsheim (1998) reviews the multitude of uses that food exchange plays in a hive. Young bees (1 day or so old) are unable to digest pollen directly and rely on nurse bees to process it for their consumption. Drones and the Queen rely on food given to them for survival. Foragers, returning with nectar, pass their load to other bees for storage in the hive. This division of labour allows the foragers to maximally exploit good food sources before they are discovered by other bee colonies. This exchange of food is widespread through the colony and is likely to act as a flow of information as much as a flow of food. Pollen gathering by pollen foragers is regulated by the feeding of pollen to the foragers (Camazine, 1993). If pollen is plentiful in the hive then foragers will be likely to be fed pollen. This inhibits their desire to gather further pollen and they are likely to switch to collecting nectar instead. This information flow has been used (Schmickl and Crailsheim, 2008) as the basis of a swarm robotic foraging algorithm. Their agents picked up virtual food items and consumed them as they moved. When they collided with other agents they exchanged information about their current food levels. This provided the agents with a food gradient which to move up (toward food) or down (toward home). They found that they needed to include agents that wandered and exchanged information but that did not follow the gradients in order to ensure that enough agents were present between the food source and home for effective information exchange to occur.

2.6.5 Aggregation and task allocation in swarm robotics

Swarm robotics is an active area of research. Models of insect interactions (as noted above) are often used to inspire agents of such swarms. This allows the agents to act independently accessing only local information. Yet, if suitably designed, group goals may be achieved. It is not necessary to design the high level complex behaviours if one can build low level actions whose multiple interactions give rise to the desired goals. The previously mentioned thermotaxis of the honeybee has been implemented in embodied agents to show aggregation behaviours in an algorithm called BEECLUST (Schmickl and Hamann, 2011). Well known algorithms derived from biological inspirations utilise their agent-agent or agent-environment local interactions e.g. ant colony optimisation ACO, artificial bee colony algorithm (ABC) (Bonabeau et al., 1999). Several review documents highlight the extensive range of behaviours that may be implemented from swarm robotic interactions, including: pattern formation; aggregation; chain formation; self-assembly; coordinated movement; hole avoidance; foraging; self-deployment; grasping; pushing; caging (Bayindir and Sahin, 2007; Mohan and Ponnambalam, 2009). Swarm robotic morphogenesis is an interesting field. Historically different approaches can be placed in one of three categories: lattices of interconnecting units; agents organising in chains or tree structures; or arising from mobile swarm interactions (Yim et al., 2007). The various flocking algorithms discussed below fall into the mobile approach and could be used to construct dynamic complex structures or patterns.

2.6.6 Stigmergy

Stigmergy is a mechanism of indirect coordination between agents or actions. The principle is that the trace left in the environment by an action stimulates the performance of a future action (or decision), by either the same or a different agent. Thus collective actions can arise through the interactions between individuals mediated via the exchange of information stored in the environment.

An example of stigmergy arises in the food storage within honeybee hives. Seeley et al. (1995) describes how from the age of about 12 days a bee leaves the central broodnest area where it has been caring for young, feeding others, and cleaning. Now it takes up the role of a food storer. It takes food from returning foragers and stores it in empty cells. It will also pack pollen into cells. It will, as necessary, also take part in

other behaviours, contributing to hive cooling, feeding other bees, building comb and protecting the hive. These tasks are carried out until the bee is about 20 days old and once again changes role and becomes a forager.

The bees become food storers by receiving food from foragers. They then look to store the food in empty cells. This deposition mechanism is essentially stigmergic: they simply respond to the presence of a laden forager by taking the food; or to the presence of an empty cell by depositing their load. However, even such simplistic following of deterministic rules can yield interesting group behaviours. Holland and Melhuish (1999) describe an experiment using a robot swarm to move pucks in an arena. By following simply deterministic rules, the combined effect is to cluster the pucks into larger and larger groups. The ant colony optimisation algorithm is itself stigmergic (Bonabeau et al., 1999). Grassé (1959), (cited by Ajith et al. (2006)), describes the stigmergic rules followed by termites to construct their grand mounds. Often stigmergy and taxis combine. Thus the royal chamber in termite colonies are driven by pheromone gradients. These define the radii at which to construct the wall of the chamber: essentially a command that says "move away from the queen until the signal is less than some threshold". The termites will then follow this radii until a gap in the current build is locate and fill it in. The presence of built wall stops the termite from depositing its load. Simulation of these rules captures the essence of this (Hill and Bullock, 2015). Similarly Camazine et al. (2001) discuss a rule based model that results in the structures seen in honeybee combs: a central brood area; surrounded by pollen and then nectar concentric zones.

2.6.7 Quorum sensing

Quorum sensing is another biological process that has yielded useful inspiration. It is another mechanism that uses local information and is seen in a variety of biological sources (Miller and Bassler, 2001). It may be described as a stimulus/response mechanism based on population density or of some form of group "voting". In bacteria it refers to the fact that gene expression can be switched depending on local cell population densities. *Vibrio fischeri* is perhaps the most studied of these. Its quorum sensing system results in the presence or absence of bioluminescence. This bacteria exists symbiotically in a number of different creatures and serves differing purposes. In the squid *Euprymna scolopes* it is used as an anti-predation device. High concentrations of the bacteria exist on the creatures underside, shining downward,

effectively hiding its shadow. In the fish *Monocentris japonicus* it is utilized as a light to attract attention from potential mates. Other biological examples include slimemolds and honeybees. Slimemolds respond to local chemical concentrations to switch between amoeboid behaviour and slimemold form. When honeybees swarm the decision for which new hive site is selected is driven by the number of scouting bees promoting a particular new hive site rising above a threshold level (Seeley and Visscher, 2004; Seeley et al., 2006).

2.6.8 Micro-rules

Another well known local interaction mechanism are the stigmergic microrules various insects employ e.g. nest building in some wasp species (Camazine et al., 2001). Both bees and ants engage in behaviours to sort their growing broods and to clean their colonies (for instance corpse removal). These, it appears, are each based to simple stigmergic driven rules (Deneubourg et al., 1991). This is an iterated stochastic mechanism that describes a model where agents move in an environment with items that need to be sorted or clustered. An unladen agent will pick up an item with a probability

$$p_p = \left(\frac{k_1}{k_1 + f} \right)^2 \tag{2.1}$$

where f is the perceived fraction of items in the neighbourhood of the agent and k_1 is a threshold constant. A loaded agent deposits its item with a probability

$$p_d = \left(\frac{f}{k_2 + f} \right)^2 \tag{2.2}$$

where k_2 is another threshold constant.

Imagine agents in an environment with scattered items. The agents are tasked with collecting the items and placing them in groups. If the initial landscape is largely homogeneous then the stochasticity of the approach will ensure that some clustering occurs. Once there is some inhomogeneity there will be positive feedback to accelerate the agglomeration of items into groups. With suitably chosen parameters (in equations 2.1 and 2.2) a laden agent will tend to deposit its load in the vicinity of pre-existing clusters, growing it. Unladen agents will tend to ignore high density structures and only pick up solitary items.

Various clustering and sorting patterns are achieved with variants of this approach. Simple rules can thus yield interesting spatial structures. The approximately concentric

rings of pupa, pollen and nectar that are formed in honeycomb can be recreated with approaches like this (Bonabeau et al., 1999). A different model inspired by the growth of the *E. coli* bacteria represents the cells as simple tubes. These grow and divide periodically. This model has been shown to demonstrate fractal structures that mimic those of real colony development Rudge et al. (2013).

2.6.9 Foraging

A important activity for any colony is the location and exploitation of food sources. Earlier we saw that the pheromone laying of ants lead to exploitation of food in nature and the use of abstracted models to yield solutions to optimisation type problems. Honeybees also exploit food sources often over many kilometres of distance. These distances, and the impossibility of laying persistent chemical signals along bee flight-paths, mean that honeybees use a more complex agent to agent communications mechanism. Over a hundred years ago the writer and amateur naturalist Maurice Maeterlinck detailed an experiment (Maeterlinck, 1901). He placed pieces of honeycomb on a surface. When a bee found it, he marked the bee with a dot of paint. He tracked these bees back to their hive and trapped them on their subsequent exit from the hive. The intent was to show that the marked bees once inside the hive could communicate in some fashion with other bees the location of the food source. If the marked bees were trapped but other bees would fly directly to the honeycomb then, he reasoned, it would be clear that they had in some way received information describing the route to the food. Rather unsatisfyingly only a single bee did so. Maeterlink does not claim this as proof of bee to bee communication as it may have resulted from random finding of the food source. However, this indicates that Maeterlink was clearly thinking along the lines of bee to bee communications.

When a bee finds a food source it is able to assess its profitability to the colony. This is a combination of the distance and nutritiousness of the source. If the profitability is great enough then the foraging bee on return to the hive will advertise the location of the food source to the hive in an attempt to recruit more bees to the exploitation of the site. This is achieved using the famous waggle dance (Frisch, 1954; Seeley et al., 1995). The dance is carried out on a vertical area of the hive close to its entrance. This is known as the dance floor. Each dance consists of a short straight portion during which the bee waggles its body and flaps its wings making a buzzing noise. At the end of this portion the bee makes either a right or left turn and circles back to its

start position and repeats the waggle portion. The turns alternate between right and left. The dance encodes much information. The length of the waggle portion of the dance correlates with the distance from the hive to the food source. The angle that this portion makes to the vertical indicates the direction of the forage site relative to the sun's direction. The profitability of the site shows as the number of repeats of the dance carried out. Whilst the bee is dancing other bees crowd round it, they 'read' this information via the physical contact and/or auditory perception of the buzzing. The scent of the flowers at the forage site from the dancing bee is also sensed by the watching bees. Whether a bee dances or not is actually more complex and relates to the quality of food source found, and the current food level of the hive and the current nectar flow into the hive. Thus a bee from a good site may not dance if the hive is currently receiving a lot of nectar already. The threshold at which a dance occurs will adjust to the current state of the hive and its food gathering. Additionally there are always several foragers that will simply scout for new sources. This assures that the colony will continue to explore for new sources of nutrients. Seeley et al. (1991) details a colony wide model. This models the numbers foraging, dancing and being recruited as a series of differential equations. The growth and decay of bee numbers recruited to food sources with differing profitabilities is compared favourably to experimental results. This is a colony-wide model, rather than an agent based one. Erskine (2011) presents a simple stochastic agent based model that achieves a similar result. Virtual bees move on a lattice *dancefloor*. Returning foragers can recruit agents on adjacent squares to their advertised forage site. For sites with higher profitability the recruitment time is increased. In a real hive, at any given time there will be many bees dancing and advertising different forage sites. Potential recruits do not however survey the range of dances on offer in order to assess which of the many sites is the most profitable. Instead they will typically follow a dancer for a few of its dances before being recruited to that site. As the number of repetitions of dances is proportionate to a site's profitability the good sites will statistically attract a larger number of recruits, each of which may subsequently also advertise the site. Such mechanisms appear to provide effective balancing between exploitation of food sources and exploration for new sources.

2.6.10 Flocking

Many species demonstrate flocking behaviours. These arise from simple interactions between the individuals of the collective. One characteristic that appears is that the

swarm as a whole is capable of swift response to local threat. For example, a predatory dolphin approaching a shoal of sardines provokes a system wide avoidance behaviour in the whole shoal. Clearly the fish nearest the dolphin can see it and can turn to try to avoid it. Fish within the shoal may not see the predator, but respond to their neighbours direction changes. As these responses are swift the whole swarm appears to change its direction near instantaneously. This rapid system-wide response is reminiscent of the loss of length scales seen in critical systems. In such systems a minor perturbation can be result in a response whose scale can be of any size (limited only by the size as the whole system). We can ask whether at least some swarm behaviours benefit from the collective operating at some form of critical point. The swarm systems we explore in the following chapters are inspired by such flocking behaviours. Here we outline what sort of behaviours these are and provide background to other flocking-based work. We will return in the later chapters to provide greater detail relevant to the work presented.

There are a number of swarm algorithms that give rise to what may be described as flocking behaviours. Craig Reynolds developed "Boids" in 1986 (Reynolds, 1987). This was an artificial life program that simulated bird flocking motions. He proposed what could be described as a kinetic interaction system. A swarm of independent agents, each with continuous position and velocity states, would iteratively update their states based on perception of its neighbours. Each agent would calculate the mean position and velocity of its neighbours (defined within a certain range). It would then adjust its own position in the following iteration by moving towards the mean position of its neighbours (termed cohesion). It would similarly adjust its velocity toward the mean velocity of its neighbours (termed alignment). A third rule would push the agent away from any agent too close to it (termed avoidance).

Reynolds' flocking algorithm has been subject to numerous variations, adding in: assumed fear, or leadership roles, or desire to stay close to roost sites etc. It has been shown (Feder, 2007) that starlings consider the number of neighbours (not a neighbourhood radius) as important, and that the influence of neighbours was spatially anisotropic. Nearest neighbour interactions combined with an energy minimization argument has been used to generate line and Vee formation flocks (Klotsman and Tal, 2011). The Reynolds model assumes that the elements of a flock (birds say) are in some way self monitoring the distances and velocities of their neighbours and calculating their next step. Biologically it is not clear that this is plausible. However these algorithms have been exploited providing Tim Burton's Batman Returns film realistic bat flocks and wildebeest stampedes in Disney's Lion King (Wikipedia, 2014).

Convincing flocking models have a commercial value.

These homogeneous swarm algorithms have been further extended by combining multiple "species". This heterogeneous variant was initially proposed in a 2D environment (Sayama, 2009). Each species has separate parameter sets to control the scale of the cohesion, alignment and avoidance forces. Particles are also allowed to accelerate and decelerate within certain ranges, and will tend to return to their species' preferred speed. It has been further developed within 3D spaces (Sayama, 2012a), and has used evolutionary approaches to discover interesting heterogeneous swarms (Sayama, 2010, 2012b). The approach allowed colliding particles the possibility of exchanging one set of parameters for that of the other particle. The function used to control this exchange acted to favour one behaviour over another e.g. If the colliding particle's recipe is favoured over the collidee's, then predator like behaviours tended to emerge. Sayama has observed many differing behaviours: blobs, amoebas, oscillators, rotating swarms, jellyfish to name a few. Even though the particle interactions are purely local and kinetic, one can't help using biological metaphors when observing the swarm chemistry in action. It is interesting to wonder how far these superficialities can been extended to become models. It is, however, not easy to determine the correct low level interactions that will provide the desired emergent group behaviour. This difficulty in designing the desired emergent outcome is a central problem to swarm interaction researches, so this system offers a useful environment within which to explore these issues. Chapter 5 presents work in this area and will present more background to swarm chemistry and flocking.

The Self-Propelled Particle model (SPP) pre-dates the Reynolds algorithm and can be seen as a simpler version (Ball, 2011a; Vicsek et al., 1995). In the pure kinetic SPP the particles have a constant speed and on each update the direction of travel is updated under the influence of each particle's neighbours. They attempt to align themselves with mean velocity of their neighbours. As particle density rises group motions emerge. This raises the question of whether there may be simpler formulations of swarm chemistry that exclude attractive and repulsive components. Whilst self driven particles are uncommon in physics, they are typical in biological systems. The emergence of cooperative motion in this model has analogies with the appearance of spatial order in equilibrium systems. Dynamic variants of SPP which are derived from consideration of Newtonian forces via attractive and repulsive pairwise particle forces, and indeed other components (Levine et al., 2000; Newman and Sayama, 2008). SPP may provide biological insights. Newman and Sayama explore the effect of a

sensory blind spot on the emergent milling behaviour of the SPP model they used. As the size of the blind zone increases the behaviour disappears, perhaps suggesting a link between collective emergent behaviours and development of sensory apparatus in biology (Newman and Sayama, 2008). An SPP model has been shown to provide a model for cell sorting (Belmonte et al., 2008). The model is parameterised and two cell types are modelled using different parameters. Mixtures of the two cell types are shown to separate into inner and outer areas.

Artificial chemistries have many similarities to Swarm Chemistry. Again they are concerned with aggregations of individual elements interacting. Here the swarms are less abstracted, tending to have mass, volume and chemical interaction rules. Whilst many formulations may be expressed as sets of differential equations there are formulations that explicitly model the chemistries by mixing different 'swarms' of particles. Typically the interactions may be expressed as reactions rather than kinetic interactions, but interesting structures can occur in such formulations e.g. the semi-permeable membranes in (McMullin, 1997). The flip-side of this approach may be described as Chemical Robotics. Here, real cell-like chemicals are designed to ensure that their interactions cause desirable behaviours. In (Stepanek, 2010) one species of particle is designed to flow against the gradient of a second species, perhaps useful as a drug delivery mechanism. Regular dynamic motions can be designed e.g. the peristaltic polymer gel of (Maeda et al., 2009). The Los Alamos Bug is a project to develop a minimal protocell (Fellermann et al., 2007; Holmes, 2005) which exhibits metabolism, heredity and containment. These are the elements the authors propose are necessary for minimal life. The dynamics of the kinetic and structural interactions are noted as being an important problem to understand (Fellermann et al., 2007). Here a dissipative particle dynamic (DPD) model is adopted to explore this. Interactions are derived from the inter-particle forces which contain three components $F_{ij} = F^c_{ij} + F^d_{ij} + F^n$. These represent cohesive, dissipative and noise force components between particles i and j and thus bare comparison with swarm chemistry.

2.7 Particle swarm optimisation

PSO is another flocking inspired system and we examine this fully in chapters 3 and 4. Here we provide a suitable overview of PSO and the concepts of criticality that will be covered more fully in the forthcoming chapters. It is often used to solve optimisation type problems.

Solutions to optimization problems typically involve the selection of a best element (with regard to some criteria) from some set of available alternatives.

We are able to evaluate a potential solution $F(X_n)$ which evaluates the solution at position X_n in the problem space. If we have evaluated the problem at locations $X_0, X_1, \ldots, X_n$ i.e. we have values $F(X_0), F(X_1), \ldots, (X_n)$. We want to have a method by which we should select X_{n+1} so that $F(X_{n+1})$ is a better solution.

red A random search will afford a chance of locating optimal solutions in finite solution spaces if one is prepared to wait for a long time. Searching the problem space with some form of gradient descent method will ensure that we find a local optima, but not necessarily a global optimum. We describe the former case as one which is pure exploration, no knowledge of the problem space that we find from the sampling is utilized to guide our search. Gradient descent, in contrast, is using knowledge of the problem evaluated at each sample point. Indeed we could say that it is only exploiting this information as once it locates a local optima it will stagnate, unable to *jump out* and explore more widely. Metaheuristics attempt to balance these two modes of operation Clerc and Kennedy (2002).

A well known example of the sort of problems we discuss is known as the travelling salesman problem. This is a discrete form of this problem type: Given $\mathcal{N}$ towns, what is the shortest route that ensures all towns are visited exactly once? As the number of towns increases this problem quickly becomes infeasibly long to solve via brute force methods. A family of approaches known as metaheuristic algorithms can provide good solutions to such problems. PSO is one of these. As noted earlier the pheromone trail feedback of ant foraging can be used to solve this sort of discrete problem. Where problems are continuous the particle swarm optimisation algorithm is a metaheuristic that has enjoyed widespread use as a means to obtain solutions. Problems of this sort are often of a high dimensionality d. A cost function maps locations in the d dimensional problem space to a value indicative of how good or bad that solution is: $F : \mathbb{R}^d \to \mathbb{R}$. Initially PSO was proposed as a simple particle based system with limited intra-particle communication but in order to overcome limitations in this system greater inspirations from insects (bee and ant), birds (cuckoo), fish, and others have been used. They share a pattern where multiple agents explore a problem space. They share information about solutions each agent has found. The shared information is the used to modify future behaviour. Additional bio-mimetic parallels are often cited to justify additional mechanisms.

For a d dimensional problem, with a cost function defined such that $F : \mathbb{R}^d \to \mathbb{R}$,

PSO considers that each particle in its swarm represents a solution to a given problem. A PSO swarm consists of a N particles each with a position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$, where $i \in \{1, 2, \ldots, \mathrm{N}\}$. The position state is its position within the d dimensional problem space, and represents a potential solution. Its velocity determines where the particle will move next and is changed on each iteration of the algorithm in order (hopefully) to arrive at better solutions or to track a dynamic solution. As well as position and velocity each particle retains limited knowledge of previous locations within the problem space. Each particle remembers the best location it has visited, $\mathbf{p}_i$ and has knowledge of the best location experienced by the whole swarm, $\mathbf{g}$.

Simple rules are used to update the particle states (Kennedy and Eberhart, 1995)

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + \alpha_1 \mathbf{R}_1(\mathbf{p}_i - \mathbf{x}_i) + \alpha_2 \mathbf{R}_2(\mathbf{g} - \mathbf{x}_i) \tag{2.3}$$

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

The velocity update is a linear combination of three contributions: An inertial term parameterised by ω, a pull towards the personal best location parameterised by α_1, and a pull towards the global best location parameterised by α_2. This is visualised in Figure 2.1. Four red particles are shown on a problem landscape. Higher areas in the landscape represent higher cost function values. One red particle shows the three components of its update rule with three arrows. The red arrow shows its current velocity. A blue arrow shows the vector to its personal best location (shown as a blue point). A black vector points toward the global best location (shown as a black point). The update for this particle on the next iteration is the weighted sum of these vectors as defined in Equation 2.3. The symbols $\mathbf{R}_1$ and $\mathbf{R}_2$ denote diagonal matrices whose non-zero entries are uniformly distributed in the unit interval.

The history of PSO has explored many variants. Chapter 3 will cover the background in detail. Here we provide a brief outline of the sort of variants that have been explored.

Parameter selection is a major component in determining whether the algorithm finds good solutions. However other factors contribute to success. If a new global best location is found this is communicated to the rest of the swarm. Essentially the swarm is fully connected. Other topologies have been shown to produce improved results (at least for certain problems) (Bratton and Kennedy, 2007). Other improvements have been shown to be possible if more use is made of the knowledge gathered during the

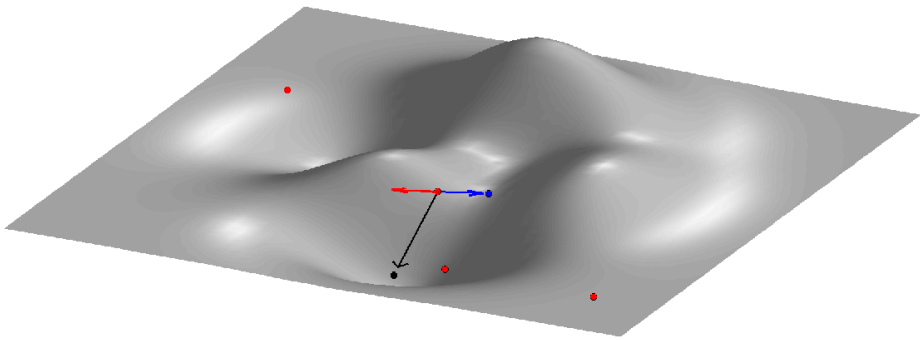


Figure 2.1: Particle Swarm Optimisation visualisation. The cost function is visualised here as a surface. The peaks represent poor solutions to the problem, the valleys are good solutions. The swarm particles, shown in red, measure the height of the surface at their current location. On each iteration the particles move to new locations. We show graphically the update rule being applied to a single particle (the one with the arrow vectors attached), but the process is applied to each particle. The red vector shows the particle's current velocity. The blue dot is this particle's current personal best location, so the blue line is the force attracting the particle back to this location. The black dot is the best location seen by the whole swarm. The black vector is this particle's attraction toward that point. A new velocity will be calculated by the vector sum of these three components. The components are scaled by the parameters and stochastic variables in the update rules in equation 2.3

swarm's sampling of the problem space. Although again improvements are often only available to specific problem types and lack generality. The attractive components in the update rules are driven by the idea that good solutions are likely to be located near other good solutions. Thus other mechanisms to drive local search of problem spaces have been employed.

Even with these studies and modifications PSO continues to suffer from the need to skillfully select parameter values for a given problem. If the swarm dynamics result in an exploding swarm then the particles leave the region of interest and require artificial constraints. If the swarm contracts too fast then premature convergence (stagnation) may result in poor solutions being found. Much activity is therefore given to detecting these conditions and taking steps to mediate their effects. Numerous bio-mimetic analogies are employed to allow swarms to jump out of stagnation or constrain their wilder divergences.

2.8 Criticality

Criticality is a term used in Physics to describe the properties of systems at a phase transition point (Binney et al 1992 cited by (Jensen, 1998)). Such systems demonstrate interesting behaviours when tuned to this *critical* point. These differ from the behaviour of the system not so tuned. Away from the critical point an external perturbation to the system causes local changes only. For example an ferromagnetic material will respond to a micro-magnetic perturbation with realignment of its magnetic dipoles nearest the location of the perturbation. Repeating the application of the external magnetic force will result in similar effects on each occasion. However, if the material's temperature is raised to its critical temperature, then a similar external drive to the system will cause macroscopic realignment of the bulk material's magnetised state. The probability of the size of this realignment follows a power law distribution. In this state repeated external perturbations can therefore result in responses at all scales.

It has been noted that many systems appear to behave in a similar manner but without the need for tuning. These kind of heavy-tail distributions of event sizes are characteristic of systems as disparate as Earthquake magnitudes, word usage frequencies in languages, neuron spike avalanches and the changing water level in the Nile. In some way these systems must self-organise to their critical point. Self-organisation refers to non-equilibrium systems that develop structures and patterns in the absence of control or manipulations by external agents (Nicolis 1989 cited by Jensen (1998)).

Self-organized criticality (SOC) can be thought of a a property of dynamical systems that have a critical point as an attractor. At this critical point macroscopic properties of the system display the spatial and/or temporal scale-invariance characteristic of the critical point of a phase transition, but without the need to tune control parameters to precise values. The sandpile model (described below) (Bak et al., 1988), and other similar models, have been used to address the frequent appearance of 1/f laws in natures. This model both self-organises and shows power laws in its avalanche size and duration distributions.

2.8.1 Sandpile model

The sandpile model is a cellular automata that shows critical behaviour. It provides a simple to describe system that may be used to discuss the features of criticality.

The model consists of a rectangular lattice of cells. Onto this grains of sand are dropped at random. Initially the system is empty, so a grain lands and sits at rest. As more sand is added the pile rises, at some locations the local gradient increases to the point that the pile becomes unstable and a slip occurs: Grains from this unstable locale redistribute to the adjacent cells. If a grain moved off the edge of the lattice it is removed from the system. Still later, we find that many locations are now full. A new grain, not only slips, but in so doing may cause its neighbouring cells to slip. We see an avalanche. The avalanche ends when all cells in the system are once again just below the threshold for local collapse. As this is an absorbing state of the system, the ensemble naturally falls into this state, thus its dynamics appear self-organising. We can think of this as a metastable state: the system is static but the piled grains are straining against each other at the point of collapse. The system is not at a minimal energy state. The system maintains itself in this state and shows a distribution of avalanche sizes that follow a power law.

2.8.2 Critical systems: characteristics and features

The sandpile model, and other critical systems, share a number of features.

They are built from many components that interact locally. The interactions consist of force or information exchange. The interactions take place internally (thus we may not see the detail of what is occurring). When externally driven in some manner the macroscopic behaviour of certain system events will be power law distributed. System events may occur at all size and time scales. Practical considerations mean that such distributions may be limited. In a sandpile no avalanche may be smaller than a single grain. System size will tend to impose an upper limit on event sizes.

Such systems exhibit a separation of timescales. A slow external driving perturbation (e.g. magnetic in the ferromagnetic material example, the sand grain drip in the sandpile model), contrasts with the fast relaxation of the system. Such systems often operate in this manner as there are thresholds in the system. We can imagine the build of stresses between tectonic plates rising until some threshold is breeched, releasing an earthquake.

2.8.3 Criticality in animal swarms

We propose that criticality in swarm systems is a mechanism which drives the creation of useful patterns and behaviours in a swarm. As much inspiration derives from nature

we might expect that biological observations would support this hypothesis. Empirical evidence seems scant, perhaps due to the difficulty in recording and analysing the movements of large numbers of individuals in a swarm.

Swarms (in particular birds and fish) often appear to react to perturbations within their environment *as if of one mind*. The approach of a predator causes a flock of birds to wheel and swerve to avoid the attack. An empirical study estimated the 3D position and velocity states of the individual birds in starling flocks (Cavagna et al., 2010) from video footage. Fluctuations in the individual bird velocities are shown to correlate over distances far greater than the birds' range of perception. This correlation length increases linearly with flock size. Thus the swarm appears to demonstrate a scale-free characteristic, at least in its flocking behaviour. This differs from bacteria swarms where correlation length appears to decay exponentially with distance. Similar results have been shown from the analysis of the 3D trajectories of swarms of several hundred midges (Attanasi et al., 2014). Whilst tracking only for ten seconds it was apparent that velocity fluctuations in the swarm were correlated over lengths much greater than the assumed communication distances between midges. Their findings actually suggest the swarm behaves in a slightly sub-critical manner. A flock, acting criticality, allows the swarm to balance responses to the world. Too ordered and it can be hard to adapt or respond to environmental changes. However, if the swarm is too disordered then defensive responses to predators may not be strong enough (Chaté and Muñoz, 2014). There may be other behavioural pairs that benefit from some balanced position to achieve optimality.

It is exciting to think that ideas from the the physics of phase transitions could be applied to biological structures, perhaps providing an organizing principle to swarm behaviours.

2.8.4 On the potential for criticality in our swarm systems

Abstract flocking systems are not real swarms. *Boids* respond to all neighbours within a given radius. Real birds cannot see backwards, so will not be influenced by all their neighbours. Nevertheless similar principles may be explored by examining such simplified models.

It seems reasonable to believe that critical points may occur with artificial flocking systems as they share many of the desired characteristics. We look at both PSO and swarm chemistry. Each consist of swarms of between 25 and 1000 or so particles. The

particles interact using local information. The systems are parameterised, so if they do not self-organise to an absorbing state that is also critical we have the possibility of tuning them. In PSO the rate of new discoveries of global best locations will depend upon the problem landscape. Typically, at least for complex functions, this rate tends to slow as the algorithm runs. Late on new discoveries occur rarely, and may act as a slow time scale external perturbation. Internally this information is shared between all particles. The particle dynamics rapidly adjust to this modified *force*.

As a result of the modest size of the swarms, the power law distribution of event sizes are typically truncated both at small sizes (due to miminum response) and large sizes (due to small system size). This may impact the ability to show power law event distributions concomitant with criticality.

In nature it is reasonable to think the some swarm behaviours should be critical. Being critical is a way for the system to be always ready to optimally respond to an external perturbation, such as a predator attack as in the case of flocks. It allows the collective to balance two requirements in a flexible manner. Criticality is often suggested as means by which such a balance may be obtained optimally. It is the properties of optimality and the balancing of a system at a point poised between two behaviours that appeals to us in this thesis. In the following chapters we will further discuss the range of PSO variants, comparator metaheuristics, criticality and criticality as applied to PSO.

Chapter 3

Criticality in particle swarm optimisation as a random dynamical system

3.1 Introduction

Particle swarm optimization and its variants enjoy widespread application in locating optimal or near optimal solutions to complex problems. However the quality of solutions found depend upon our ability to overcome the algorithm's drawbacks: the selection of suitable parameter values and the avoidance of the swarm's premature convergence. For a given set of problem functions one can empirically determine the *correct* parameter values or the best restart strategy, yet these solutions may fail to be optimal if presented with a different arbitrary function to solve. Here, the quasi-linear swarm dynamics that arise from consideration of the random components of the algorithm's dynamic matrix are considered. The relationship between the algorithm's parameters at the edge between convergent and divergent swarm behaviours is derived. The algorithm can be seen experimentally to operate at its best towards this margin.

In nature one observes many swarms: herds of mammals, schools of fish, flocks of birds etc. They often demonstrate near simultaneous movements across their populations such as sudden coordinated direction changes in the response to an approaching threat. Those individuals closest to an oncoming predator adjust their heading to avoid the encounter. These changed directions are communicated through the swarm nearly instantly. Such a loss of length scales is reminiscent of state changes in physical systems. Ferromagnetic materials if held at a specific temperature

may show large responses to small local magnetic perturbations. Such behaviour is described as being critical or showing criticality in its response. The behaviour of the Particle swarm optimisation (PSO) algorithm shows analogous behaviour.

Knowledge and control of the dynamics of a PSO swarm is shown to be essential to the use of this metaheuristic in solving optimisation problems. Whilst desired execution duration and the nature of the problem space introduce constraints into the process of locating solutions, it is from consideration of the properties of the product of the stochastic matrices of PSO's update rules that the dynamic capabilities of the algorithm may be determined.

First the necessary background to PSO and criticality is explored in particular those instances where the two have been previously combined. Next some previous attempts to explore the theoretical dynamics of PSO are discussed. Whilst these offer much insight into the behaviours that may be expected from the algorithm they have typically ignored the multiplicative role of the random element of the update rules. These have often been replace by their expectation values or viewed as a *drunken walk* about the predicted dynamics of the system, enhancing the local search capabilities. It is shown that they contribute to the dynamics of the swarm's capabilities in a manner that alters its expected behaviour.

In section 3.2 the products of random matrices are considered, in particular those that represent the PSO algorithm. The locus of α and ω values that give rise to swarms that will neither diverge nor collapse over time is derived. This curve matches the observed behaviour of an unconstrained PSO swarm i.e. one where neither positional or velocity constraints are imposed on the dynamic of the swarm.

3.1.1 Particle swarm optimisation

PSO is a metaheuristic method for obtaining solutions to search or optimisation type problems. The aim is to locate the best solution to problems that may have high dimensionality or parameterisation. The *best* solution is at a global minimum (or depending upon problem construction a global maximum). How best to find the solution? If time is no object then a random search will continue to locate improving solutions. This is usually not the case! For simple problem manifolds e.g. a sphere function, gradient descent may be all that is required to find good solutions. However, real world problems are rarely as conducive to simple methods: they may exhibit discontinuities or other potentially deceptive features.

PSO has enjoyed much attention and widespread usage. Around 700 PSO papers are grouped into 26 discernible application areas in (Poli, 2008). Google Scholar reveals over 150,000 results for "Particle Swarm Optimization" in total and 24,000 for the year 2014. It was inspired by the cooperative behaviours of flocks of birds or schools of fish to locate a problem's solution (Kennedy and Eberhart, 1995). The swarm is constructed from a number of particles, each of which occupies a position within the problem space being evaluated. The problem is evaluated at each particle location: essentially sampling the problem space. Additionally particles have velocity states. New locations are computed by applying the particle velocities as an update to their positions. Their velocities are then updated based on a sum of an inertia term and terms that represent the sharing of local and global knowledge that the particles have gleaned about the the problem space derived from their sampling procedure. Their future dynamics are thus modified, in part, by the knowledge of the problem surface that they are uncovering.

The position and velocity of particles, after random initialisation, are iteratively updated using the following linear dynamic of the following form.

$$\begin{aligned} \mathbf{v}_{i,t+1} &= \omega \mathbf{v}_{i,t} + \alpha_1 \mathbf{R}_1 (\mathbf{p}_i - \mathbf{x}_{i,t}) + \alpha_2 \mathbf{R}_2 (\mathbf{g} - \mathbf{x}_{i,t}) \\ \mathbf{x}_{i,t+1} &= \mathbf{x}_{i,t} + \mathbf{v}_{i,t+1} \end{aligned} \tag{3.1}$$

Here $\mathbf{x}_{i,t}$ and $\mathbf{v}_{i,t}$, $i = 1, \ldots, N$, $t = 0, 1, 2, \ldots$, represent, respectively, the d-dimensional position and velocity vectors of the i-th particle in the swarm at time t. The velocity update contains an inertial term parameterised by ω and includes attractive forces towards the personal best location $\mathbf{p}_i$ and towards the global best location $\mathbf{g}$, which are parameterised by α_1 and and α_2, respectively. The symbols $\mathbf{R}_1$ and $\mathbf{R}_2$ denote diagonal matrices whose non-zero entries are uniformly distributed in the unit interval[1] . The number of particles N is quite low in most applications, usually amounting to a few dozens.

In order to function as an optimizer, problems are cast as a non-negative cost function $F : \mathbb{R}^d \to \mathbb{R}$. $F(\mathbf{x}) = 0$ defines $\mathbf{x}$ as an optimal solution of the problem. In many problems, where PSO is applied, states with near-zero costs can also be considered as good solutions. Indeed for a given problem function one can

[1]This implements a component-wise multiplication of the difference vectors with a random vector and is known to introduce a bias into the algorithm as particles prefer to stay near the axes of the problem space (Janson and Middendorf, 2007; Spears et al., 2010). We have retained this form of stochasticity in our algorithm for the purpose of comparability.

compare the success of different algorithms by considering which obtains the smallest value solution for a given criteria such as the number of function evaluations or computational cost etc. The cost function is evaluated at the state of each particle for each time step. If $F(\mathbf{x}_{i,t})$ is better than $F(\mathbf{p}_i)$, then the personal best $\mathbf{p}_i$ is replaced by $\mathbf{x}_{i,t}$. Similarly, if one of the particles arrives at a state with a cost less than $F(\mathbf{g})$, the $\mathbf{g}$ term is replaced in all particles by the position of the particle that has discovered the new solution. If its velocity is non-zero, a particle will depart from the current best location, but it may still have a chance to return guided by the force terms in the dynamics.

The history of PSO has explored many variants. Instead of this fully connected swarm, different connectivities have been shown to yield improved performances (Bratton and Kennedy, 2007; Kennedy, 1998). However in all cases the success of the algorithm for a given problem will often require careful selection of the parameters ω, α_1, and α_2. Poorly chosen parameters can result in the stagnation of the algorithm in a suboptimal location. Here we are using a simple PSO with its original fully connected topology.

3.1.1.1 PSO Parameter Selection and Resultant Behaviours

In order to perform optimally, the algorithm has to ensure an appropriate mix of local search, i.e. the exploitation of existing knowledge of the problem space, and exploration of areas not yet adequately sampled. The optimal balance of exploration and exploitation will depend on the nature of the problem space and to some extent how long the algorithm is allowed to run for. In PSO this mix is achieved via the parameterisation of the algorithm. PSO is often applied to problems with little formal specification and trial-and-error search remains the only general option for parameter tuning. Small ω and large α_1 and α_2 values (within limits) will encourage the particles to converge toward the currently known good locations. Overdone, the swarm may prematurely converge to a local minima and achieve no further improvements. This is termed stagnation and is often deemed a failing of the algorithm. However this is not always the case. If the algorithm is given a time limit then collapse to a local minima at the end of this time may be the best action it could take. For example, (Worasucheep, 2008) utilises both failure to locate better solutions and loss of mean particle velocity to calculate a trigger signal. Once the signal crosses a threshold a diversity increasing procedure is initiated for all except the best particle. This is successfully applied to a typical PSO algorithm. Sometimes the detection of stagnation can be used to trigger

a change in the algorithm either by simply restarting the particles in freshly initialised locations or used in a hybrid algorithm to switch behaviour to one more attuned to exploitation.

Typically, PSO variants are tested against a number of test functions which have known optima. A number of criteria for success are measured (including average error, average number of iterations of obtain an optimum solution and minimum error). Other approaches have looked to use information from all particles in the update mechanisms. A comprehensive learning PSO (Liang et al., 2006) stochastically selects from a particle's best location or another random particle's best location on a dimension by dimension basis. All particles can thus influence the evolution of the swarm. They claim to be good on multimodal functions but poor on unimodal ones. This highlights the difficulty in finding a single strategy that succeeds against all criteria.

The algorithm's parameters play a major role in the dynamics of the swarm. One set of parameters may lead to a collapse in swarm diversity, another set to an exploding swarm expanding beyond the confines of the defined problem space. Often this can be controlled for. Particle velocities may be simply truncated to limit the motion of the particles and reduce the likelihood of divergence (Kennedy, 1998). Similarly the particle position states outside the problem space may be repositioned on the boundary. Velocity multipliers, derived from the algorithm's parameters, provide another means to constrict velocities (Clerc and Kennedy, 2002). The mathematical derivation of this constriction factor essentially limits the parameter values so that divergence is avoided. Other parameter controls may be applied for instance via linear decreases in the ω value as the algorithm progresses (Shi and Eberhart, 1999). Larger ω values tend to encourage greater exploration by the swarm of the problem space. By reducing this parameter's value the swarm is guided to spend more time exploiting the locations that seem promising. Other approaches to varying PSO's parameters have been explored including, random, increasing, decreasing and chaotic (Bansal et al., 2011). These mechanisms will generally restrict a diverging swarm. However this risks encouraging premature convergence. Other mechanism have thus been used to avoid stagnation: by using particle repulsion (Chowdhury et al., 2013; Riget and Vesterstrøm, 2002); or random velocities (García-Villoria and Pastor, 2009). Other means to affect the balance between exploratory and exploitative behaviours have been derived from other biological inspirations. For instance animals such as bat (Yang, 2010), cuckoo (Yang and Deb, 2009) and fish (Wang et al., 2005) using respectively, echolocation, brood parasitism or leaping motion as inspirations. The cuckoo search algorithm also

co-opts an earlier idea of using Lévy flights to drive local searches. Lévy flights are random walks where the step-lengths are drawn from a heavy tail distribution. This is suggestive of potential benefits arising from adding power-law distributed movements to PSO.

Rather than modifying the algorithm's parameters, one can alter how the search near known good locations is conducted. Indeed the standard PSO algorithm adds stochastic noise to the linear sum of the various knowledge terms. This has the effect of allowing the particles to have a somewhat randomised search that is drawn toward the known good locations. However, this has the tendency to focus searches along the axes of the search space. A chaotic search was explored (Liu et al., 2005). Here they combined a parameter update strategy with a chaotic local search mechanism. Particles with good results had their ω parameter linearly reduced whereas particles with poor results had their ω values increased to a maximum value. On its own this assured a mix of diverse behaviours. The logistic function, $x_{n+1} = \mu x_n(1 - x_n)$, was employed to create a chaotic search pattern about the location of the best performing particle. Results compare favourably with many PSO variants, but it is not clear whether both of these components are needed to achieve the results reported.

3.1.1.2 Alternative Metaheuristics

PSO is not the only metaheuristic to provide solutions to optimisation problems. Algorithms may use an adaptive probability model to choose from a number of strategies in a success-dependent fashion. Self-adaptive algorithms have been proposed for differential evolution (Qin and Suganthan, 2005) and also for PSO (Wang et al., 2011; Zhan et al., 2009). These hyperheuristic methods measure the behaviour of the swarm and use this to determine which algorithm is most appropriate to run at that particular time. A stagnating swarm may be switched to follow a ruleset that favours increasing diversity, or a divergent swarm may be changed to encourage future convergence. The TRIBES mechanism removes the velocity update component of PSO and the inherent need to set parameter values (Clerc, 2010). Multiple sub-swarms grow and shrink (removing the need to set a swarm size) depending on performance. The swarms exchange information on good locations and new positions are created by weighted sums of the positions of particles representing good solutions. Recent successful approaches have been derived from the Covariance Matrix Analysis Evolutionary Strategy (CMA-ES) (Hansen and Ostermeier, 2001). This technique has performed well in recent benchmark tests, capturing top spots (Liao and Stutzle, 2013;

Loshchilov, 2013). This is a (μ, λ) evolutionary strategy. A multivariate normal distribution derived from the μ best parents are used to generate the next generation of solutions (λ in number). After each iteration the distribution's covariance matrix is adapted to direct the future selection toward new and better solutions. The technique may still suffer from premature convergence. However, detection of such stagnation may be used as a signal to trigger algorithmic restarts or other procedures to increase the diversity of the swarm to obtain good results (Loshchilov, 2013).

3.1.1.3 The Problem of Making Comparisons Between Algorithms

PSO is therefore only one of many potential algorithms proposed to solve a given problem. To differentiate between them one needs to find a means to make comparisons: to determine which is better. Of course better can mean different things: best solution; best solution after N iterations; least computationally intensive etc. Competition formats have proved popular method to establish the capabilities of multiple algorithms. These typically consist of a suite of test functions that provide a broad range of challenges, be it high dimensional spaces, multi-modalities, deceptive landscape features etc. Competitors are expected to locate the best locations within each function given a given budget of function calls. Statistical comparisons can then be computed to provide a ranking of solution finding capabilities or of computation load. One weakness of the competition format maybe that the winning algorithms, are by definition, good at the functions included, but that other function suites may yield different results. It has been shown using genetic programming techniques that objective functions may be evolved that favour different algorithms (Langdon and Poli, 2007). Here PSO, differential evolution, CMA-ES, and simple hill climbing algorithms were used. In each pairing of algorithms the authors were able to find problems that favoured either of the algorithms. This does not negate the value of the competition format, but does mean that it is necessary to show that the functions used in the format map to range of real world problems, although intellectually wider function types retain an interest.

In this chapter much of our empirical results are obtained using the 28 functions defined for the 2013 International Conference on Evolutionary Computation (CEC2013).

An interesting approach, certainly augmenting knowledge of the algorithm's dynamics, is to seek to use the sampling of the problem space to characterise the nature of the function being explored. This can be used to modify the future operation

of any algorithm. Geometries of problem surfaces have been examined in this context (Mersmann et al., 2011). Whereas Malan and Engelbrecht (2014) utilises information-like measures to capture problem space features that can similarly be used to guide selection of appropriate algorithms.

3.1.2 On criticality

Criticality in equilibrium thermodynamics is used to refer to the properties of a system at a transition point between phases. At this point small perturbations can propagate throughout the whole system (Jensen, 1998). Ferromagnetic materials, for example, gain their magnetism ultimately via the alignment of magnetic dipole moments arising from electron spin states. The state of any individual dipole may be influenced by the magnetic field it finds itself in. If this material is at a high temperature, then the thermal noise will be greater than the influence of small perturbations and no effect is seen. At low temperatures the dipole moments are effectively frozen in whichever state they find themselves in. Small external magnetic perturbations will have little or no effect in either case. When the system is tuned by heating it to point between these two states, then any small external magnetic perturbations are able to flip the state of nearby dipoles. Further, these changes can propagate throughout the material resulting in a chain of dipole flips. A characteristic of the such systems is that changes in system states can exhibit power law distributions. The small external magnetic perturbation will frequently cause small localised shifts in magnetic states and domain structures in the material, but can also result in occasional large shifts in the magnetic state of the material. In order to show this type of behaviour, this system needs to be very finely tuned to a critical temperature.

More widely the term criticality may refer to any dynamical system which behaves in a manner like this (Bak et al., 1988). Bak notes that there are many systems in nature that exhibit similar power law type distributions. We see such heavy tail distributions in the earthquakes described by Gutenburg-Richter law. Similarly power laws crop up in: Zipf's law in ranked distributions of word usage in English; fractal geometries in nature; and 1/f type noise in physical systems. The ferromagnetic example above requires tuning but it seems infeasible for all the occurrences of power-laws to require such fine tuning. It is proposed that systems consisting of many interacting units may evolve automatically into a critical state: effectively self-organising to a point poised between order and chaos (Bak, 1997). No outside agent is required to tune the

system. If perturbed such a system will respond with events that follow a power law distribution, but will result in the system returning to its critical state.

One means to explore the properties of such systems may be explored by looking at simple models such as the proposed sandpile model (Bak et al., 1988). This numerical model imagines a square lattice grid: at each locale sand grains may be deposited. As the sand pile grows due to the slow deposition of grains, the gradient between grid locations can grow: each grain added adds +1 to the local gradient. If the gradient and any location increases beyond some specified threshold (say 3) then the grains at that location collapse. Four grains are redistributed to the adjacent squares and the location of collapse in reduced by the corresponding amount (-4). Early in the sandpile's evolution the collapses are minor, however as time passes and grains continue to be added, more and more locations increase toward the point of collapse. Now a single added grain may lead to a collapse and as the grains are reallocated to neighbours, those adjacent cells are found to also lead to collapse. The avalanche triggered by a single grain may be small or large. Avalanche sizes in this model are shown to follow a power law distribution. Similar dynamics have been noted earthquake magnitudes (Olami et al., 1992), punctuated evolution (Bak and Sneppen, 1993), neuronal avalanches (Beggs and Plenz, 2003; Eurich et al., 2002; Levina et al., 2007), and even rainfall (Deluca et al., 2015). The presence of criticality in a system can be shown to optimise the system in some way. Shew et al. (2011) showed the criticality of cortical neuronal avalanches results in optimal information capacity and transmission. The functional benefits of critical dynamics for information processing in the brain have been discussed, see the overview by Shew and Plenz (2013). These studies motivate the present study and whether criticality can also be beneficial for PSO's optimisation and search capabilities.

In such systems, the presence of power laws in system event sizes imply that events of any size are possible. Practical limits arise from the finite size of systems. If we were to engineer critical dynamics into the swarm size of a PSO algorithm then we would be assured that the swarm would sooner or later extend throughout the whole problem space and thus avoid stagnation. A random search strategy would likewise avoid stagnation, but our approach assures that the algorithm can still favour exploitation of the problem space over exploration.

Previously there have been approaches to adding criticality to PSO. A sandpile-like approach was employed by adding an additional counter to each particle (Lovbjerg and Krink, 2002). If a particle came close to another it incremented its counter. Once

over a threshold the particle relocated within the problem space and redistributed its accrued value to other particles allowing *avalanches* to occur. It is somehow elegant that feedback from the swarm's own behaviour is used to modify the future behaviour of the swarm. There was limited evidence that the swarm behaved in a critical manner and the problem of setting parameter values remained. Richer and Blackwell (2006) implemented a PSO algorithm inspired by the Lévy flight random walk behaviour of many foraging animals. They modified a Gaussian PSO algorithm, which performs velocity updates by drawing from Gaussian distributions scaled by distance of the particle from local and global best locations, to use Lévy distributions instead. The nature of this power-law approach produced a greater number of outliers in a given problem space, resulting in a more powerfully exploring swarm. Their results showed that this Lévy swarm outperformed both standard PSO and Gaussian PSO approaches. The Bak-Sneppen algorithm (Bak and Sneppen, 1993) provides a Self-organized criticality (SOC) model of punctuated evolution seen historically in the number of species in the fossil record. This has been used to create a criticality augmented PSO variant (Fernandes et al., 2012). The generated power-law distributed random numbers are used directly to modify the velocity update parameters in a statistically balanced way. Additionally, local search about the particle's position was driven by random numbers from the same distribution. This power-law distribution may be created off-line adding little or no overhead to the algorithm. Favourable results are demonstrated in comparison to standard PSO.

3.1.3 Our approach

As it is known that parameter selection can make a large difference to the performance of the PSO algorithm a great deal of effort has been expended on locating the *best* values for a given problem. As we have discussed speed of solution convergence or the need to deal with different problems often requires that we search anew for values to use for new challenges. Previous studies have examined the dynamics that arise from consideration of the update rules.

An early exploration of the PSO dynamics (Kennedy, 1998) considered a single particle in a one-dimension space where the personal and global best locations were taken to be the same. The random vectors in equation 3.1 were replaced by their averages such that apart from random initialization the algorithm was deterministic. As each dimension is updated separately the system is cast as essentially one dimensional.

Also as particle interactions only arise from the communication of updated global best values one can consider the case where the global best is genuinely the optimum result. No updates occur and the dynamics of a single particle can be studied alone. Varying the parameters was shown to result in a range of periodic motions and divergent behaviour for the case of $\alpha_1 + \alpha_2 \geq 4$. The addition of the random vectors was seen as beneficial as it adds noise to the deterministic search.

Control of velocity, not requiring the enforcement of an arbitrary maximum value as in (Kennedy, 1998), is derived in an analytical manner by (Clerc and Kennedy, 2002). Here eigenvalues derived from the dynamic matrix of a simplified version of the PSO algorithm are used to imply various search behaviours. Thus, again the $\alpha_1 + \alpha_2 \geq 4$ case is expected to diverge. For $\alpha_1 + \alpha_2 < 4$ various cyclic and quasi-cyclic motions are shown to exist for a non-random version of the algorithm.

In (Trelea, 2003) the analysis again considered the dynamics of a single particle in a one dimensional problem space (unless a new global best location is found the particles essentially act independently). A deterministic version of PSO, setting $\mathbf{R}_1 = \mathbf{R}_2 = 0.5$ was used for this analysis. The eigenvalues of the system were determined as functions of ω and a combined $\alpha = \alpha_1 + \alpha_2$ (and $\alpha_1 = \alpha_2$). This led to the derivation of three conditions: When $\omega < 1$, $\alpha > 0$ and $2\omega - \alpha + 2 > 0$ the particle would converge. Harmonic oscillations occur for parameter conditions $\omega^2 + \alpha^2 - 2\omega\alpha - 2\omega - 2\alpha + 1 < 0$. Zigzag dynamics are expected when $\omega < 0$ and $\omega - \alpha + 1 < 0$. The regions that these conditions define in the $\omega : \alpha$ parameter space overlap. Where parameter values lie in more than one region we will see behaviours that combine these differing dynamics. For example oscillatory motions whose amplitudes decrease as the particle converges to a solution etc. As with the preceding papers the discussion of the random numbers in the algorithm views them purely as enhancing the search capabilities by adding a *drunken walk* to the particle motions. Their replacement by expectation values was thus believed to simplify the analysis with no loss of generality.

Empirical evidence for the role of parameter selection in the capability of the PSO algorithm to location solutions can be carried out. Whilst the algorithm is shaped by three parameters it is often the case that α_1 and α_2 are assigned the same values. Thus we can explore the parameter space by considering different pairs of ω and α. Clearly if the ω value is greater than unity each particle's velocity will tend to increase on each update. The twin attractions of personal and global best locations will tend to be overcome by the inertial term. The swarm will expand outwards. We can also consider negative ω values. Again values less than negative one will also result in

diverging swarms. Typical implementations of PSO usually set α values between 0 and 4. Larger or negative values are found to yield divergent swarms. For values within these broad regions it is not apparent which values yield the best results. Locating good parameter pairs for specific problems often involves a degree of trial and error. We can explore PSO capabilities by repeatedly executing a simple variant of the algorithm for each pairing of parameters in the $\alpha : \omega$ space. With many repetitions and a suitably large iterations budget for each run, results can be averaged to even out variations in performance seen in individual executions. Fig 3.1 shows the cost function landscape for a typical problem function. The xy-plane represents the position in parameter space and the vertical position shows the average value solution for the algorithm. For positive ω values between zero and one there appears to be a broadly banana-curve shaped portion of the parameter space that results in the best performances. For negative ω values the best pairings appear to have a linear relationship. Parameter pairings that perform well occupy the regions in the valley of this surface plot. This pattern consistently emerges over all functions. With different objective functions the speed with which this picture emerges is variable. Not all locations in the valley are equally good, nor is the variation between the foot of the valley and locations up its slopes equivalent across all functions.

All responses are shown in Appendix A.

The analyses above do not appear to suggest that this curved relationship (for positive ω) should exist between parameters.

Rather than make deterministic simplifications as discussed above, Jiang et al. (2007) considering the expectation values and variances of PSO's stochastic update matrix. Rewriting the update rules in a recursive form, they calculated the eigenvalues of the system for different parameter pairings, arguing that swarm stability exists where the largest eigenvalue of the system is less than or equal to one. However this last assumption is wrong. It is possible for sequences of updates whose eigenvalues are greater than one to result in a stable system. We have to consider not just the size of the eigenvalue, but the direction of the eigenvector at each update. The curve they predict marks the locus of $\omega : \alpha$ pairs they believed guaranteed swarm convergence i.e. parameter values between this curve and the origin will result in convergence. Similar approaches yielded matching curves (Poli, 2009) and utilising a weaker stagnation assumption (Liu, 2014). We will refer to this parameter locus as the Jiang curve. This curve is displayed for comparison in Figure 3.5.

Experimentally one can run PSO against a variety of functions and explore the

convergence or otherwise of the swarms. This can then be compared with these theoretical approaches. In Poli (2009) the comparison is made with very short runs (30 iterations). Their intention being to perform a huge number of repeats in order to confirm the statistics of the expectation and variance beliefs underlying their approach. The algorithm was initialised already in a state of stagnation in order to confirm that convergence occurs as predicted. In Liu (2014) iteration budgets were also limited: their protocol uses 3100 function evaluations per run (about 155 iterations). They studied a range of parameter pairings both inside and outside the Jiang curve. Whilst those within the curve do converge, even those outside occasionally achieved good results. It is shown in this thesis that this is due to the region of convergence extending beyond the Jiang curve. As one approaches the locus of parameter values proposed in this thesis one needs many more iterations to ensure convergence. Low numbers of function evaluations will likely result in poor results. In Cleghorn and Engelbrecht (2014b) PSO is run with experimental budgets of 2000 iterations (and 64 particles). Good agreement with the Jiang curve is shown. Their problem spaces are 50 dimensional making improved solutions even harder to find. They constrict diverging swarms which will impact the dynamics. With much greater iteration budgets initially diverging swarms may eventually return to a converging state. This thesis shows that with iteration budgets well beyond 2000 the best performing parameter pairs often lie beyond the Jiang curve.

The Jiang curve does not mark the division between convergent and divergent swarms. The iterated use of the random factors $\mathbf{R}_1$ and $\mathbf{R}_2$ adds a further level of complexity to the dynamics of the swarm which affects the behaviour of the algorithm in a non-trivial way. Empirical results are shown later that demonstrate the existence of best parameter values outside the Jiang curve. For this purpose we will formulate the PSO algorithm as a random dynamical system and present an analytical solution for the swarm dynamics in a simplified but representative case (Sect. 3.2) and compare the theoretical predictions to results of simulation for a number of benchmark functions (Sect. 3.3). Finally, in Sect. 3.4, we discuss the relation of our model to the original PSO algorithm.

We present an analysis of the quasi-linear swarm dynamics that arise from consideration of the iterated contributions of the random components of the algorithm's dynamic matrix. Our study shows that a region of the ω:α parameter plane results in a curve dividing convergent and divergent behaviours i.e. the location where a PSO swarm is poised between infinite exploration and collapse to stagnation. The

form of the relationship between the parameters that generate this curve differs from those suggested in previous analyses. Using an unconstrained swarm we show that empirically the algorithm performs best when approaching this margin.

Deviations from stability represent the ability of the swarm to sample the problem space. Such deviations are possible when poised on the curve, yet on average the swarm will return from such deviations. Thus the swarm avoids the two extremes of behaviour: convergence (and its stagnation) and divergence (and the inability to locate good solutions). Near this curve the best performance of PSO is empirically confirmed (for positive ω) to follow a curved relationship between ω and $\alpha = \alpha_1 + \alpha_2$, with $\omega \approx 1$ at small α, and $\alpha \gtrapprox 4$ at small ω. Large values of both α and ω are found to cause the particles to diverge leading to results far from optimality, while at small values for both parameters the particles converge to a nearby solution which sometimes is acceptable.

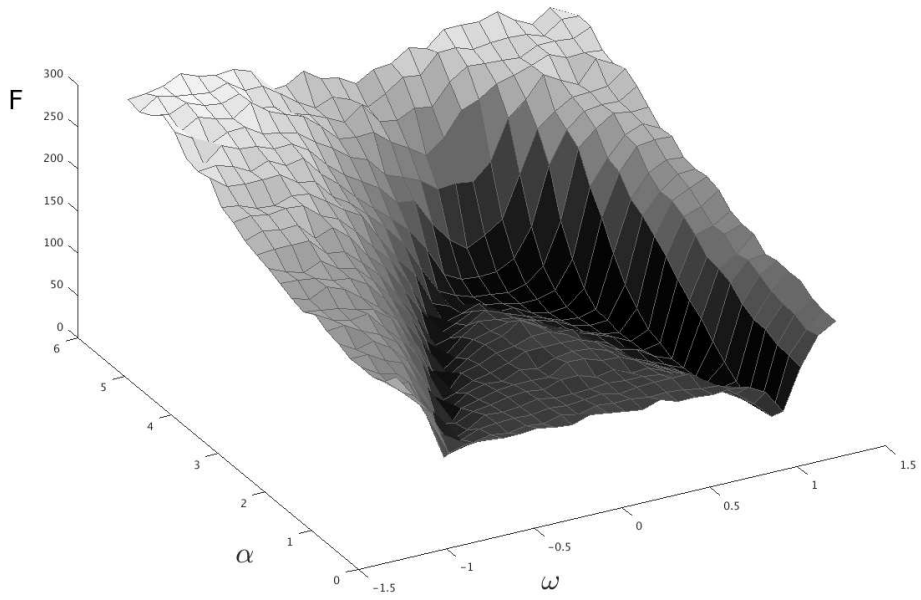


Figure 3.1: Typical PSO performance as a function of its ω and α parameters. Here a 25 particle swarm was run for pairs of ω and α values ($\alpha_1 = \alpha_2$). Cost function here was the $d = 10$ non-continuous rotated Rastrigin function (CEC2013). Each pairing was repeated 100 times and the minimal costs after 2000 iterations were averaged.

3.1.4 Matrix formulation

In order to analyse the behaviour of the algorithm it is convenient to use a matrix formulation by inserting the velocity explicitly in the second equation (3.1). If we consider particle states $\mathbf{z} = (\mathbf{v}, \mathbf{x})^T$ we can rearrange the update rules. In Eq. 3.2 we stack the updates for a single particle.

$$\begin{pmatrix} \mathbf{v_{t+1}} \\ \mathbf{x_{t+1}} \end{pmatrix} = \begin{pmatrix} \omega \mathbf{v_t} + \alpha_1 \mathbf{R_1} (\mathbf{p} - \mathbf{x_t}) + \alpha_2 R_2 (\mathbf{g} - \mathbf{x_t}) \\ \mathbf{x_t} + \omega \mathbf{v_t} + \alpha_1 \mathbf{R_1} (\mathbf{p} - \mathbf{x_t}) + \alpha_2 R_2 (\mathbf{g} - \mathbf{x_t}) \end{pmatrix} \tag{3.2}$$

In Eq. 3.3 we separate the state dependent parts from those elements that are independent of velocity or position.

$$\begin{pmatrix} \mathbf{v_{t+1}} \\ \mathbf{x_{t+1}} \end{pmatrix} = \begin{pmatrix} \omega \mathbf{I}_d & -\alpha_1 \mathbf{R_1} - \alpha_2 \mathbf{R_2} \\ \omega \mathbf{I}_d & \mathbf{I}_d - \alpha_1 \mathbf{R_1} - \alpha_2 \mathbf{R_2} \end{pmatrix} \begin{pmatrix} \mathbf{v_t} \\ \mathbf{x_t} \end{pmatrix} + \alpha_1 \mathbf{R_1} \begin{pmatrix} \mathbf{p} \\ \mathbf{p} \end{pmatrix} + \alpha_2 \mathbf{R_2} \begin{pmatrix} \mathbf{g} \\ \mathbf{g} \end{pmatrix} \tag{3.3}$$

In terms of our state variable $\mathbf{z} = (\mathbf{v}, \mathbf{x})^T$ this is more generally as shown in Eq 3.4

$$\mathbf{z}_{t+1} = M\mathbf{z}_t + \alpha_1 \mathbf{R}_1 (\mathbf{p}, \mathbf{p})^\top + \alpha_2 \mathbf{R}_2 (\mathbf{g}, \mathbf{g})^\top \tag{3.4}$$

with

$$M = \begin{pmatrix} \omega \mathbf{I}_d & -\alpha_1 \mathbf{R}_1 - \alpha_2 \mathbf{R}_2 \\ \omega \mathbf{I}_d & \mathbf{I}_d - \alpha_1 \mathbf{R}_1 - \alpha_2 \mathbf{R}_2 \end{pmatrix}, \tag{3.5}$$

where $\mathbf{I}_d$ is the unit matrix in d dimensions. Since the second and third term on the right in Eq. 3.4 only change on discovery of better solutions it is often the case that they will remain constant for many iterations. The analysis of the algorithm can focus on the properties of the matrix M. In spite of its wide applicability, PSO has not been subject to deeper theoretical study, which may be due to the multiplicative noise in the simple quasi-linear, quasi-decoupled dynamics. In previous studies the effect of the noise has largely been ignored.

3.2 Critical swarm conditions for a single particle

3.2.1 Outline of methodology

The following pages discuss the methods by which we can characterise the dynamics of our PSO swarm. As this covers a number of stages it is worth providing an outline

of the steps required.

1. We can consider the dynamics of a single particle as each particle is independent unless a new global best position is found. To do this we consider the system as a random dynamical system.

2. Initially we consider the dynamics of the particle whose personal best location is equal to the swarm's global best. Iterated updates may result in a converging or diverging swarm. We determine the conditions which lead to a swarm being stable i.e. marking the boundary between these two states.

3. State updates are linear. The dynamics are thus determined by the infinite product of stochastic update matrices. Furstenberg and Kesten (1960); Khas'minskii (1967) provide a solution for this. The system is characterised by considering its Lyapunov exponent. To do this the probability distribution of the particle or particles in the system's state space is required (termed invariant measure (Khas'minskii, 1967)). Additionally the distribution of stochastic matrices is required. The form of both of these distributions is dependent upon the parameters α_1, α_2, and ω. To calculate the Lyapunov exponent we utilise the linear nature of the system update rules. Particles can be consider to sit on a unit circle in the system's state space. A single update may result in a move inward or outward. Summing these movements over all updates weighted by where particles are likely to be in the state space yields our Lyapunov exponent.

4. This can be estimated numerically using a method proposed by (Vanneste, 2010).

5. We discuss the impact of personal best not equal to global best may have on the dynamics.

3.2.2 PSO as a random dynamical system

We study the dynamics of the particle swarm by looking at the behaviour of a single particle as done previously in earlier analyses (Kennedy, 1998; Trelea, 2003). This can be justified because the particles interact only via the shared knowledge of the global best position such that, while $\mathbf{g}$ is unchanged, single particles exhibit qualitatively the same dynamics as in the swarm. For the one-particle case we have necessarily $\mathbf{p} = \mathbf{g}$, such that shift invariance allows us to set both to zero in Eq. 3.4. This gives us the

following stochastic-map formulation of the PSO dynamics.

$$\mathbf{z}_{t+1} = M\mathbf{z}_t \tag{3.6}$$

Extending earlier approaches we will explicitly consider the randomness of the dynamics, i.e. instead of averages over $\mathbf{R}_1$ and $\mathbf{R}_2$ we consider a random dynamical system. Each iteration of the algorithm generates a new matrix (Eq.3.5) with freshly drawn random values for $\mathbf{R}_1$ and $\mathbf{R}_2$. The evolution of the swarm is thus determined by the multiplicative effects of these matrices. To explore this we can replace the α_1 and α_2 with a new parameter $\alpha = \alpha_1 + \alpha_2$, with $\alpha_1, \alpha_2 \geq 0$ and $\alpha > 0$. We rewrite the update matrix (Eq. 3.5) as a new dynamic matrix M given by

$$\mathcal{M}_{\alpha,\omega} = \left\{ \begin{pmatrix} \omega\mathbf{I}_d & -\alpha\mathbf{R} \\ \omega\mathbf{I}_d & \mathbf{I}_d - \alpha\mathbf{R} \end{pmatrix}, \mathbf{R}_{ij} = 0 \text{ for } i \neq j \text{ and } \mathbf{R}_{ii} \in [0,1], \right\}. \tag{3.7}$$

Here $\mathbf{R}$ in both rows is the same realization of a random diagonal matrix that combines the effects of $\mathbf{R}_1$ and $\mathbf{R}_2$. The diagonal elements of $\mathbf{R}_1$ and $\mathbf{R}_2$ are uniformly distributed in $[0,1]$. Thus the diagonal elements of $\alpha_1\mathbf{R}_1$ are uniformly distributed in $[0,\alpha_1]$, and the diagonal elements of $\alpha_2\mathbf{R}_2$ are uniformly distributed in $[0,\alpha_2]$. To understand the distribution of matrices that can be drawn from $\mathcal{M}_{\alpha,\omega}$ we consider the sum of two uniform random distributions of unequal range. This is is given by a convolution of the two probability distributions. The solution is given as follows (Lytle, 2001; Ma, 2011)

$$P_{\alpha\mathbf{R}}(s) = \begin{cases} \frac{s}{\alpha_1}\frac{\alpha_2-\alpha_1}{\alpha_1\alpha_2} & \text{if } 0 < s \leq \alpha_1 \\ \frac{\alpha_2-\alpha_1}{\alpha_1\alpha_2} & \text{if } \alpha_1 < s \leq \alpha - \alpha_1 \\ \frac{((\alpha_2+\alpha_1)-s)}{\alpha_1}\frac{\alpha_2-\alpha_1}{\alpha_1\alpha_2} & \text{if } \alpha - \alpha_1 < s \leq \alpha \end{cases} \tag{3.8}$$

if the variable $s \in [0,\alpha]$ and $P_{\alpha\mathbf{R}}(s) = 0$ otherwise. Equation 3.8 expresses the solution for the case where $\alpha_1 \leq \alpha_2$. If $\alpha_2 < \alpha_1$ then the solution has the same form but with α_1 and α_2 swapped. $P_{\alpha\mathbf{R}}(s)$ has a triangular shape for $\alpha_1 = \alpha_2$, a box shape in the limits $\alpha_1 \to 0$ or $\alpha_2 \to 0$, and is trapezoidal for other pairings of α_1 and α_2. Thus the selection of particular α_1 and α_2 parameters will determine the distribution matrices drawn from $\mathcal{M}$. We will call this distribution $P_{\alpha,\omega}(\mathcal{M})$. The shape of this distribution is therefore dependent upon the choice of parameters. For $\alpha_1 \approx \alpha_2$ deviations from the theory may occur because in the multi-particle case $\mathbf{p}$ and $\mathbf{g}$ will be different for most

particles. We will discuss this as well as the effects of the switching of the dynamics at discovery of better solutions in Sect. 3.4.3.

3.2.3 Stability

We may describe the behaviour of the PSO swarm by considering whether it is convergent, divergent or stable. This can be modelled by Lyapunov exponents. This provides a means to describe the dynamics of a system. If one considers two particles initially positioned arbitrarily close together and subject to the dynamics of the PSO update rules then we may observe whether the particles move apart or move together as the algorithm runs. Eq. 3.9 provides a model of our particles' motions (Wikipedia, 2016). The λ value captures the nature of the dynamics. For two points at time $t = 0$ a distance δZ_0 apart we can relate the future separation of these points δZ_t. For swarms where particles are converging the calculated λ is less than zero. For divergent swarms λ is greater than zero.

$$\delta Z_t = e^{\lambda t} \delta Z_0 \tag{3.9}$$

To estimate the Lyapunov exponent for PSO we can use the methods of Furstenberg and Kesten (1960); Khas'minskii (1967). As the system state is updated in a linear manner we can consider the effect of these updates on particles on a unit circle in the system's state space. Each update moves the particles inwards or outwards depending upon both where the particles are within the system's state space and the particular stochastic matrix drawn for the update. Clearly, with each iteration of the PSO algorithm, we will generate a different set of random numbers. These update particle states, given by Eq.3.6. The iterated behaviour of our swarm is thus determined by the product of stochastic matrices drawn from set $\mathcal{M}_{\alpha,\omega}$. We need to consider how such products behave on average. Such products have been studied for several decades (Furstenberg and Kesten, 1960) and have found applications in physics, biology and economics. In terms of stability of the swarm we wish to determine the α and ω pairs that result in swarms that are neither convergent nor divergent i.e. those that give rise to a Lyapunov exponent $\lambda = 0$. The analysis below derives this, alternatively one may use a numerical approach such as the resampled Monte Carlo method (Vanneste, 2010).

Whilst a stable swarm does not discover any new solutions, its dynamical properties are determined by an infinite product of matrices from the set $\mathcal{M}_{\alpha,\omega}$ given by Eq.3.7.

This provides a convenient way to explicitly model the stochasticity of the swarm dynamics such that we can claim that the performance of PSO is determined by the stability properties of the random dynamical system described by Eq.3.6.

Since the equation (3.6) is linear, the analysis can be restricted to vectors on the unit sphere in the $(\mathbf{v}, \mathbf{x})$ space, i.e. to unit vectors

$$\mathbf{a} = (\mathbf{x}, \mathbf{v})^\top / \| (\mathbf{x}, \mathbf{v})^\top \|, \tag{3.10}$$

where $\| \cdot \|$ denotes the Euclidean norm. In order to determine whether the swarm is on average expanding, contracting or stable we assess on each iteration whether particles move from the unit circle: outward for divergence; inward for convergence. Unless the set of matrices shares the same eigenvectors (which is not the case here) standard stability analysis in terms of eigenvalues is not applicable. Instead we will use means from the theory of random matrix products in order to decide whether the set of matrices is stochastically contractile i.e. does the swarm converge in the infinite limit given the distribution of matrices generated for given parameter values. Figure 3.2 visualises the process. A particle in the PSO swarm may be shown on a unit circle in the system's velocity:position state space. On each iteration the algorithm's update rules move the particle elsewhere in this space. If the particle moves outward from the unit circle then it is contributing to expansion of the swarm (as shown in the figure). If moving within the circle then it is contracting. To assess the behaviour of the whole swarm we will wish to sum these individual moves. At the end of the iteration we project the particle back to the unit circle. As the system update is linear, the scale of the system is not relevant. A matrix update that moves a particle 10% out from the circle or 50% in from the circle does so whatever the size of the circle. All updates are thus made relative to the unit circle for convenience. The process is repeated.

The resultant effect of the update shown in Fig 3.2 was to shift the position of the particle on the unit circle. We need to consider also the location of the particle in the system's state space when an update is applied. One way is to think of having multiple particles spread through the state space i.e. positioned around the unit circle. These other particles, that started elsewhere, may have finished in other locations. However, the dynamics of this system essentially treat the particles individually. Only when a new personal best improves upon the swarm's global best does one particle influence the others. In general such improvements are rare and as the algorithm runs they tend to occur less and less often. The stability of the system may thus be explored by only

considering a single particle. Any given update of particles in a swarm's state space will result in some particles moving outward, or inward from the unit circle. The likelihood of these moves is dependent upon where on the unit sphere a particle lies prior to the update being applied. In order to assess the whole effect we therefore need to determine the distribution of particles on the unit circle under the dynamics of the update rules and the specific parameter values. Whilst updates are stochastic some portions of the state space act somewhat like attractor states i.e. particles in these portions of the unit circle are likely to remain there perhaps only rare matrix updates will appreciably shift them elsewhere. Figure 3.3 visualises this for a single iteration of a small swarm. The particles in the top right are moved outward during the shown update. Their re-project will return them to a similar location. The remaining particles move inward, by a larger amount. On this iteration the sum of these moves may suggest the swarm is perhaps contracting a little. However, the particle in the bottom right will re-projected to join the main group of particles. If this upper left location has an overall tendency to be expansive then on subsequent iterations more and more particles may be contributing more divergent behaviour to the overall swarm.

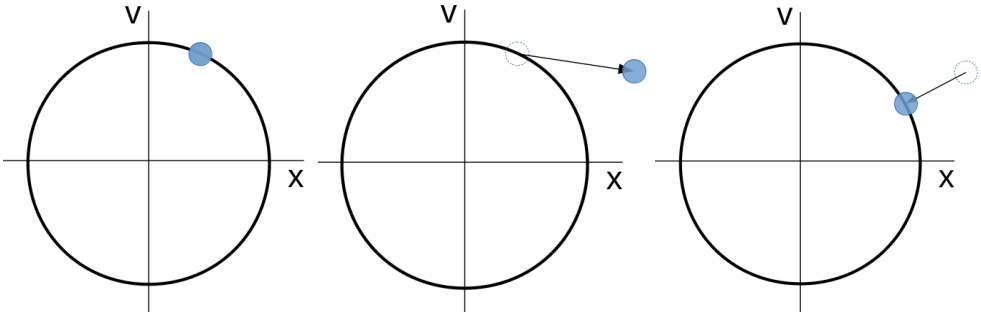


Figure 3.2: PSO update and projection onto unit circle process. Here we visualise the process of a single particle in one dimension. (Left) shows a single particle sitting on the unit circle in this state space. (Center) Applying PSO update rules results in the particle moving to a new state which is not on the unit circle. If the particle moves out from the circle it is expanding, if inward then contracting. (Right) the particle (whether outward or inward moving) is re-projected back to the unit circle ready for the next update. The process thus results in the particle remaining a single unit away from the state space origin, but its position on the unit circle may change on each update.

We can estimate the stationary distribution of particles on the unit circle, $\nu_{\alpha,\omega}(\mathbf{a})$, by the following process. A number of particles, $\mathcal{N}_p$, are placed around our unit circle. For each particle we create a set of new particle locations by applying the update Eq 3.6 $\mathcal{N}_u$ times. Each update is re-projected onto the unit circle (as per Eq 3.10). This gives us $\mathcal{N}_{new} = \mathcal{N}_p\mathcal{N}_u$ new particles. We can then randomly sample $\mathcal{N}_p$ of these

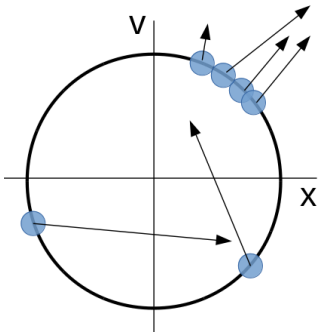


Figure 3.3: PSO update and projection onto unit circle process. Here we visualise a number of particles on the unit circle. To assess whether the whole swarm is expanding or contracting we need to consider the effects of both the location on the unit circle that each particles lies, and the dynamics of the PSO algorithm as applied to a particle at that location. Here the PSO update rules are moving the lower particles closer to the origin, so are contractile for those particles. However, the upper particles are moving outward representing a divergent move. Summing the sizes and directions of these moves gives an indication of the effect on the swarm for this iteration. However we can see that when the moved particles are re-projected onto the unit circle there will now be 5 particles in the top right area.

and repeat the process. Different PSO parameter values result in different stochastic update matrices ($\mathcal{M}_{\alpha,\omega}$). These yield different stationary distributions of the particles. Figure 3.4 shows a number of these. Equation 3.11 expresses this process for state space of dimension d.

$$\nu_{\alpha,\omega}(\mathbf{a}) = \int d\nu_{\alpha,\omega}(\mathbf{b}) \int dP_{\alpha,\omega}(M)\, \delta(\mathbf{a}, M\mathbf{b}/\|M\mathbf{b}\|), \ \mathbf{a},\mathbf{b} \in S^{2d-1}. \tag{3.11}$$

where $\nu_{\alpha,\omega}$ is the stationary distribution (called invariant measure by Khas'minskii (1967)) of the system i.e. the probability distribution of the particles in the system state space. For particles on the unit circle we derive this by considering their moves expressed by the δ function on a given update. This in integrated over the distribution of stochastic matrices $dP_{\alpha,\omega}(M)$.

The existence of the invariant measure requires the dynamics to be ergodic which is ensured if at least some of elements of $\mathcal{M}$ have complex eigenvalues, such as being the case for $\omega^2 + \alpha^2/4 - \omega\alpha - 2\omega - \alpha + 1 < 0$ (see above, (Trelea, 2003)). This condition excludes a small region in the parameters space at small values of ω, such that there we have to take all ergodic components into account. There are not more than two components which due to symmetry have the same stability properties. It depends on the parameters α and ω and differs strongly from a homogeneous distribution, see

Fig. 3.4 for a few examples in the case $d = 1$.

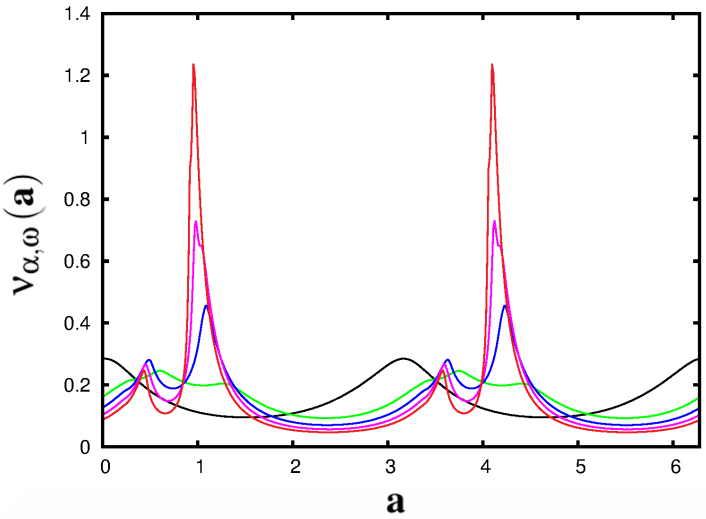


Figure 3.4: Stationary distribution $\nu_{\alpha,\omega}(\mathbf{a})$ on the unit circle ($\mathbf{a} \in [0, 2\pi)$) in the (x, v) plane for a one-particle system (3.6) for $\omega = 0.7$ and $\alpha = \alpha_2 = 0.5$ (black), 1.5 (green), 2.5 (blue), 3.5 (magenta), 4.5 (red). The distribution with peak near π is for $\alpha = 0.5$, otherwise main peaks are highest for largest α). The distribution $\nu_{\alpha,\omega}(\mathbf{a})$ is plotted vertically, against $\mathbf{a}$ horizontally.

The properties of the asymptotic dynamics of the PSO swarm can be described based on a double Lebesgue integral over the unit sphere S^{2d-1} and the set $\mathcal{M}$ (Khas'minskii, 1967; Tutubalin, 1965). As with Lyapunov exponents, the effect of the dynamics is measured in logarithmic units in order to account for multiplicative action.

$$\lambda(\alpha, \omega) = \int d\nu_{\alpha,\omega}(\mathbf{a}) \int dP_{\alpha,\omega}(M) \log \|M\mathbf{a}\| \tag{3.12}$$

If $\lambda(\alpha, \omega)$ is negative the algorithm will converge with probability 1, while for positive λ arbitrarily large fluctuations are possible. While the measure for the inner integral in Eq.3.12 is the distribution of the random multiplier as given earlier in Eq. 3.8. The stationary distribution ν on the unit sphere for the outer integral was given in Eq 3.11.

Critical parameters are obtained from Eq. 3.12 by the relation

$$\lambda(\alpha, \omega) = 0. \tag{3.13}$$

Solving Eq. 3.13 is difficult in higher dimensions, so we rely on the linearity of the system when considering the $(d = 1)$-case as representative. Figure 3.5 represents

solutions of Eq. 3.13. Alternatively one may use a numerical approach such as the resampled Monte Carlo method (Vanneste, 2010). This approach is described in Appendix E. The figure shows three curves, representing three potential solutions to this problem. Parameter pairs that lie inside a given curve give rise to swarms whose largest Lyapunov exponent is less than zero. These swarms are therefore stable in the sense that they will converge or become static. For parameter pairings outside each curve, the swarm generated will have a largest Lyapunov exponent greater than zero and will tend to explode. Parameter pairs that yield swarms with Lyapunov exponents equal to zero are therefore stable in the infinite limit. However they can make deviations into either divergent or convergent behaviours for extended periods of time. We should note that this arises from the infinite product of our stochastic matrices and is true whether we have one or many particles. PSO experiments use finite iteration counts, so any individual trial may yield a set of random matrices whose product may generate a behaviour that for the limited nature of a single experiment differs from this theoretical approach. Averaged, though, we would expect this picture to emerge. The set of parameter pairs on the curve result in a critical swarm, whose deviations are not described by either convergence or divergence.

The solid black curve in Fig. 3.5 represents the solution for $d = 1$, $\alpha = \alpha_1 + \alpha_2$ and $\alpha_1 = \alpha_2$. The black dashed curve is the solution for $d = 1$, $\alpha = \alpha_2$ and $\alpha_1 = 0$. This results in a larger variance rendering the dynamics less stable. This results in the locus of stability moving inwards, towards the origin somewhat. The green dotted curve shows the curve of stability predicted by (Jiang et al., 2007), and given as a polynomial in (Cleghorn and Engelbrecht, 2014a). Our approach generated the black curves shown depending on the relative values of α_1 and α_2. Experimentally it is shown below that the outer curve, rather than the Jiang curve, represent the limit parameter pairs that lead to convergent swarms.

Inside the contour $\lambda(\alpha, \omega)$ is negative, meaning that the state will converge with probability 1. Along the contour and in the outside region large state fluctuations are possible. Interesting parameter values are expected near the curve where due to a coexistence of stable and unstable dynamics (induced by different sequences of random matrices) a theoretically optimal combination of exploration and exploitation is possible. For specific problems, however, deviations from the critical curve can be expected to be beneficial. In order to solve a practical problem there is usually a finite iteration budget for the algorithm. In that case it is beneficial to allow the swarm to converge at some point. Thus the swarm being somewhat sub-critical, to allow such a

convergence is desirable. The degree to which this we should allow this is a function of the iteration budget and is also related to the nature of the problem space being explored.

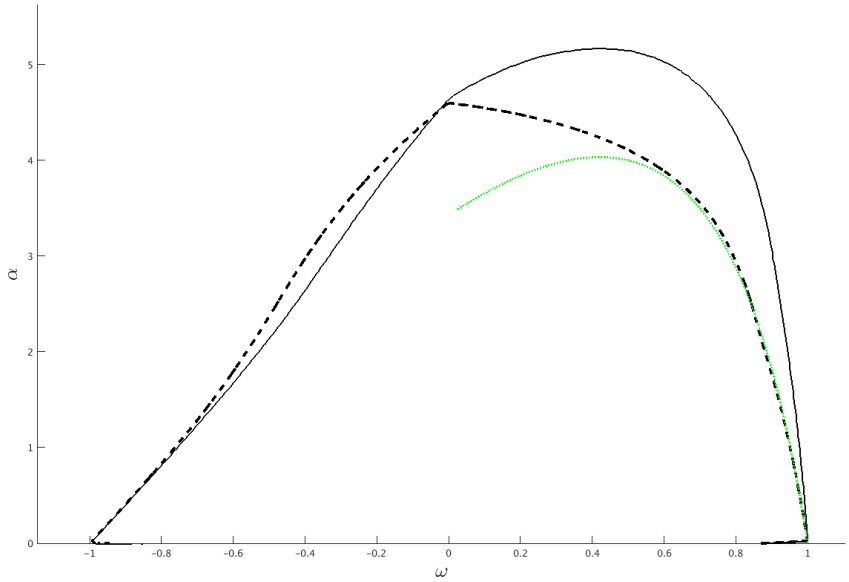


Figure 3.5: Solution of Eq. 3.13 representing a single particle in one dimension with a fixed best value at $\mathbf{g} = \mathbf{p} = 0$. The curve that has higher α-values on the right (black solid curve) is for $\alpha_1 = \alpha_2$, the other curve (black dashed curve) is for $\alpha = \alpha_2$, $\alpha_1 = 0$. Except for the regions near $\omega = \pm 1$, where numerical instabilities can occur, a simulation produces an indistinguishable curve. In the simulation we tracked the probability of a particle to either reach a small region (10^{-6}) near the origin or to escape beyond a radius of 10^6 after starting from a random location on the unit circle. Along the curve both probabilities are equal. The green dotted curve is that predicted by (Jiang et al., 2007) as given in (Cleghorn and Engelbrecht, 2014a) for comparison.

3.2.4 Personal best vs. global best

Due to linearity, the particle swarm update rule (3.1) is subject to a scaling invariance which was already used in Eq. 3.10. We now consider the consequences of linearity for the case where personal best and global best differ, i.e. $\mathbf{p} \neq \mathbf{g}$. For an interval where $\mathbf{p}_i$ and $\mathbf{g}$ remain unchanged, the particle i with personal best $\mathbf{p}_i$ will behave like a particle in a swarm where together with $\mathbf{x}$ and $\mathbf{v}$, $\mathbf{p}_i$ is also scaled by a factor $\kappa > 0$.

The finite-time approximation of the Lyapunov exponent (see Eq. 3.12)

$$\lambda(t) = \frac{1}{t} \log \langle \|(\mathbf{x}_t, \mathbf{v}_t)\| \rangle \tag{3.14}$$

will be changed by an amount of $\frac{1}{t} \log \kappa$ by the scaling. Although this has no effect on the asymptotic behaviour, we will have to expect an effect on the stability of the swarm for finite times which may be relevant for practical applications. For the same parameters, the swarm will be more stable if $\kappa \to 0$ and less stable for increasing κ, provided that the initial conditions are scaled in the same way. Likewise, if $\|\mathbf{p}\|$ is increased, then the critical contour will move inwards, see Fig. 3.7. Note that in this figure, the low number of iterations lead to a few erroneous trials at parameter pairs outside the outer contour which have been omitted here. We also do not consider the behaviour near $\alpha = 0$ which is complex but irrelevant for PSO. The contour (Eq. 3.13) can be seen as the limit $\kappa \to 0$ such that only an increase of $\|\mathbf{p}\|$ is relevant for comparison with the theoretical stability result. When comparing the stability results with numerical simulations for real optimisation problems, we will need to take into account the effects caused by differences between $\mathbf{p}$ and $\mathbf{g}$ in a multi-particle swarm with finite runtimes.

3.3 Simulations

Metaheuristic algorithms are often tested in competition against benchmark functions designed to present different problem space characteristics. To confirm the stability results above testing against the 28 functions defined for the 2013 International Conference on Evolutionary Computation (CEC2013) was performed. This contains a mix of unimodal, basic multimodal and composite functions designed to provide a range of challenges to various metaheuristic algorithms. In the competition the domain of the functions in this test set are all defined to be $[-100, 100]^d$ where d is the dimensionality of the problem. We retain this only as the spatial region within which the particles are initialised. We use 10-dimensional problems throughout. Our implementation of PSO performed no spatial or velocity clamping. Particles are free to travel where they wish so long as they are not clearly diverging. We define this as the case where absolute particle positions or velocities are greater than 10^{12}. In this way we ensure minimal interference with the dynamics of the particles in motion. In all trials a swarm of 25 particles was used. We repeated the algorithm 100 times, on each

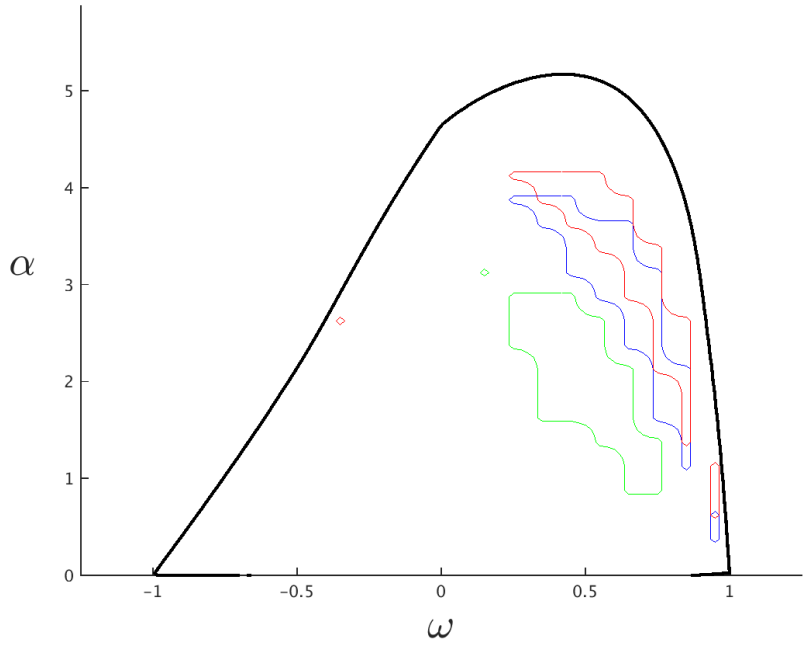


Figure 3.6: The 5% best parameter regions for 20 (green), 200 (blue), and 2000 (red) iterations. As we increase the number of iterations performed the best performing parameter pairs shifts toward the critical curve. Functions costs were averaged over 100 runs and all 28 CEC benchmark functions. The curve of zero Lyapunov exponent for $N = 1$, $d = 1$, $\alpha = \alpha_1 + \alpha_2$, and $\alpha_1 = \alpha_2$ is shown in black.

occasion allowing 20, 200, 2000 iterations to pass before recording the best solution found by the swarm. For the competition 50000 fitness evaluation were allowed which corresponds to 2000 iterations with 25 particles. Other iteration numbers were included for comparison. This protocol was carried out for pairs of $\omega \in [-1.1, 1.1]$ and $\alpha \in [0, 6]$ This was repeated for all 28 functions. The averaged solution costs as a function of the two parameters showed curved valleys similar to that in Fig. 3.1 for all problems. For each function we obtain different best values along (or near) the theoretical curve defined by the stability condition Eq. 3.13. There appears to be no preferable location within the valley. Some individual functions yield best performance near $\omega = 1$. The global average performance over all test functions is better in the valley somewhere between the extremes of $\omega = 0$ and $\omega = 1$, see Fig 3.6.

3.3.1 Combining the results from multiple cost functions

Whilst the stochastic map, Eq 3.6 defined the essential dynamics of the system, a real swarm with many particles and many shifts in their p_i and g values will cause small

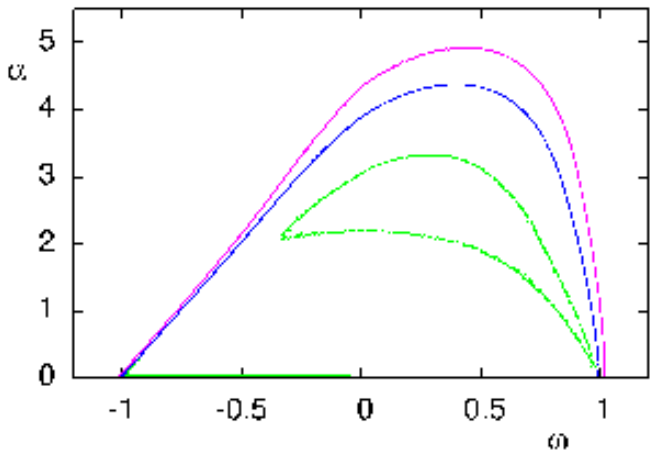


Figure 3.7: For $\mathbf{p} \neq \mathbf{g}$ we define neutral stability as the equilibrium between divergence and convergence. Convergence means here that the particle approaches the curve connecting $\mathbf{p}$ and $\mathbf{g}$. Curves are for a one-dimensional problem with $\mathbf{p} = 0.1$ and $\mathbf{g} = 0$ scaled (see Sect. 3.2.4) by $\kappa = 1$ (outer curve) $\kappa = 10$ and $\kappa = 25$ (inner curve). Results are for 200 iterations and averaged over 100000 repetitions.

deviations that may make interpretation of results less clear. These effects will differ for different problems. Exploration of a Sphere function will result in many early shifts in the particle knowledge terms leading to quick agreement of a solution. It is less easy to envisage the effects on more complex function surfaces. Therefore it is useful to average these effects by combining the 28 functions used. The potential range of cost values returned by each function varies widely. The Sphere function may return values whose magnitudes may vary over many decades. Other functions have differences between minimum and maximum values of only one or two decades. Simply averaging the results from these will result in the values from poorly performing swarms acting on a Sphere problem swamping the result.

To more fairly combine the results an estimate of the average value of each function is used to normalise the results obtained for each problem function. The following procedure was used: Each function was randomly sampled within the $[-100, 100]^d$ spatial domain. These data were used to estimate an average value of each function. These average values, V_{ave} are then used to scale the results. This is not quite normalizing into the range $[0, 1]$ as poorly performing swarms may in fact do worse than the average of random sampling. In practise results are broadly scaled to similar ranges, see Figure 3.8.

$$V'_{\alpha,\omega} = (V_{\alpha,\omega} - V_{min})/(V_{ave} - V_{min}) \tag{3.15}$$

At each α, ω pair PSO was executed 100 times for a given objective function. The average cost function value at each pairing forms a set of results $V_{\alpha,\omega}$, the smallest of which is V_{min}. $V'_{\alpha,\omega}$ is the approximately normalised set of results. We can use this to sum and average across all objective functions as they are all now roughly in the range [0,1], so no function dominates the results of the others.

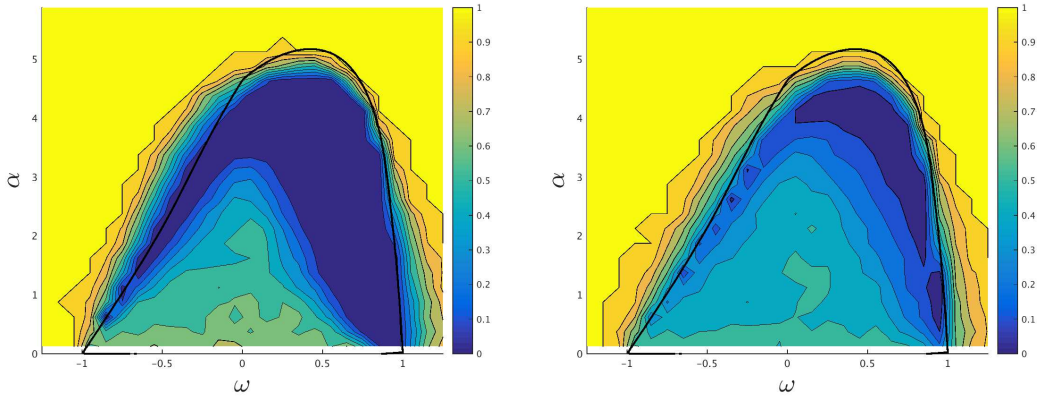


Figure 3.8: Cost function values rescaled using estimate of average values. Left hand image is for a Sphere Function. This cost function returns values that range over many decades of scale. Right hand image is of non-continuous rotated Rastrigan's Function. This cost function returns a much more restricted range of values. Here, each function's average has been estimated and used to scale the results. Both now show results in approximately in the range $[0, 1]$. For each function our PSO algorithm is run 100 times at each $\omega : \alpha$ pairing. Each run lasted 2000 iterations. Both retain the characteristic shape showing the best performing parameter pairings.

3.4 Discussion

3.4.1 Relevance of criticality

Our analytical approach predicts a locus of α and ω pairings that maintain the critical behaviour of the PSO swarm. The precise shape depends upon these parameters and the relative values of α_1 and α_2. Outside this curve the swarm will diverge unless steps are taken to constrain it. Inside, the swarm will eventually converge to a single solution. On the curve the swarm is poised between divergence and convergence. In order to locate a solution within a problem space the swarm needs to converge at some point, so the curve represents an upper bound on the exploration:exploitation mix that a swarm manifests. For parameters on the critical curve, fluctuations are still arbitrary

large. Therefore, sub-critical parameter values can be preferable if the settling time is of the same order as the scheduled runtime of the algorithm. Additionally, if a typical length scale of the problem is known, then the finite standard deviation of the particles in the stable parameter region can be used to decide about the distance of the parameter values from the critical curve. These dynamical quantities can be approximately set, based on the theory presented here, such that a precise control of the behaviour of the algorithm is in principle possible.

The observation of the distribution of empirically optimal parameter values along the critical curve confirms the expectation that critical or near-critical behaviour is the main reason for success of the algorithm. Critical fluctuations are a plausible means to perform well in a search problem if little is known about the cost landscape apart from perhaps certain smoothness assumptions. The majority of excursions will exploit the smoothness of the cost function by local search, whereas the fat tails of the distribution allow the particles to escape from local minima.

3.4.2 Comparison with earlier explanations

In figure 3.5 critical loci of parameter values are plotted. These, as discussed, are derived from the treatment of PSO as a random dynamical system. Two curves are plotted in black: one (inner) represents particles that have $\alpha = \alpha_2$, $\alpha_1 = 0$; the other (outer) represents particles with $\alpha = \alpha_1 + \alpha_2$ and $\alpha_1 = \alpha_2$. We also plot a curve (in dark green) showing an oft cited earlier analysis of swarm stability (Jiang et al., 2007). Empirically we can show that this latter solution must be wrong. We note that inspection of the curves in figure 3.5 suggest that there may exist a simple relationship between our solution and that of Jiang et al. (2007). Visual inspection suggesting that our outer curve's vertical position may be 1.25 times that of the earlier solution. Figure 3.9 replots Jiang's curve with this multiplier applied. We can see that this relationship is not true and indeed that the rescaled curve's shape now appears different enough that such a comparison is unlikely.

3.4.2.1 Curves of stability when viewed against average cost function results

Following the procedure in section 3.3.1 allows us to average the solutions achieved for the 28 functions used. Figure 3.10 shows where in parameter space the best results can be obtained (on average) across all functions. With lower iteration budgets available to the algorithm better results occur where the swarm is able to converge early and exploit

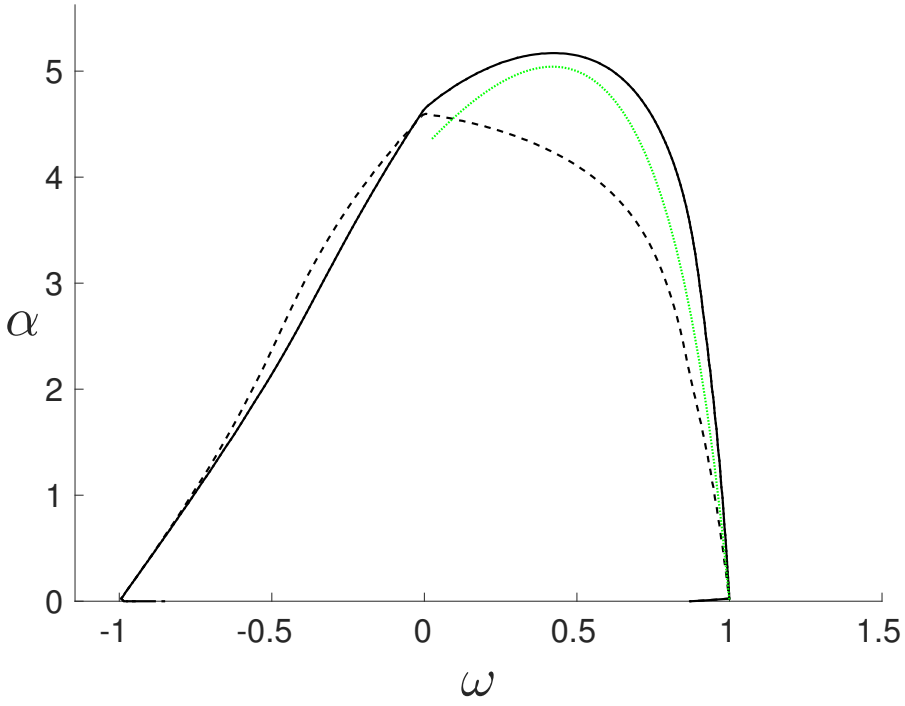


Figure 3.9: Our stability compared with 1.25 * Jiang's curve. Our stability curves as previously shown in figure 3.5 are compared to the green dotted curve showing the curve predicted by (Jiang et al., 2007) multiplied in the α direction by 1.25.

whatever knowledge is gained early. As we allow greater iterations the algorithm, as expected, benefits from slower convergence. This figure also shows the stability curves predicted in this thesis and that previously presented by Jiang et al. (2007). For the lower iteration counts all the good locations occur within both curves. With 2000 iterations some of the best locations sit outside the Jiang curve. These curves represent the point beyond which we would expect divergence in the PSO swarm. We would expect that good solutions should not occur beyond these curves. Of course as the algorithm is stochastic it is feasible that some runs with parameters taken from outside the curve may result in good solutions. However, this result is suggestive that the Jiang curve may not be a good solution for stability prediction.

3.4.2.2 Curves of stability when viewed against individual cost function results

Average results for PSO run against 28 cost functions for 2000 iteration suggest that the Jiang stability curve doesn't contain all the best parameter pairings. Looking at the best locations for individual functions (see Appendix A for examples) demonstrates

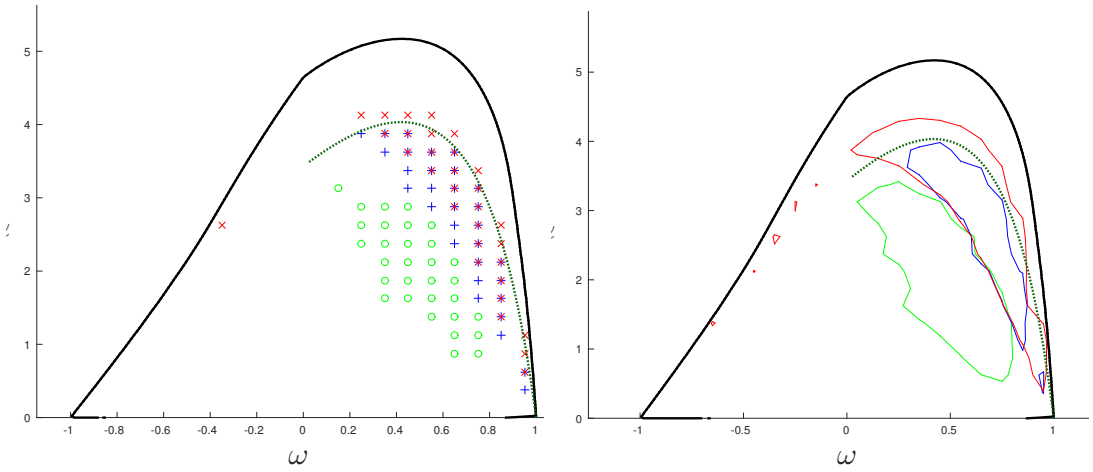


Figure 3.10: Average cost function value as function of iteration budget. Iteration budgets of 20 (green), 200 (blue) and 2000 (red) are shown. The points in the left hand figure and the contours in the right hand image represent the 5% best parameter pairings. The locus of stable parameter pairs as predicted by this thesis (black) and Jiang (dark green) are shown.

that variability exists between cost functions. Certain problems yield good solutions for parameter pairs further within the stable region than others. The effect of averaging over all cost functions will thus tend to move the best locations further within the stable region. The Jiang curve will tend to look like a better solution when viewed on average rather than against more extreme results.

Yet the upper limit of stability should be the same for all functions. We can look at specific functions to highlight this. Figure 3.11 shows that for some functions the location of best results in the parameter space is outside the Jiang curve.

3.4.2.3 Curves of stability when viewed against detailed cost function results

Jiang's result appears at odds with the empirical results. Instead the critical locus predicted by treating PSO as a random dynamical system appears to be consistent with the results so far seen. All best results lie on parameter pairs within the locus of stability predicted. We would expect that as iteration budget is increased we would continue to move away from the Jiang curve, but remain constrained by the curve our treatment predicts.

To explore this we chose two functions (7 and 21 in the CEC2013 set) and ran the PSO algorithm for parameters pairs along a slice of the parameter space lying on $\omega = 0.55$. This ω value was selected as there was a large vertical separation between the two proposed stability solutions. The α values were varied with a much finer

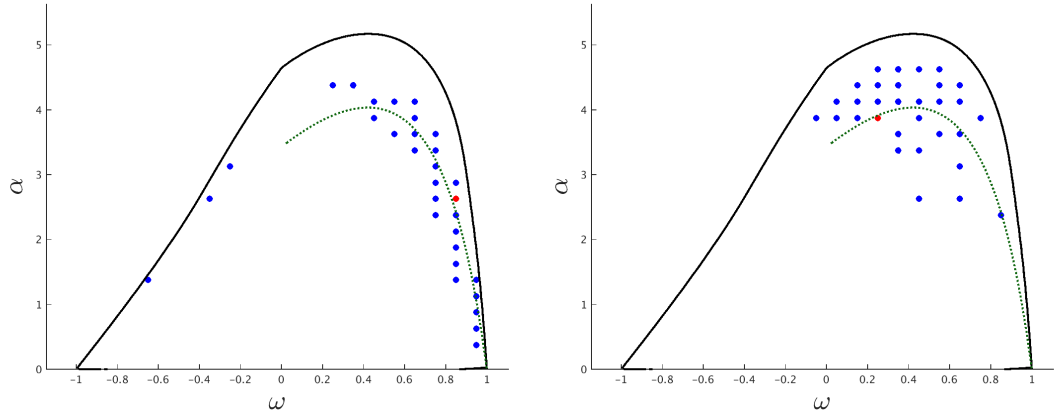


Figure 3.11: Specific cost function value as function of iteration budget. PSO is run for 2000 iterations against CEC2013 functions 7 (left) and 21 (right). 5% best parameter pairs are shown (blue dots), best parameter pair (red dot). As elsewhere this thesis' stability curve is shown in black and Jiang's in dark green.

granularity from 2 to 6. Again 100 repetitions were performed and iteration budgets of 20, 200, 2000 and 20000 were used.

Figure 3.12 shows the result. The stability solutions are shown as vertical lines. The α stability position of our solution was estimated to be at $\alpha = 5$. For the Jiang curve the polynomial fitting the curve was used to calculate its position as $\alpha = 3.94$. With increasing iteration budget we appear to get closer to the stability limit and achieve better results. It is also clear that results are better outside Jiang's predicted curve.

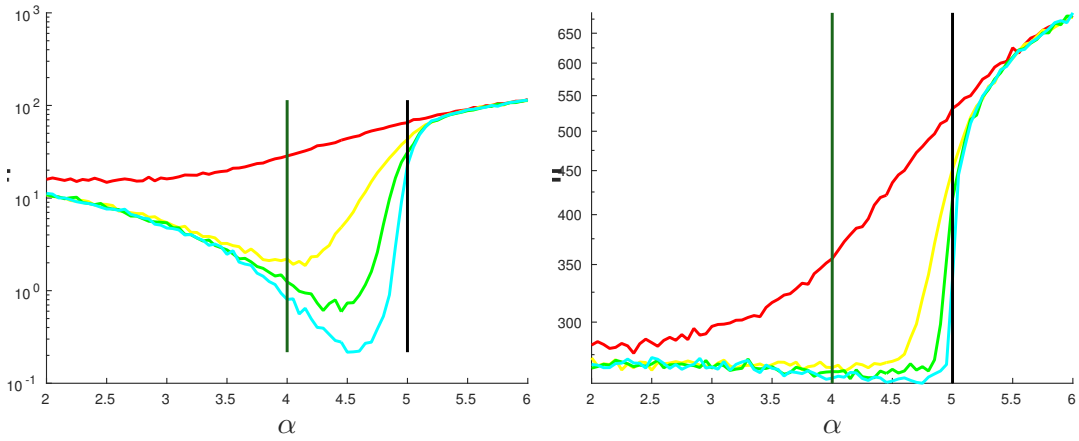


Figure 3.12: Detailed cost function value as function of iteration budget. PSO is run for 20 (red), 200 (yellow), 2000 (green) and 20000 (cyan) iterations against CEC2013 functions 7 (left) and 21 (right). The parameter $\omega = 0.55$ throughout, α varies from 2 to 6. As elsewhere this thesis' stability curve is shown in black and Jiang's in dark green. Note logarithmic scale used.

3.4.3 Switching dynamics at discovery of better solutions

Eq. 3.4 shows that the discovery of a better solution affects only the constant terms of the linear dynamics of a particle, whereas its dynamical properties are governed by the linear coefficient matrices. However, in the time step after a particle has found a new solution the corresponding force term in the dynamics is zero (see Eq. 3.1) such that the particle dynamics slows down compared to the theoretical solution which assumes a finite distance from the best position at all (finite) times. As this affects usually only one particle at a time and because new discoveries tend to become rarer over time, this effect will be small in the asymptotic dynamics, although it could justify the empirical optimality of parameters in the unstable region for some test cases.

We can consider what happens when new discoveries are made. A weakly converging swarm can still produce good results if it often discovers better solutions by means of the fluctuations it performs before settling into the current best position. For cost functions that are not 'deceptive', i.e. where local optima tend to be near better optima, parameter values far inside the critical contour (see Fig. 3.5) may give good results, while in other cases more exploration is needed. The iteration budget available for solving a given problem may give some guidance on how subcritical we wish the swarm to be. Small budgets should favour highly subcritical behaviour.

3.4.4 The role of personal best, present best and best ever

A numerical scan of the (α_1, α_2) plane shows a valley of low fitness values. For negative ω values there is a linear relationship with α. For small positive ω values, a roughly linear and relation with $\alpha_1 + \alpha_2 = \text{const}$ is seen, i.e. only the joint parameter $\alpha = \alpha_1 + \alpha_2$ matters. For large ω, and accordingly small predicted optimal α values, the valley is less straight. This may be because the effect of the known solutions is relatively weak, so the interaction of the two components becomes more important. In other words if the movement of the particles is mainly due to inertia, then the relation between the global and local best is non-trivial, while at low inertia the particles can adjust their $\mathbf{p}$ vectors quickly towards the $\mathbf{g}$ vector such that both terms become interchangeable.

At medium values of ω the difference between the solutions for $\alpha_1 = \alpha_2$ and for $\alpha_1 = 0$ is strongest, see Fig. 3.7. In simulations this shows to a lesser extent revealing a shortcoming of the one-particle approximation. Because in the multi-particle case, $\mathbf{p}$ and $\mathbf{g}$ are often different, the resulting vector will have a smaller norm than in the

one-particle case, where $\mathbf{p} = \mathbf{g}$. The case $\mathbf{p} \neq \mathbf{g}$ violates the assumption of that the dynamics can be described based unit vectors. While a particle far away from both $\mathbf{p}$ and $\mathbf{g}$ will behave as predicted from the one-particle case, at length scales smaller than $\|\mathbf{p}-\mathbf{g}\|$ the contractile forces will tend to be reduced such that the inertia becomes more effective and the particle is locally less stable which shows numerically in optimal parameters that are smaller than predicted.

Finally, we should mention that more particles, longer runtime (as shown earlier) as well as lower search space dimension increase the potential for exploration. They all lead to the empirically determined optimal parameters being closer to the critical curve than in the opposite cases where search in a relatively larger space benefits from a local search such that smaller parameters are favoured. The effect of lower spatial dimension can be seen in figure 3.13. The best parameter pairings moves closer to critical curve as the problem space is reduced.

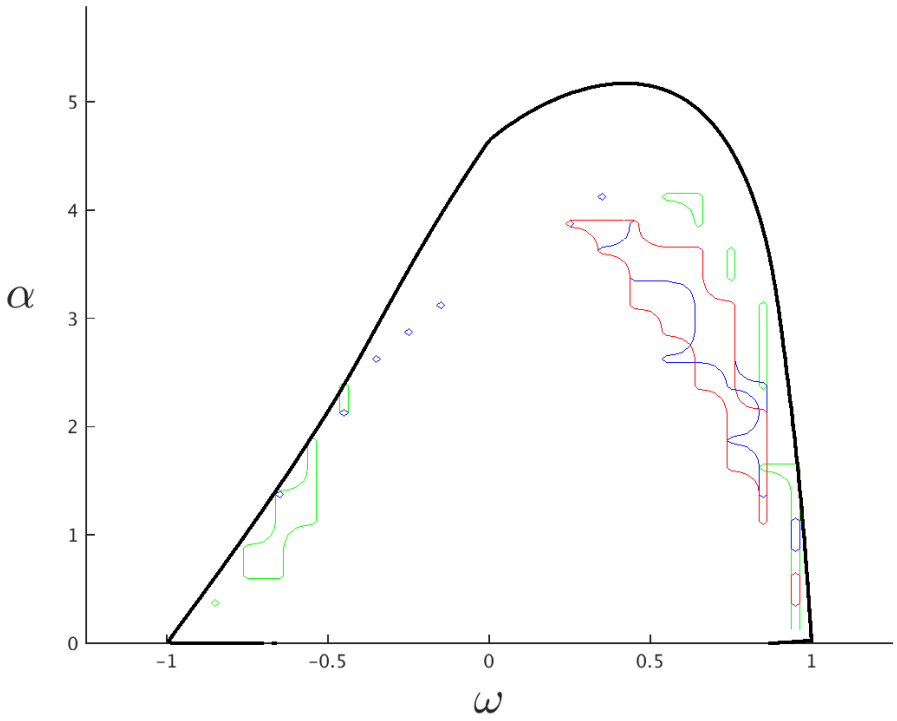


Figure 3.13: (left) The 5% best parameter regions for 2D (green), 5D (blue), and 10D (red) problem spaces: For lower dimensional problem spaces the region shifts towards the critical curve. Cost averaged over 100 runs and 28 CEC benchmark functions Zero Lyapunov exponent for $N = 1$, $d = 1$, $\alpha_1 = \alpha_2$ (black).

3.5 Conclusion

PSO is a widely used optimization scheme which is theoretically not well understood. Existing theory concentrates on a deterministic version of the algorithm which does not possess useful exploration capabilities. We have studied the algorithm by means of a product of random matrices. This predicts parameter pairs that give rise to stable swarms. Empirically optimal solutions occur as one approaches this curve of stability from the subcritical side. A weakness of the current approach is that it focuses on the standard PSO (Kennedy and Eberhart, 1995) which is known to include biases (Clerc, 2006; Spears et al., 2010), that are not necessarily justifiable, and is outperformed on benchmark sets and in practical applications by many of the existing PSO variants. Similar analyses are certainly possible and are expected to be carried out for some of the PSO variants or other metaheuristics.

Chapter 4

CriPS: A near critical swarm

4.1 Introduction

The previous chapter showed that for the standard Particle Swarm Optimisation (PSO) algorithm there exists a locus of parameter pairings that result in the swarm behaving in a critical manner. The set of ω and α pairings defining this curve represent the division between divergent and convergent behaviours. Empirically PSO performs best on the sub-critical side. The proximity of parameters to this curve is dependent in part upon the number of iterations that the algorithm is run for. Too close and there is a risk that due to the algorithm's stochastic nature a sequence of updates leads, by chance, to a divergent swarm. Even if convergent the swarm may spend too little time exploiting its knowledge of good locations. Too far from the curve and the swarm will stagnate early.

However we also saw that the curve of stability is influenced by the distance between a particle's personal best and the swarm's best values. The likely distances between these points will be dependent, at least to some extent, upon the nature of the problem space being solved. For a sphere function, for example, improvements are only ever found for particles moving closer to the true global optima. As problem surface complexity increases (in the sense of many hard to locate and spatially dispersed optima) this is no longer true. A particle exploring further out may happen upon a new best solution that will then be shared with the swarm.

For an unknown problem we lack knowledge regarding the likely distribution of distances between personal and global bests that a PSO swarm will experience as it attempts to locate good solutions. We know that we want to select parameters that lead to a slightly sub-critical swarm for the number of iterations we wish to run for.

Whilst we can select ω and α values for the standard PSO algorithm that ensure that it is operating just below the critical region it fails to adapt to particular problem spaces except by its reaction to updates to the best solutions found. Ideally we would wish to gain the benefits of a convergent swarm, whilst avoiding stagnation. In this chapter we modify the PSO algorithm online by using the swarms' interaction with the problem space on each iteration to adjust the parameters dynamically. We will nudge a diverging swarm toward exploitation and a converging swarm to be more adventurous. Having seen the benefits of criticality (or at least sub-criticality to PSO) we would like our swarm to still manifest as critical (or sub critical).

The novel algorithm presented is called Critical Particle Swarm Optimisation (CriPS). An earlier investigation suggested that one might potentially balance the swarm behaviours by utilising a feedback signal (Cordero, 2012). However in that approach the swarm used was spatially constrained and required careful tuning to avoid stagnation. These limitations are removed here. A version of this work was presented at the 2015 European Conference on Artificial Life (Erskine and Herrmann, 2015a). Here it is extended to fully describe the approach. The mechanism is shown to to avoid the need to set any parameter to specific values. Additionally stagnation is avoided. The behaviour of the swarm is not believed to be critical (or only rarely so) but does exhibit a statistically balanced mix of exploitative and exploratory behaviours in a somewhat sub-critical manner. There is some evidence that some degree of criticality tuning could be performed automatically. However in chapter 3 we saw that sub-criticality may be the preferred state for the swarm from a practical perspective. Performance comparisons with simple PSO variants show good results. Performance comparisons with more adept metaheuristic algorithms (such as those competing in CEC2013) yield modest results. This is to be expected as CriPS's performance may be upper bounded by the best results achievable by standard PSO given its best parameter settings for a given problem. The advantage of CriPS is that this setting need not be known.

4.2 Approach

Particle Swarm Optimisation (PSO) makes use of a dynamical system for solving a search task. Instead of adding search biases in order to improve performance in certain problems, we aim to remove algorithm-induced scales by controlling the swarm with a mechanism that is scale-free except possibly for a suppression of scales beyond the

system size. In this way a very promising performance is achieved due to the balance of large-scale exploration and local search. The dynamics of the swarm are derived in part from the original PSO update rules, but also from modifications made to the algorithm's parameters via a feedback signal derived from the swarms interaction with the objective function. The Critical Particle Swarm (CriPS) can be easily combined with many existing extensions such as chaotic exploration, additional force terms or non-trivial topologies.

The CriPS approach adjusts the algorithm's parameters online. A range of PSO variations were mentioned earlier in terms of velocity truncation (Kennedy, 1998), or velocity constriction (Clerc and Kennedy, 2002). There are many other parameter modifications methods that have been explored, such as online decreasing of ω to encourage exploitation of problem space knowledge (Shi and Eberhart, 1999). It has also been proposed to explicitly increase the diversity, e.g. by using repulsion (Chowdhury et al., 2013; Riget and Vesterstrøm, 2002) or random velocities (García-Villoria and Pastor, 2009). These approaches show potential improvements over a standard PSO approach. Thus we may be motivated to suspect that variation and/or control of the swarm during the algorithm's execution may be a useful strategy.

Our aim is to modify the swarm dynamics of the PSO algorithm such that exploration and exploitation of the problem space are statistically balanced automatically. We know that this can be achieved by the algorithm if its parameters are appropriately set. For a black box problem we may not yet know the appropriate settings. However, we can attempt to modify the algorithm's parameters online such that the swarm spends at least some time *correctly* set. Intuitively, we induce a more stable behaviour should the swarm tend to diverge, but change the parameters of the algorithm towards the unstable regime if the swarm is likely to collapse. In this way we hope to achieve an optimal compromise between local fine-grained search near candidate optima and large scale exploration. The swarm will stay near a critical regime between stability and instability which sometimes is compared to the *edge of chaos* (Langton, 1990), although the term *criticality* seems more suitable here.

Another motivation of our approach can be drawn from Bayesian theory. An algorithm that traverses the search space in a step-wise fashion, specifies implicitly or explicitly a prior for this step size, i.e. an assumption that steps should have a certain length distribution in order to optimally approach the goal of the search. If it is known that the goal is within a certain box, the optimal distribution will assign

zero probability to lengths larger than the size of the box. If the objective function is smooth, then often a minimal step size can be fixed which may help to speed up the search. If, however, no information on the problem scale is available, then the best choice to use a non-informative prior that allows for step sizes on all scales. The shape of such a prior is given by a power-law with an exponent of unity, but also for other exponents the step lengths can still be scale-free. The exponent may then be determined by additional information on the dimensionality of the problem or from the general structure of the objective function: If the objective has a number of smooth local optima, then relatively more of the larger jumps are needed than for a function that is shaped like a fractal. In the previous chapter determination of criticality arose from consideration of the stability of the dynamics of the system. Here, we will adjust the parameters of the algorithm as it runs, thus the equivalent approach is not possible. Instead we can look for the presence of power-laws in the distribution of the sizes of system events. We will study this in the results section.

While criticality in piles of granular matter happens around specific angles, criticality of a swarm of particles can be described in a much easier way: the swarm can only be critical if it neither collapses nor diverges. First the CriPS algorithm is presented in section 4.3.1. The algorithm is compared to variants of the standard PSO approaches (section 4.4.4) and with a set of more modern metaheuristic algorithms (Sections 4.4.5 and 4.4.6). Whilst performance is clearly of interest, we are keen to establish that our aims (the avoidance of stagnation, and a feedback driven mixing of exploration and exploitation) are met. These are shown in sections 4.4.1 and 4.4.3. Finally we will discuss our findings and address the relation to the theory of criticality and to other optimisation algorithms that are based on criticality in section 4.5.

4.3 Methods

4.3.1 The CriPS algorithm

Our aim is to utilise the critical dynamics of the PSO algorithm. Rather than adding additional parameters, or simply picking numbers from a power law distribution, our algorithm uses the dynamic of the swarm itself as a signal to modify its future behaviour. In order to maintain the responsiveness of the swarm to the objective function, we will control the parameters based on a measure of the swarm's diversity: this being any measure that is indicative of the spatial extent to which the swarm's

particles are distributed through the domain of the problem space.

Multiple metrics can be chosen to efficiently analyse the dynamics of the swarm. Simple measures such as the average distance between every particle of the swarm, or the average distance between every particle and the centroid of the swarm are both clearly concerned with the size of the swarm. The difference between two successive such measurements will indicate whether the swarm is growing (diverging) or collapsing (converging). A third possibility is to measure the average velocity norm of all particles. This measures the dynamics of the swarm in a slightly different way. The velocity of the particles is the core of the PSO algorithm. By measuring the norm of the velocity, in essence, the ability of the particles to move around is being assessed. The difference between two such measures quantifies the exploratory and exploitive behaviour of the swarm. If the difference is positive, the swarm has a tendency to explore, if the difference is negative, the swarm tends to exploit. For the results presented here we use the average velocity norm as our measure of swarm diversity.

The change in the metric value between iterations provides a feedback signal to update the parameter values. The feedback signal is given by:

$$\Delta S = S(t+1) - S(t). \tag{4.1}$$

where $S(t)$ is the mean velocity norm of the swarm's particles at iteration t. Following each iteration we update the parameters of the PSO algorithm using

$$\omega(t+1) = \omega(t) - \varepsilon f(\Delta S). \tag{4.2}$$

$$\alpha_1(t+1) = \alpha_1(t) - \varepsilon f(\Delta S). \tag{4.3}$$

$$\alpha_2(t+1) = \alpha_2(t) - \varepsilon f(\Delta S). \tag{4.4}$$

Equations 4.2 through 4.4 ensure that each parameter is increased or decreased in the same manner, and by the same amount. The swarm's parameters are moved along a gradient either out from the origin of $\omega : \alpha$ parameter space or in toward it. Moving outward, results in the swarm becoming more likely to diverge. Whilst inward movement in parameter space makes the swarm more likely to converge. The locus of critical parameter values is a continuous curve in this space, we can move inward/outward in this somewhat arbitrary direction and be assured to cross the critical curve at some point. As each parameter is being treated the same we introduce θ to

refer to each and all the parameters (ω, α_1, α_2). Thus our update is

$$\theta(t+1) = \theta(t) - \varepsilon f(\Delta S), \tag{4.5}$$

The function $f(\Delta S)$ is used to scale ΔS broadly to the scale of the parameters. Here we use a sigmoid function scaled by the problem space (though linear functions and step functions have also been successfully used). This ensures that small changes in swarm size have less impact on than large changes. The parameter $\varepsilon \leq 1$ controls the size of parameter updates. The rescaled sigmoid, we use here, is given by

$$f(\Delta S) = \tanh\left(\frac{\Delta S}{2\sigma}\right), \tag{4.6}$$

where σ is a scaling factor that is related to the extension of the search space in units of ΔS. For our experiments we used the mean velocity norm of the particles as our metric S, our σ is chosen to be less than or equal to the maximum extent of the problem space so that the sigmoid (Eq. 4.6) returns either -1 or 1 for swarms where the mean velocity norm exceeds the extent of the problem space per iteration.

Having selected the initial parameters (ω, α_1 and α_2) values, the update rule, Eq 4.5, will apply the same magnitude change to each parameter on each update. Recalling the PSO stability curves plotted on the parameter plane (Figure 3.5), we can see that this will cause the parameters to move along a line approximately outward from, or inward toward, the origin. So long as the parameters spend some time outside the line we should be assured that the swarm spends some time being divergent, and thus will avoid stagnation. Similarly, as long as it spends time within the line, it will potentially be performing exploitation near known good locations. Empirically we saw earlier that best performance was to be found inside the line, but that there was no clear *best* parameter pair, so we have no reason to believe that we need to make more complex deviations around the parameter plane to cross the stability line in more than one location. The behaviour of the parameters under this regime is show in section 4.4.2.

CriPS is shown in Algorithm 1.

Algorithm 1: CriPS algorithm.

Input: Objective function F(), Maximum iterations I, Target Fitness value V

Output: Returns location of best evaluation of F() achieved in I iterations

Initialise:

minSize $\leftarrow$ lower limit of F()'s dimension in each dimension;

maxSize $\leftarrow$ upper limit of F()'s dimension in each dimension;

particle positions X $\leftarrow$ uniform random [minSize, maxSize];

particle velocities V $\leftarrow$ uniform random [minSize, maxSize] per iteration;

PSO parameter $\omega \leftarrow 0.815$;

PSO parameter $\alpha_1 \leftarrow 1$;

PSO parameter $\alpha_2 \leftarrow 1$;

CriPS parameter $\varepsilon \leftarrow 1$;

CriPS parameter $\sigma \leftarrow$ maxSize - minSize;

calculate initial mean velocity norm S;

while *Termination conditions are not met* **do**

  Standard PSO:

  do standard PSO velocity update;

  do standard PSO position update;

  calculate new fitness values F(X);

  update personal and global bests

  CriPS additions:

  calculate new mean velocity norm S';

  $\Delta S \leftarrow$ S' - S;

  foreach *PSO parameter* Θ **do**

    $\Delta\Theta \leftarrow \varepsilon \times \tanh\left(\frac{\Delta S}{2\sigma}\right)$;

    $\Theta \leftarrow \Theta - \Delta(\Theta)$;

  end

  S $\leftarrow$ S';

end

4.3.2 Discussion of the CriPS algorithm in action

We can consider the effect of our metric feedback on the behaviour of our swarm.

4.3.2.1 Simplified velocity update

Consider a simplification of the PSO velocity update rule where the particles in the swarm share the same personal and global best fitness values. This is shown in equation 4.7, where α is twice the normal α_1 or α_2 values.

$$\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + \alpha\mathbf{R}(\mathbf{g} - \mathbf{x}_i) \tag{4.7}$$

For a well evolved swarm this simplification may be true, however even if this is not the case the particles are each being drawn to some compromise position between personal and global best fitness locations. These compromise positions are all within the problem space of interest, whereas the CriPS swarm, not being spatially constrained, can be outwith this space. The simplification should not materially change the following discussion. The new particle velocity is therefore the sum of an inertial term ($\omega\mathbf{v}_i(t)$), which is a vector in the direction of the current particle velocity, and a vector that always points towards the location that is the currently best (or compromise) location found.

4.3.2.2 The Mean Velocity Norm Metric

The algorithm uses a metric to inform us whether our swarm in increasing or decreasing spatially. An increase in this metric indicates that on average the particles in our swarm are accelerating. The rate of expansion or contraction will be increasing or decreasing depending on the change in metric signal. If the swarm is shrinking, but the particles are accelerating, then our positive signal will tend to indicate that in the next few iterations the particles will zoom past the swarm centroid and resume a swarm expansion. If the metric is decreasing then the velocity term will tend toward zero. The remaining term, the pull toward the global best will remain non-zero, so the tendency is for this to represent a shrinking swarm.

We note that only in the case of a zero size swarm with all particles at rest would the metric also be zero and the swarm would be static.

4.3.2.3 The Application of Metric and its Resultant Contribution to the Swarms' Dynamics

The feedback signal is mapped to the range [-1, 1] via a squashing function. A sigmoid function is used, although the precise function does not seem important. The scaling

to the interval [-1, 1] is used only to make the changes to parameter values occur at or below the size scale of the parameters.

Excluding the state where the swarm is of zero size and at rest there are (broadly) four categories of behaviour that can be considered arising from the combinations of whether the swarm is expanding or contracting and whether the swarms' mean velocity norm is increasing or decreasing (equivalent to its particles, on average, accelerating or decelerating).

1. Swarm expanding, particles accelerating. Particle acceleration means that we have a positive feedback signal ($\varepsilon f(\Delta S)$). This is subtracted from the parameters, ω and α. Reducing ω will result in the inertial terms of the update tend towards zero. However, whilst the α value is also reduced, the fact that in an expanding swarm the distance from the global best location and particle locations will tend to be increasing means that the next velocity update will increasingly favour the pull toward the global best. Thus the swarm will tend to start shrinking again.

2. Swarm expanding, particles decelerating. Particle deceleration means that we have a negative feedback signal ($\varepsilon f(\Delta S)$). This is subtracted from the parameters, ω and α. Increasing ω will tend to try to increase the velocity. Increasing α and the fact that in an expanding swarm the distance from the global best location and particle locations will tend to be increasing means pull toward the global best should increase faster. As the direction of the velocity is pulled toward the location of the global best, the increasing ω value will then act to further this.

3. Swarm shrinking, particles accelerating. Here all terms will be reducing: both parameters and the distance between the global best location and the particle locations. What pull there is will tend toward the global best location.

4. Swarm shrinking, particles decelerating. The $\mathbf{g} - \mathbf{x}_i$ part is decreasing due to the swarm shrinkage, but both parameters will be increasing. The contribution to the velocity update will increasingly be from the inertial component.

4.3.3 Evaluation

To evaluate the behaviour of the CriPS controlled swarm, a comparison between this algorithm and others across a range of typical problems is required. It is also of interest to examine the CriPS swarm's ability to avoid stagnation and to what extent

it mixes exploratory and exploitative behaviours. CriPS is compared with our own implementation of a standard PSO over a range of objective functions. It is also important that the algorithm can be shown to perform reasonably across a range of scenarios to encourage the idea that it may be useful to further develop the algorithm. Further comparisons are made with other PSO variants' published results (Bratton and Kennedy, 2007; CEC2013; Fernandes et al., 2012). The functions used in these comparisons are defined over finite spatial domains. CriPS is does not constrain the positions or velocities of its particles so test objective functions are modified to return poor cost function values outside the spatial ranges they are normally defined for. For example an N dimensional Schwefel function returns a fitness value given by

$$f(x) = \begin{cases} 418.9829N - \sum_{i=1}^{N} x_i \sin(\sqrt{|x_i|}), & x_i \in [-500, 500], \forall i. \\ 1.7977 \times 10^{308}, & otherwise. \end{cases} \tag{4.8}$$

This ensures that the PSO algorithm is free to explore anywhere within the N dimensional space, but should only find prospective optima within the region of interest (NB the value 1.7977×10^{308}, in Eq 4.8, is the maximum real number in Matlab) The intent of our algorithm is that particles will spend part of their time looking for solutions near to known good locations and part of their time searching farther afield, perhaps in areas as yet under-explored. These modes can be thought of as exploitation and exploration of the problem. In reality there in a continuum of possibilities from one extreme to the other. It is likely that the 'best' mix of behaviours could be some function of the topology of the problem space. We are looking to engineer a dynamic that naturally achieves a mix of behaviours through the swarm's interaction with the problem space. Testing is performed with objective functions with known minima. Real world functions have unknown best solutions so no assumption that our swarm has found the optima can be made. The swarm should therefore to continue to explore all of the problem space (at least sometimes). To characterise the swarm's exploration the spatial diversity of its particles can be used. We calculate the mean swarm distance (MSD) from the swarm centroid as a measure of this. The larger the MSD the more spread out in the problem space our particles are.

Algorithm comparisons were performed setting initial PSO parameter values of $\omega = 0.815$, $\alpha_1 = 1.0$, $\alpha_2 = 1.0$. These are deliberately poor choices when used with standard PSO for the problem functions we are using. The intent was that CriPS should gain no benefit from starting with good parameter values. As stated above we use the

mean velocity norm metric. CriPS introduces two new parameters (ε and σ).

To demonstrate the behaviour of the CriPS swarm we will vary these new parameters and explore their effect on the dynamics of the algorithm. In Section 4.4.1 the algorithm is shown to exhibit a *bursting* dynamic. The swarm remains small for extended periods of time, before showing dramatic increases of size before shrinking again. The scale of of these changes in size is greater than the problem domains. Section 4.4.3 examines whether the swarm size changes follow power-law distributions that would indicate critical behaviour.

For the comparison studies (Sections 4.4.4 onwards) we do not wish to introduce an additional selection burden, so we set $\varepsilon = 1$ and σ to equal the maximum distance in any one dimension of the problem space. For the domain of definition of the Schwefel function ($[-500, 500]^N$), for example, this is 1000. Sections 4.4.4 through 4.4.6 present the results of these comparisons.

4.4 Results

Below we show that the CriPS modified dynamics of the PSO swarm leads to a bursting behaviour that ensures continued exploration of the problem space. The dynamics are such that there is a mixture of swarm size changes (see Sec 4.4.1, the frequency of which is somewhat power-law distributed (see Sec 4.4.3): with many small exploitative changes, and fewer large exploratory moves. This shows that the quasi-random changes of the size of the swarm follow a somewhat critical distribution which implies a search across all length scales. In comparison with a basic standard PSO approach our algorithm performs better across a range of objective functions both in terms of obtaining better fitness values and in finding improvements more frequently. Finally comparisons are made with results reported in the literature.

4.4.1 Coverage of search space

The CriPS algorithm introduces two new parameters. Although we would wish not to be burdened by setting these, we start by exploring the impact they have on the behaviour of our swarm. Different values of ε and σ will modify the behaviour of the swarm. They both affect the size of feedback signal generated on each iteration. Therefore we fixed σ to 200 (1/5 of the size of the Schwefel function's problem space) and ε to various values to tune the mix of exploration and exploitation. The swarm has

25 particles.

We calculate the mean swarm distance (MSD) of the particles in the swarm from the centroid of the swarm at each iteration. For the Schwefel objective function used here our problem space is of size 1000 in each of 20 dimensions, the maximum distance across our problem space is $\sqrt{20} \times 1000 \approx 4472$. Figure 4.1 shows the time evolution of the mean swarm distance from the swarm centroid. It is obvious that the swarm can become larger than the problem size, allowing all locations to be reached. We also see that the swarm spends larger periods of time in a compact form than in an expanded form. This is equivalent to saying that the swarm spends more time looking for better solutions near the known good locations than it spends exploring the rest of the problem space. The dynamics of the swarm is nevertheless complex as different particles explore or exploit the space in different ways, similarly the balance of these behaviours is also different in different dimensions. The dimension- and particle-specific dynamics are not synchronised as shown in the plots in Fig. 4.2.

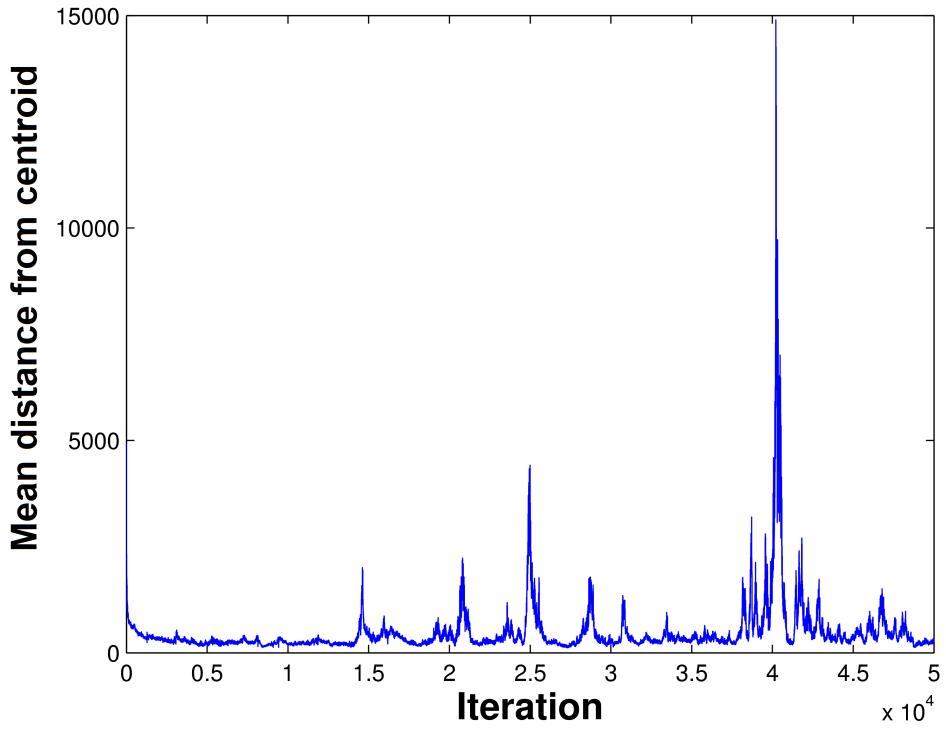
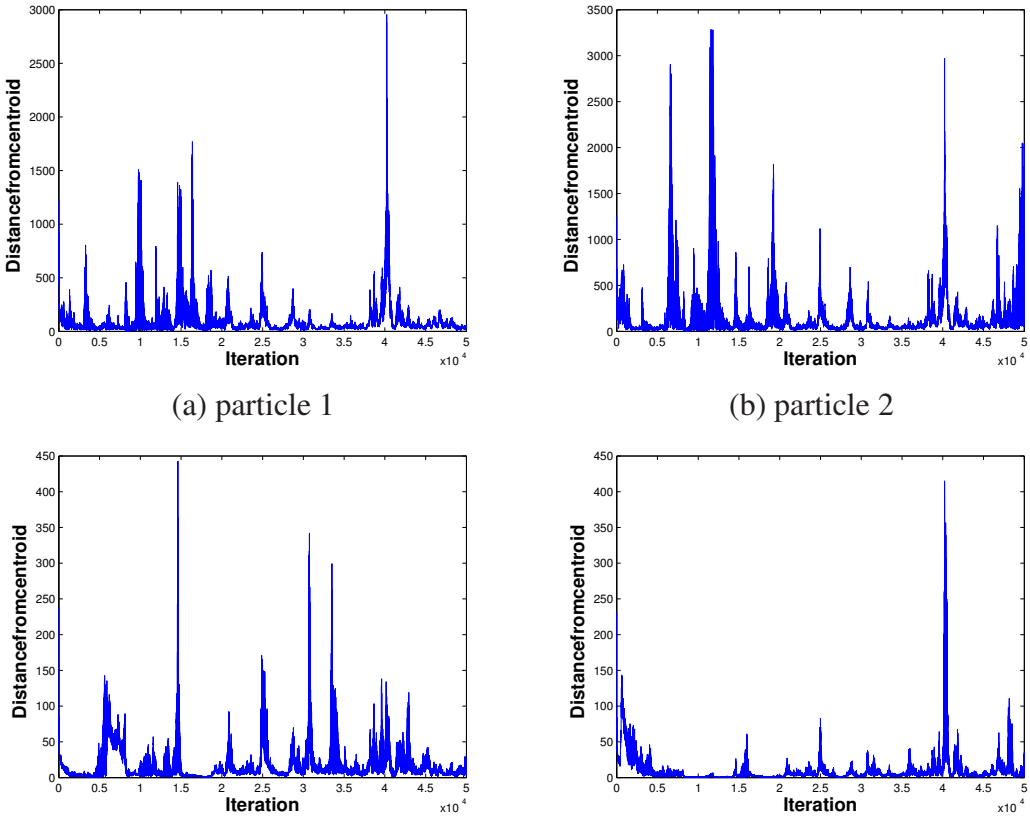


Figure 4.1: Mean distance of particles from the swarm's centroid for Schwefel objective function. The swarm had 25 particles. The swarm shows a bursting type behaviour. Maximum swarm size can be seen to be larger than the size of the problem's region of interest. Here our algorithm used $\varepsilon = 0.15$ and σ set to $\frac{1}{5}$ of the problem space's size.



(a) particle 1

(b) particle 2

(c) dimension 1

(d) dimension 2

Figure 4.2: Swarm dynamic in detail. The top row shows two different particles in our swarm. The distance of each particle from the swarm's centroid is shown. Particle behaviours are similar but not synchronised. The lower plots show the dynamics of the mean swarm distance in two of the twenty dimensions. Again the behaviour in each dimension is similar but unsynchronised. Here our algorithm used $\varepsilon = 0.15$ and σ set to $\frac{1}{5}$ of the problem space's size.

4.4.2 CriPS's effect on parameter distributions

The CriPS algorithm modifies the PSO parameters on each iteration. For a converging swarm it increases the parameters, for a diverging swarm it decreases the parameters. On each iteration a change to the parameters is calculated and applied equally to all of the PSO parameters (ω, α_1 and α_2). The algorithm's parameters thus move along a straight line in the system's parameter space. Chapter 3 showed that there is a locus of parameter pairings (ω and $\alpha = \alpha_1 + \alpha_2$) that result in optimal behaviour. CriPS should ensure that parameter values spend some time allowing PSO to behave optimally to have any chance of performing well. Figure 4.3 shows the set of parameter values used by CriPS when solving a Schwefel function under the same conditions as section 4.4.3. We plot this against the stability curves presented in chapter 3, so we can see that the

parameter values cross these lines. Figure 4.4 shows the distributions of the ω and α values. The peaks of these distributions lie near the curve of stability so is consistent with optimal behaviour.

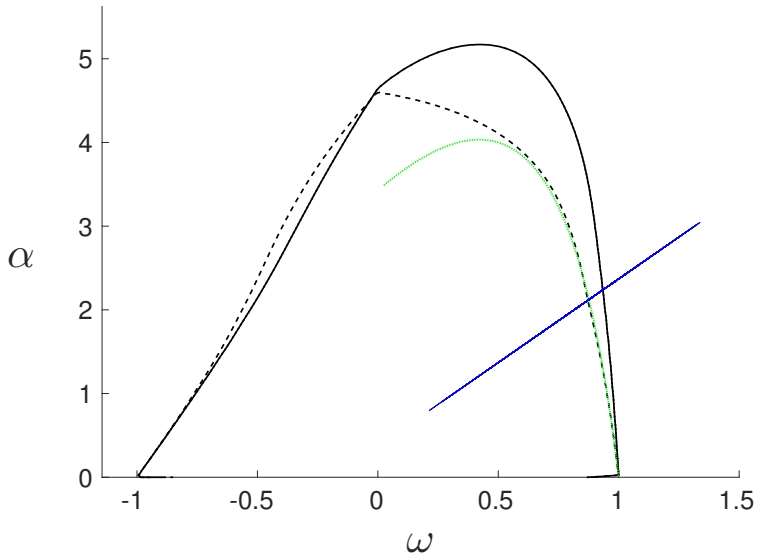


Figure 4.3: Example of CriPS parameter values. CriPS was run using the swarm conditions used to assess system criticality (Section 4.4.3). The blue line shows the locus of parameter values in the system's parameter used by CriPS throughout the run. The stability curves of chapter 3 are also displayed.

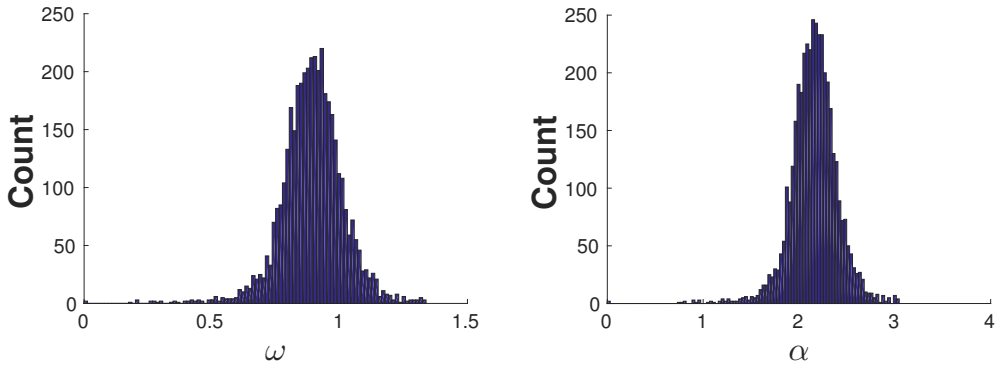


Figure 4.4: Distributions of CriPS parameter values. CriPS was run using the swarm conditions used to assess system criticality (Section 4.4.3). The distribution ω (left) and α (right) parameter values are shown as histograms.

4.4.3 Assessment of criticality

One feature in critical systems is the presence of heavy tail distributions in certain measures of those systems. True power-laws may only be apparent in infinite systems.

Where systems are finite the distribution of events will be truncated at scales near the system size. Thus to investigate the presence of power-laws in our swarm requires the study of larger PSO systems. We increased the number of particles in the swarm to 250 and studied the dynamics for 50000 iterations. All executions located the target minima to within a small error (< 0.001). In the previous section we saw that swarm diversity showed variations between occasional large extensions and more frequent periods of small extent. We can explore this by viewing the distribution of changes in swarm size between iterations. Our diversity is measured by the mean swarm distance from the swarm centroid (MSD), so we can plot the frequency of changes in this (ΔMSD). Figure 4.5 (top left) shows this on a log-log plot. Smaller ΔMSDs occur more frequently than larger ΔMSDs. This figure gives limited evidence for a power-law of the form

$$y \approx 83000\, x^{-2.3}. \tag{4.9}$$

The plot shows deviations from a potential power law at both small and large event sizes. This is not necessarily unusual. The process of building the histogram for this plot places data into bins. At the upper end of the distribution these bins containing low incidence, rare events, tend to make the data appear noisy. One means of cleaning this view is to plot the data a cumulative distribution plot (Newman, 2005). This can make it easier to fit a straight line. We choose not to do this as our distribution is unlikely to be a true power law. One reason to say this is that it is often required that the distribution behaves linearly in the log-log plot for at least two decades which is not reached here. Chapter 3 showed that if α and ω sat on the curve of stability then the system behaved critically. As CriPS, on each iteration, makes additive changes to these parameters the system is continually moved from super- to sub- critical regions (and back). The system is not therefore truly critical and won't show a true power law. The apparent straight line in figure4.5 (top left) is likely to only represent a pseudo-criticality, in the sense that it averages to a critical-like behaviour (de Andrade Costa et al., 2015). However, we are interested here in the practical effect of this mix of explorative and exploitative moves and whether they are benefiting our algorithm.

If the straight line represents a true power-law relationship then we should suspect that it is possible to tune the system via a parameter variation from subcritical, through critical, to supercritical behaviours. The starting values of the parameters ω, α_1, and α_2 appear to make little difference to the behaviour of the algorithm. This is to

be expected as the algorithm is itself modifying these parameters. Instead we look at the ε parameter. Large values ($\varepsilon = 0.5$) appear to result in a subcritical swarm with a shortage of large swarm movements, whilst a small value ($\varepsilon = 0.075$) results in an excess of large moves making the swarm appear supercritical. Finally the equivalent plot for the standard PSO algorithm , shows that with the parameters used the PSO swarm makes many large moves, favouring exploration over exploitation. Different values for the parameters selected for the standard PSO may have improved its performance. Fig. 4.5 shows these variants.

4.4.4 Comparison with PSO

The previous sections explored the nature of the CriPS algorithm. We showed that stagnation is avoided and a mixture of swarm behaviours is achieved. In this section we compare the performance of this algorithm with, first, our own PSO implementation, and second, with the results of a number of PSO variants in Bratton and Kennedy (2007)

In section 4.4.5 we will compare with other criticality inspired PSO variants. Finally in section 4.4.6 we compare performance with a basket of metaheuristic algorithms competing in a recent competition.

In each case we will endeavour to adopt the testing protocol used in the comparator study.

4.4.4.1 CriPS versus PSO implementation

Our first comparison is made with our own implementation of a standard PSO algorithm we use a range of problem functions : Sphere, Generalized Schwefel, Generalized Rastrigan, Ackley and Generalized Griewank. We take these object functions and their definitions from Bratton and Kennedy (2007)

Some algorithms can show a bias toward minima that occur at the problem space's origin or can be influenced by the initialisation positions of the particles (Monson and Seppi, 2005). We therefore adopt the initialisation scheme from the same paper and problem definitions i.e. 30 dimensional problems, 30 particles, algorithm run 30 times for 10000 iterations.

For parameter selection for the PSO algorithm we choose two sets: $\omega = 0.7$, $\alpha_1 = \alpha_2 = 2$ and $\omega = 0.815$, $\alpha_1 = \alpha_2 = 1$. The former set places a stronger weighting on the attraction toward the learnt good locations, the latter set uses a larger inertial term

weighting. The latter selection also matches the initial setting of parameters for our CriPS algorithm.

Table 4.1 lists the mean fitness values and standard deviations obtained. Table 4.2 shows the results of statistical testing of these results. Our approach performs better in all cases, except for a single test against the Sphere function, when it behaves as well as the relevant PSO algorithm.

Function	**PSO** ($\omega = 0.7, \alpha_1 = \alpha_2 = 2$)	**PSO** ($\omega = 0.815, \alpha_1 = \alpha_2 = 1$)	**CriPS**
Sphere	7111	0.00031	0.00044
	(12444)	(0.00016)	(0.0011)
Schwefel	5061	4072	2884
	(1411)	(253)	(522.1)
Rastrigan	204.31	185.33	34.07
	(67.53)	(37.168)	(23.1707)
Ackley	20.23	19.794	19.6452
	(0.6614)	(0.0585)	(0.2503)
Griewank	94.08	1.262	0.02486
	(181.1)	(5.6732)	(0.0288)

Table 4.1: Comparison of CriPS algorithm with our implementation of PSO. The mean fitness and standard deviations (in parentheses) from 30 runs against each of the objective figures for each algorithm is shown. In all cases the problems are defined as 30 dimensional, there are 30 particles in the swarm, and each run is executed for 10000 iterations. The particles are initialised in the same volume as defined in the Bratton and Kennedy (2007) paper. Here our algorithm used ε = 1 and σ set to the size of the problem space. All objective functions return a value of zero at their global optima.

The PSO algorithm, with $\omega = 0.815$, $\alpha_1 = \alpha_2 = 1$ has the better results of the two PSO settings. We use this to perform extended tests. These executed the algorithms against the Schwefel function using an iteration budget of 50000. This was repeated 30 times. The CriPS algorithm achieved a mean of 2470.5 and standard deviation of 506.8. PSO achieved a mean of 4071.4 and standard deviation of 252.5. However, we can look at how often improvements were located by each algorithm within each 10000 iteration segment of the run. The results, shown in tables 4.3 and 4.4 show that whilst the frequency of locating improvements falls with increasing iterations our algorithm continues to locate better solution more frequently.

CriPS vs Algorithm	Sphere	Schwefel	Rastrigan	Ackley	Griewank
PSO ($\omega = 0.7, \alpha_1 = \alpha_2 = 2$)	+	+	+	+	+
PSO ($\omega = 0.815, \alpha_1 = \alpha_2 = 1$)	0	+	+	0	+

Table 4.2: Statistical comparison of CriPS algorithm's results with PSO. The PSO algorithm was run with two sets of parameters, as indicated. Better, or similar performance is indicated by $+$, or 0 respectively. We use pairwise one sided Welch's t-test to make the comparison using a 5% significance level.

4.4.4.2 CriPS versus other PSO variants

Here we compare the performance of CriPS with the PSO variants used in Bratton and Kennedy (2007). This compares three variants of the PSO over 14 objective functions. For this initial exploration we select a subset of five functions, matching those studied in the previous section, namely: Sphere, Generalized Schwefel, Generalized Rastrigan, Ackley and Generalized Griewank. Experimental setup is as before: functions are 30 dimensional, each is executed 30 times, swarm size is 30 particles, execution duration is 10000 iterations. Our algorithm utilises the same particle initialisation as before: $\omega = 0.815$, α_1=α_2=1, ε=1, and $\sigma =$ one dimensional size of the problem space.

Table 4.5 shows the results obtained by the CriPS algorithm in comparison to those reported by (Bratton and Kennedy, 2007). We perform one-sided pairwise Welch's T tests on the data to test for significant differences. The results of this statistical comparison is presented in table 4.6. The CriPS algorithm outperforms the original PSO algorithm for all functions. It outperforms the Gbest PSO against 3 of the 5 functions, and outperforms the LBest PSO in 2 of the 5 functions.

4.4.5 CriPS compared to other criticality inspired PSO variants

We also compare with the approach given by Fernandes et al. (2012). In their paper they present a Bak-Sneppen criticality augmented PSO (BS-PSO) algorithm, which they compare with several earlier algorithms. The BS-PSO algorithm uses the Bak-Sneppen algorithm to generate power-law distributed numbers. These are used to vary either the ω parameter on its own or together with the α_1 and α_2 parameters. The latter approach generally gave rise to their best results. Additionally they used the

Iteration range	**PSO** ($\omega = 0.815, \alpha_1 = \alpha_2 = 1$)	**CriPS**
0-10000	2255.1 (342.1)	2499.6 (366.5)
10001-20000	4.1 (9.5)	1864.3 (921.4)
20001-30000	1.1 (0.55)	217.7 (449.7)
30001-40000	10.1 (45.8)	140.4 (540.2)
40001-50000	0 (0)	20.3 (82.4)

Table 4.3: Number of improvements located by CriPS versus simple PSO implementation. Each algorithm is run 30 times, for 50000 iterations. The mean and standard deviation (in parentheses) of the number of times an improved fitness is located was counted within each 10000 iteration period. In all cases the problems are defined as 30 dimensional, there are 30 particles in the swarm. The particles are initialised in the same volume as defined in the Bratton and Kennedy (2007) paper. Here our algorithm used $\varepsilon = 1$ and σ set to the size of the problem space.

same distribution to drive additional local search. We compare our CriPS algorithm to their non-local search variant of BS-PSO. Their test conditions were to use 30 dimensional objective functions (Sphere, Rosenbrock, Rastrigan and Griewank) and a two dimensional function (Schaffer). Following this comparator paper's protocol 20 particles were used for 3000 iterations (1000 iterations for the Schaffer function). Each objective function was tested 50 times. Particle initialisation and volumes over which the problems are defined are as specified in Fernandes et al. (2012). Table 4.7 shows the results of these trials. Table 4.8 shows the statistical comparison between the two algorithms. Our algorithm outperforms the BS-PSO algorithm in 2 of the 5 functions. Additionally the Rosenbrock function shows no statistical difference in performance between the algorithms.

4.4.6 CEC2013 comparisons

The IEEE Congress on Evolutionary Computation conference regularly holds competitions for metaheuristic algorithms used to solve optimisation problems. A recent competition (CEC 2013) required algorithms to run against 28 different benchmark functions designed to provide a wide ranging test of the capabilities of the algorithms. Competitors were required to execute 51 runs against each function

CriPS vs Algorithm	**0-10000**	**10001-20000**	**20001-30000**	**30001-40000**	**40001-50000**
PSO ($\omega = 0.815$, $\alpha_1 = \alpha_2 = 1$)	+	+	+	0	0

Table 4.4: Statistical comparison of CriPS algorithm's results with PSO. The number of fitness improvements located by each algorithm is compared. Better, or similar performance is indicated by $+$, or 0 respectively. We use pairwise one sided Welch's t-test to make the comparison with a 5% significance level. NB This test is likely inappropriate for the final two results as it assumes distributions are largely normal which is not true for these.

in each of 10, 30 and 50 dimensions. Each run was limited to a maximum number of evaluations of the objective function being solved (the limit being 10000 multiplied by the number of dimensions). The full competition looked at computational efficiency and the speed with which improving fitnesses were obtained. Here we make a comparison of CriPS with the algorithms in the competition purely on the basis of final fitness values achieved. As before we set parameters to $\omega = 0.815$, α_1=α_2=1, ε=1, and $\sigma =$ one dimensional size of the problem space. CEC 2013 performed a ranking exercise using the individual results for every function and algorithm. This allowed the 21 algorithms to be ordered from best to worst. The published results for the individual algorithms only present the summary statistics (mean and standard deviations) for each algorithm. Thus we base our comparisons only on the reported mean values. We took the algorithms that finished in positions 1, 5, 10, 15, 17 and 21. For each function:dimension pair the mean fitness achieved of these six algorithms and the CriPS algorithm are ranked, best to worst. We then take the average ranking over the 84 results (28 functions run in each of the three dimensionalities). The aim is to provide an approximation of where the CriPS algorithm may have placed in the CEC2013 competitions. Table 4.9 summarises the ranking result. The individual results are presented in appendix C. Our estimate of the capability of the CriPS algorithm is that it would be placed roughly between those algorithms that finished 15th and 17th in the CEC2013 competition. Functions 1 through 7 of this competition's challenges are relatively simple functions where simple gradient descent may be a useful approach. As our algorithm makes no use of such approaches it does poorly by comparison with these functions. Individual runs may result in a good solution being found: CriPS

Function	PSO	Gbest PSO	Lbest PSO	CriPS
Sphere	2.7562	0.0	0.0	0.00045
	$(0.0448)^1$	$(0.0)^1$	$(0.0)^1$	$(0.00114)^2$
Schwefel	4211	3508	3360	2884
	$(44)^1$	$(33)^1$	$(34)^1$	$(522.1)^2$
Rastrigan	400.7194	140.4876	144.8155	34.0739
	$(4.2981)^1$	$(4.8538)^1$	$(4.4066)^1$	$(23.1707)^2$
Ackley	20.2769	17.6628	17.5891	19.6452
	$(0.0082)^1$	$(1.0232)^1$	$(1.0264)^1$	$(0.25027)^2$
Griewank	1.0111	0.0308	0.0009	0.02386
	$(0.0031)^1$	$(0.0063)^1$	$(0.0005)^1$	$(0.02879)^2$

Table 4.5: Comparison of CriPS algorithm with those reported in (Bratton and Kennedy, 2007). Each algorithm reports the mean fitness from 30 runs against each of the objective figures. Additionally the [1]standard errors or [2]standard deviations are reported in parentheses. In all cases the problems are defined as 30 dimensional, there are 30 particles in the swarm, and each run is executed for 10000 iterations. The particles are initialised in the same volume as defined in the Bratton and Kennedy (2007) paper. Here our algorithm used $\varepsilon = 1$ and σ set to the size of the problem space.

CriPS vs Algorithm	Sphere	Schwefel	Rastrigan	Ackley	Griewank
PSO	+	+	+	+	+
Gbest PSO	-	+	+	-	+
Lbest PSO	-	+	+	-	-

Table 4.6: Statistical comparison of CriPS algorithm's results with those reported in (Bratton and Kennedy, 2007). Better, or worse performance is indicated by $+$, or $-$ respectively. We use pairwise one sided Welch's t-test to make the comparison with a 5% significance level.

frequently solving the Sphere function with solutions decades below the considered zero of 10^{-8}. An occasional failure results in a much larger mean solution. This derives from the algorithm exploding out to explore the problem space when logic would suggest that simply exploiting gradient descent would be the best course. We could tune the parameters to reduce the bursting of the swarm and improve its performance, but the aim was to be parameter free. Removing the simpler functions these from the test set elevates CriPS into the 3rd quartile of results. We also note that for function 8 CriPS appears to win.

Function	best non-local search BS-PSO	CriPS
Sphere	0.0	0.0030
	(0.0)	(0.0052)
Rosenbrock	85.6	73.7832
	(79.8)	(118.864)
Rastrigan	202	173.5997
	(41.6)	(39.4008)
Griewank	0.0165	0.03978
	(0.0224)	(0.0485)
Schaffer	0.00555	0.0
	(0.0048)	(0.0)

Table 4.7: Comparison of CriPS algorithm with those reported in (Fernandes et al., 2012). Each algorithm reports the mean fitness from 50 runs against each of the objective figures. Additionally standard deviations are reported in parentheses. In all cases the problems are defined as 30 dimensional (except the two dimensional Schaffer function), there are 20 particles in the swarm, and each run is executed for 3000 iterations. The particles are initialised in the same volume as defined in the Fernandes et al. (2012) paper. Here our algorithm used $\varepsilon = 1$ and σ set to the size of the problem space.

4.5 Discussion

The CriPS algorithm employs feedback from the swarm's exploration of a problem space to modify the standard PSO's parameters. Interesting swarm dynamics were explored. Comparisons with standard PSO and leading metaheuristics have been conducted. We discuss this below and speculate on future directions.

CriPS vs Algorithm	Sphere	Rosenbrock	Rastrigan	Griewank	Schaffer
best non-local search BS-PSO	-	0	+	-	+

Table 4.8: Statistical comparison of CriPS algorithm's results with those reported in (Fernandes et al., 2012). Better, worse or similar performance is indicated by $+$, $-$ or 0 respectively. We use the pairwise one sided Welch's t-test to make the comparison with a 5% significance level.

Algorithm	**CEC 2013 position**	**Average ranking**
NBIPOP-aCMA-ES (Loshchilov, 2013)	1	1.95
NIPOP-aCMA-ES (Loshchilov, 2013)	5	2.35
CDE:b6e6rl (Tvrdík and Polakova, 2013)	10	2.88
JANDE (Elsayed et al., 2013b)	15	4.86
CriPS	n/a	4.96
TPC-GA (Elsayed et al., 2013a)	17	5.51
PLES (Papa and Silc, 2013)	21	5.50

Table 4.9: Summary of the comparison of CriPS algorithm with CEC 2013 competition algorithms. Algorithms placed 1, 5, 10, 15, 17, 21 in the competition form the basis of this comparison. The mean results of each of these algorithms are ranked with CriPS's results across all 84 function:dimensionality pairings. The rankings are averaged above.

4.5.1 On CriPS's observed dynamics

The CriPS algorithm can be tuned via its ε parameter to operate in a somewhat critical manner. Given that we know critical behaviour naturally arises in PSO via its iterated stochastic updates, this implies that the parameter update mechanism need not undermine this behaviour. With the correct setting, the swarm is able to explore the full region of interest of a test function. The extension of the swarm can vary with a power-law distribution. As the standard PSO element of the algorithm draws the swarm toward the current best locations these size changes assure that areas of the problem space near known good values are well exploited. The long tail of the power-law distribution will ensure that exploration of the full problem space occurs. These two competing mechanisms stop the algorithm from stagnating.

Our studies of our algorithm applied to the various test functions suggest that it is insensitive to the initial settings of the ω, α_1, and α_2. This is expected since the algorithm is continually modifying the parameter values throughout its execution. The range over which this insensitivity extends shall be explored in future work. The important parameter to set is ε. Varying this parameter tunes the behaviour of the swarm from subcritical to supercritical behaviours. For problems where the optimum values are unknown we believe that detection of critical behaviour should be easier to achieve than correctly setting the ω, α_1, and α_2 parameters. Such a meta-process should self adjust the ε value to obtain a good swarm response to the function rather than trying to detect good results. Changes to the PSO parameters are derived from the size of the changes in the swarm metric being employed and the size of ε. Large

ε values will therefore tend to amplify any changes in the swarm metric and therefore larger changes to the PSO's parameters will occur at each iteration. Thus if the swarm is expanding, a large ε will tend to quickly reduce parameters (notably ω seems most important) and the swarm's expansion will be quickly reversed. A shrinking swarm will likewise have its dynamic swiftly reversed. Thus a large ε leads to a swarm where many small changes occur, but few large changes. Such a swarm fails to show the large diversity needed to avoid stagnation. The counter case of a small ε leads to a swarm with a great many large excursions, often much larger than the problem space of interest. Whilst stagnation will be avoided, fewer small movements are made, so there is less opportunity to improve on solutions already located. Any meta-process thus needs to monitor the frequency of large swarm metric changes that correspond to movements larger than the defined problem space. If many are big, then ε should be increased. If too low (i.e. there is little chance of exploring the problem space) then the parameter should be reduced.

We have reviewed the capabilities of the algorithm largely in the terms of the total number of particle iterations. The algorithm, because of its use of a calculated swarm metric and its use of a squashing function, has an additional computational overhead than standard PSO. It is not clear if the squashing function is needed. Both linear functions and step functions were briefly investigated and appear to produce similar results. The standard PSO's use of random stochastic variables in the velocity update was included in our algorithm. In chapter 3 we saw that the product of these stochastic matrices led to potential critical behaviour of the PSO swarm for specific parameter values. CriPS's behavioural mix is further modified by online parameter changes driven from a feedback signal derived from the swarm's interaction with the problem space. Future work should look at whether including both these processes is counter productive.

The CriPS algorithm, with its ε parameter set to ensure critical behaviour showed an automatic balancing of swarm size changes over all scales of the problem space. Larger, exploratory moves are less frequent than smaller, exploitative moves. Stagnation is avoided resulting in improving results the longer the algorithm is run. The mechanism that gives rise to this links the interaction of the swarm with its problem environment to modify of the algorithm's own parameters. The swarm shows occasional bursts in size that provide the potential to escape from any local minimum that the swarm may have become becalmed in. In this way the algorithm is tuning itself to the problem space. This ability to self-tune the parameters of our algorithm should

then be demonstrated across a much wider range of test and real world functions in the future.

4.5.2 On our comparative studies

We performed comparative studies with other PSO algorithms. CriPS performed well against simple PSO variants, but more modestly against leading metaheuristic algorithms. Whilst CriPS overcomes the problems of stagnation there is nothing in the algorithm shaping the evolution of the swarm towards better solutions beyond the original PSO algorithm dynamic. The best algorithms in the CEC2013 combination used a combination of evolutionary strategies (ES) and covariance matrix analysis (CMA) to guide their swarms towards improved solutions. CMA creates distributions of future particle candidate solutions based on known good solutions. New particle locations are selected from these distributions. The evolutionary part narrows these distributions over time. Whilst CMA-ES still suffers stagnations, restart and swarm growth strategies have proved successful strategies.

4.5.3 On benchmarking and testing

The CEC2013 competition consists of 28 functions designed to test the capabilities of the competing algorithms. The protocol assessed the algorithms by ranking each set of results for function:dimension:run. For each algorithm the ranks were summed and averaged. If the problem functions are well chosen then the relative capabilities of each algorithm is in some way summarised for these functions and possibly for other problem functions of a similar nature. One weakness of a rank sum approach in comparing the algorithms is that no weighting is given to how much better one algorithm is than another. Two algorithms, A and B, could each win on half of the functions. The rank sum would suggest that they were of equal merit. However algorithm A may always win by orders of magnitude when it wins, but only lose by a fraction when it loses. We should note that as the CEC2013 competition treats each individual run of an algorithm as a separate data point the spread of results is implicitly included in the comparative assessment. However my comparison between CriPS and a selection of the competition algorithms only uses the mean values of 51 runs against each problem function.

However, in lieu of an alternative, still tractable, approach this seems a reasonable means to assess the relative merits of the algorithms. However, given the infinite

number of potential problem functions that could be constructed, we should be somewhat circumspect that any finite benchmark set provides an comprehensive recommendation for a given PSO derived metaheuristic. It is not our intent here to reject the benchmark function approach, only to highlight the fact that there are limitations to this approach.

There are alternatives to using benchmarking to assess the capabilities of an algorithm. Recent approaches that use the results obtained as an algorithm progresses and tune the future performance may provide future directions for development. Adaptive PSO's switch algorithms as the swarm behaviour changes (say from exploring to exploiting) Zhan et al. (2009), Wang et al. (2011). Similarly properties of the problem space's manifold may be detected and used to infer the nature of the problem space for example in exploratory landscape analysis (ELA) Mersmann et al. (2011). Benchmark functions are designed to exhibit different high level characteristics. ELA measures low level properties of the space. These are deduced from the fitness values are calculated by the running algorithm. The authors then map these low level properties to the high level features. Problem spaces with similar characteristics, it is posited, would then be detectable, and metaheuristic algorithms known to exploit such spaces could be deployed. An alternative approach may be to look at the statistics of different algorithms as they explore different problem spaces. With a suitable database of such responses it may be possible to test an unknown problem with an algorithm or set of algorithms and compare their performances with known problem spaces to identify one the unknown problem is most like. One could then select the algorithm known to perform well.

4.5.4 On future directions

Future work should examine whether different ε values would improve the results of the instances where CriPS performs less well. This would allow a meta-process to be designed to make this algorithm yet more general.

It would be interesting to explore something analogous to CriPS' online feedback mediated parameter modifications as an alternative to the empirically determined restart and swarm growth strategies employed by the winning CMA-ES algorithm.

Improved performance may arise from control of CriPS's additional parameters. Whilst this may seem to simply shift the problem of setting PSO's parameters to a new set, it is envisaged that this may be handled by a meta-process. With an unknown

problem we cannot tell if the solutions we are finding are good, so we cannot decide how to change the ω and α_s. However we can monitor the dynamic of the swarm and decide if the distribution of shifts in swarm size are conforming to our target distribution. If not we can adjust the ε value.

4.6 Conclusions

We have presented an algorithm for metaheuristic optimisation that aimed to both employ criticality in a particle swarm and avoid stagnation. The CriPS algorithm modifies the PSO parameters using feedback from the swarm dynamics. Correctly set, swarm movements at all size scales of the problem space can occur: larger exploratory moves being less frequent than smaller exploitative moves. The algorithm does not use any other information about the objective function than the standard version of PSO. Instead our algorithm self-adapts its parameters based on holistic information about the swarm. We have used the average velocity norm, but other properties of the swarm such as local density, swarm extension or gradient of the objective function within the swarm can be used as well. The PSO parameters are obtained from the swarm properties by a simple control algorithm. It would clearly be interesting to study more advanced control schemes.

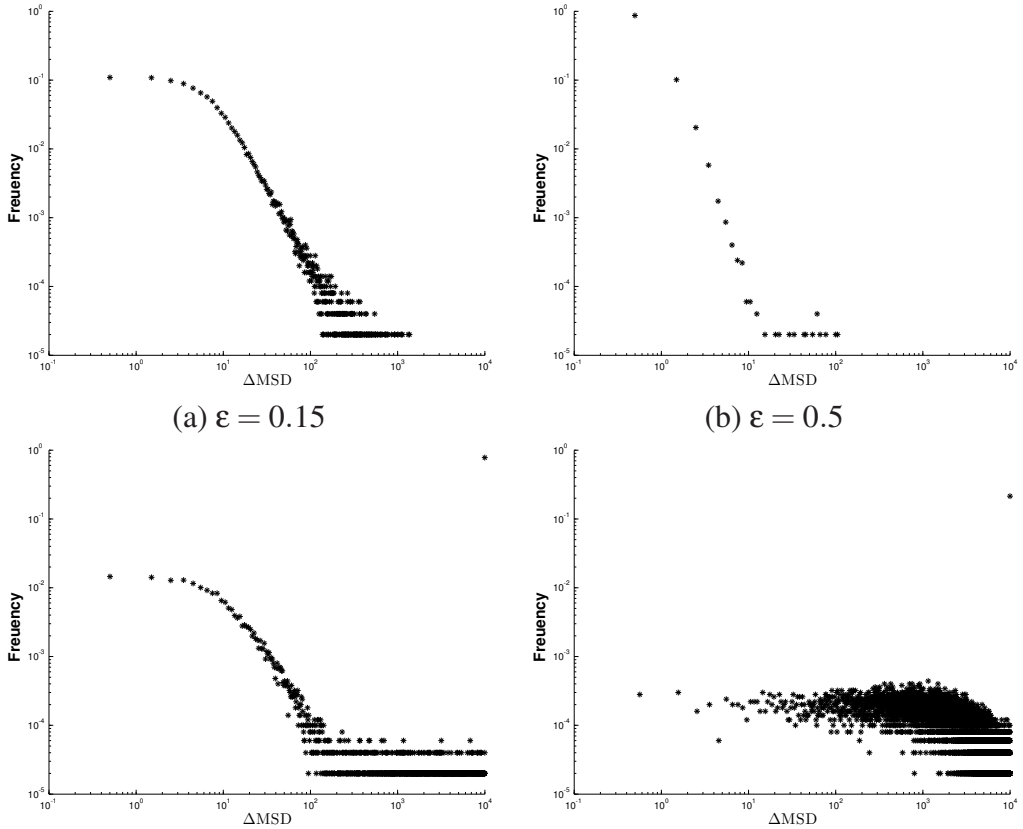
Both Lovbjerg and Krink (2002) and Fernandes et al. (2012) have employed mechanisms to incorporate criticality into the PSO algorithm. Their results are suggestive that it may be of some benefit. The former paper introduces an additional parameter that requires tuning to a problem. We also introduce an additional parameter (ε): in theory this can be set by examining the swarm's behaviour rather than judging the quality of the result. Looking for the 'bursting' dynamic of the swarm size appears to be enough to guide the setting of this parameter. The latter paper uses numbers drawn from a power-law distribution to both modify the PSO's velocity update parameters and to perform a local search. This addition of the power-law distributed numbers is blind to the problem space being explored. Our approach takes a different tack, using feedback from our swarm's interaction with the particular objective function to shape the response of the swarm itself.

We have used the information homogeneously across the swarm, i.e. although the parameters are temporally variable, they are identical for all particles at any given moment of time. Heterogeneous swarms (Sayama, 2009) are known to produce a much richer variety of behaviours which appears to be promising not only for modelling of

biophysical systems (Erskine and Herrmann, 2013) but potentially also for application in optimisation, see (Engelbrecht, 2010).

CriPS outperforms other simple PSO variants. It achieves results that are better than or equivalent to standard PSO in the tests conducted. By modifying the parameters of PSO online using feedback from the algorithm's interaction with the problem space it is able to avoid stagnation whilst keeping the parameter values near (on average) the critical locus of parameter values.

In comparison with other metaheuristics this new algorithm performed more modestly. When compared with the CEC2013 competition functions CriPS was able to always win against function 8 (Rotated Ackley's function). It also performed well against function 9 and 16 (Rotated Weierstrass Function and Rotated Katsuura Function). These functions all shared some common properties (Multi-modal, Non-separable, Asymmetrical) that may be beneficial for CriPS's success. However, the modest showing may reflect the limitations of PSO as an algorithm rather than this new approach. As such it might be useful to explore the use of this feedback mechanism to modify other metaheuristics that suffer stagnation and parameter selection issues. PSO variants like Cuckoo Search avoid stagnation by explicitly never converging. Its Lévy flights derived searches made about the current good solutions ensure continuing exploration. However, this distribution of steps is fixed, and fails to take into account the nature of the problem space that the swarm's sampling is uncovering. Recent metaheuristics such as CMA-ES adjust their search based on statistics gathered about the problem space. They are able to obtain good solutions. However, these can still be deceived into converging on a local minima and require restart strategies to be employed. Geometry and information based measures have previously been examined as the means to use the sampling of the problem space to drive future swarm evolution. The CriPS algorithm's swarm diversity feedback is another fruitful method. Not only is stagnation avoided, but it becomes possible to guide the swarm toward critical or sub-critical behaviour which (for PSO at least) is shown to be optimal.



(a) $\varepsilon = 0.15$ (b) $\varepsilon = 0.5$

(c) $\varepsilon = 0.075$ (b) Standard PSO

Figure 4.5: CriPS swarm size changes showing critical behaviour. The objective function was a 20 dimensional Schwefel function. The swarm size was 250 particles. Modifying the ε parameter changes the behaviour of the algorithm. (top left) Swarm dynamic shows approximate critical behaviour. Plotting the frequency of swarm size changes shows a straight line on a log-log plot. This suggests that there may be a power-law present. Here our algorithm used ε = 0.15 and σ set to $\frac{1}{5}$ of the problem space's size. (top right) Tuning ε=0.5 shows that the swarm makes fewer large changes in size such that exploration of the problem space is limited. (lower left) Tuning ε=0.075 shows a swarm that makes many large changes in size i.e. more extensive exploration is being performed (including many size changes greater than the problem space being explored). Parameter σ remains unchanged throughout. (lower right) for comparison, the bottom plot shows the equivalent plot for the standard PSO algorithm (with ω=0.7, α_1=2, and α_2= 2. Note that thelast bin of the histogram shows all swarm sizes above the upper bound. For the top plot this number is zero.

Chapter 5

Cell division like behaviour in a heterogeneous swarm environment

The previous chapters explored criticality in particle swarm optimisation (PSO). This arose from the properties of the infinite product of its stochastic update matrices. The exchange of information between the particles in PSO modified the stability of the system but did not cause its critical behaviour. In this chapter we examine a different system where critical behaviour arises from an interplay between the particle's parameters and the local information each particle has access to.

This chapter presents work that explored a novel emergent behaviour found within the swarm system known as Swarm Chemistry. The history of this system may be traced to Craig Reynolds Boids algorithm (Reynolds, 1987). In this, a set of particles with position and velocity states were iteratively updated via a simple set of rules. These specified kinetic interactions amongst near neighbours only but the swarm, as a whole, developed motions akin to bird flocking movements. The strengths of the interactions were the same for all particles. Hiroki Sayama extended this model introducing particle heterogeneity and additional velocity controls (Sayama, 2009). This 'Swarm Chemistry' system shows a multitude of emergent shapes and motions that may be thought of as different behaviours. We begin by describing this system in detail before presenting what we describe as a behaviour visually redolent of certain cell-division-like motions. We extend this model and show a repeating cell division mechanism akin to prokaryotic binary fission. The presence of such a behaviour arising from a collection of particles that interact solely through locally applied kinetic interaction rules may be of interest to origin of life theories. The reasoning behind such a claim are discussed. Such theories need to be able to demonstrate the feasibility

of mechanisms that allow replication of chemical sets without the existing cellular replication mechanisms. The fact that simple division can occur in systems with only kinetic interactions is therefore suggestive that forces arising from say, ionic attraction or repulsions, or simply from the interplay of surface tensions, could give rise to division/replication processes. We show the robustness of our behaviour across a finite range of parameter variations and the inclusion of additional particle types to our swarm.

Much of this chapter was originally presented first in a paper presented at The European Conference on Artificial Life (ECAL) 2013 entitled "Cell Division Behaviour in a Heterogeneous Swarm Environment". Following the conference I was invited to extend the paper for inclusion in a special edition of the journal Artificial Life, where it appeared under the same title (Erskine and Herrmann, 2015b). It is rewritten and further expanded upon here.

5.1 Reynold's Boids algorithm

In 1987 an algorithm was presented that generated convincingly realistic bird flocking motions in a swarm of interacting particles (Reynolds, 1987). Each particle interacted only with its near neighbours. The swarms were shown to generate behaviours that visually exhibited similar motions to flocking birds or the schooling of fish. Craig Reynolds named the individual particles of the swarm Boids. Each Boid had a position and a velocity state (although velocity magnitude was fixed). Additionally they had a notional range of perception. Only within this range they could detect the position and velocity of their neighbours. As such each agent of the swarm only had access to local information relating to the swarm. With each timestep the algorithm computes an update to every Boid's state by following a set of three rules:

1. Cohesion. Each Boid calculates the mean position of its neighbours. On the next iteration the Boid adjusts its position so as to move toward this mean position.

2. Alignment. Each Boid calculates the mean velocity of its neighbours. On the next iteration the Boid adjusts its velocity so as to move toward this mean velocity.

3. Avoidance. Each Boid experiences a repulsive force from any near neighbour. The force is inversely proportional to the square of the intra-particle separation. This force acts as a vector pushing the Boid away from that neighbour.

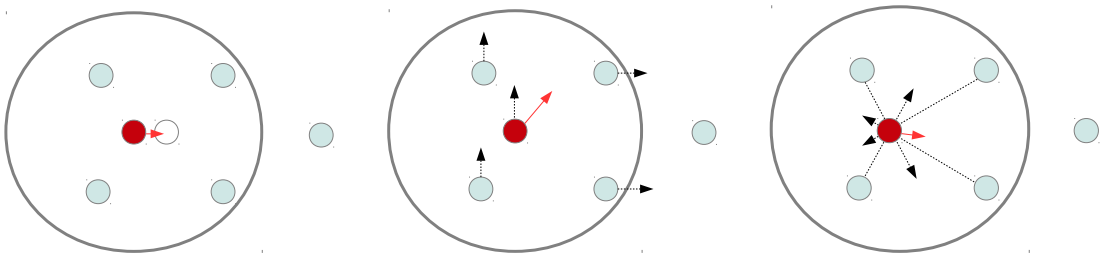


Figure 5.1: The interactions between particles in a swarm following Craig Reynolds' Boids algorithm. Each particle is shown as a small circle. The red circle is the particle currently being considered. Its neighbourhood radius is shown as the large black circle. The light blue circles are the rest of the swarm. Only those blue particles within the neighbourhood radius of the red particle will be influence it. The red arrows indicate the resultant target vector for each of the three rules. Black arrows indicate the current velocity vectors of the particles. The three update rules are shown, from left to right: Cohesion; Alignment; Avoidance.

Thus each boid will experience a pull toward the mean position of its neighbours, some rotation in its velocity vector and individual directed repulsive forces on each iteration. These rules may be visualised as shown in Figure 5.1.

5.2 Hiroki Sayama's swarm chemistry

The Boids algorithm treated the swarm as a set of homogeneous agents. This was extended by allowing multiple swarms to interact (Sayama, 2009). Each swarm may have different sets of parameters: varying the cohesive pull, the intra-particle repulsion or the velocity realignment strength felt. Additionally the magnitude of the velocity each particle has is allowed to change as the rules are applied. Each particle has a preferred, or a natural speed. Thus, if the pulling and pushing of the interactions speeds up (or slows) a particle it will in time naturally return to its preferred speed. The full algorithm also includes the element of noise in the system: there is a probability of random steering. Sayama calls this *whim*.

5.2.1 Swarm chemistry algorithm

Each particle is defined by a total of eight parameters. These consist of the three kinetic interaction parameters (cohesion, alignment and avoidance), the neighbourhood radius, the preferred and maximum speeds, the rate by which a particle returns to its preferred speed, and the chance of random steering. Changing the set of parameters for a particle will change the strengths of the various interaction forces that it will experience.

Particles with the same set of parameter values may be thought of as belonging to a *species*. Similarly we can talk of a heterogeneously mixed swarm as containing multiple species of particles.

We use the following to denote the various parameters that constitute a single species of particle:

- c_1: The cohesion parameter defining the strength of attraction to the mean position of a particles neighbours. c_1 is real valued in the range [0,1].
- c_2: The alignment parameter defining the strength of pull toward the mean velocity of a particles neighbours. c_1 is real valued in the range [0,1].
- c_3: The avoidance parameter defining the strength of repulsion from neighbouring particles. c_1 is real valued in the range [0,100].
- c_4: The whim parameter defining the chance that a particle ignores its neighbours influence on any given iteration. This essentially adds noise into the system. c_1 is real valued in the range [0,1].
- c_5: The speed control parameter defining the rate that a particle will return to its preferred speed. c_1 is real valued in the range [0,1].
- rad: The neighbourhood radius, the distance over which neighbouring particles are perceived.
- spd: The preferred speed of a particle
- msp: The maximum speed of a particle

For each particle i in the swarm we consider the set of particles, $\mathcal{N}$ that are within its neighbourhood radius. Particle i has a position and velocity state $\{\mathbf{x_i}, \mathbf{v_i}\}$. To update these states each particle considers the states of those particles in $\mathcal{N}$.

The average position of these is

$$\langle \mathbf{x} \rangle = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} \mathbf{x}_j \tag{5.1}$$

The average velocity of the particles within the neighbourhood radius of particle i is

$$\langle \mathbf{v} \rangle = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} \mathbf{v}_j \tag{5.2}$$

The kinetic interactions between a particle i and its j neighbours can thus be considered as an update to $\mathbf{v_i}$. There are three components to this update due to the three update rules:

The cohesive force

$$\mathbf{F_c} = -c_1 \left(\mathbf{x}_i - \langle \mathbf{x} \rangle \right) \tag{5.3}$$

The alignment force

$$\mathbf{F_a} = -c_2 \left(\mathbf{v}_i - \langle \mathbf{v} \rangle \right) \tag{5.4}$$

The inverse square repulsive force

$$\mathbf{F_p} = c_3 \left(\sum_{j \in \mathcal{N}} (\mathbf{x}_i - \mathbf{x}_j) / |(\mathbf{x}_i - \mathbf{x}_j)|^2 \right) \tag{5.5}$$

The velocity adjustment due to these kinetic interactions is the sum of these components which may be considered an acceleration on the particle.

$$\begin{aligned} \mathbf{a}_i &= -c_1 \left(\mathbf{x}_i - \langle \mathbf{x} \rangle \right) - c_2 \left(\mathbf{v}_i - \langle \mathbf{v} \rangle \right) \\ &\quad + c_3 \left(\sum_{j \in \mathcal{N}} (\mathbf{x}_i - \mathbf{x}_j) / |(\mathbf{x}_i - \mathbf{x}_j)|^2 \right) \end{aligned} \tag{5.6}$$

The dynamics are further modified by the c_4 parameter which is the probability of ignoring the neighbours' effects in the current iteration. The particle's velocity is updated using the acceleration $\mathbf{a}_i$.

$$\mathbf{v}'_i \leftarrow \mathbf{v}_i + \mathbf{a}_i \tag{5.7}$$

The magnitude of a particle's velocity has an upper bound. This is one of the swarm's parameters. Similarly each swarm has a parameter that is the preferred magnitude of the particles' velocity. If a particle is not travelling at this preferred velocity, v_n, then parameter c_5 is used to nudge the velocity back toward its preferred velocity using

$$\mathbf{v}_i \leftarrow c_5 \frac{v_n}{|\mathbf{v}'_i|} \mathbf{v}'_i + (1 - c_5) \mathbf{v}'_i \tag{5.8}$$

Finally each particle's position is updated using

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i \tag{5.9}$$

5.2.2 Behaviours in swarm chemistry

For the purposes of visualising the swarms we use a 3-tuple of the cohesion, alignment, and avoidance parameters in order to create the colour used to display the particles of a particular species. Whilst this means that, in theory, the colour used is not unique, it has not in practise caused confusion.

Reynold's Boids were homogeneous swarms and generated flocking motions. Here, we can mix two or more species of particles and observe their behaviours resulting from their iterated interactions. Unusual structures and dynamic behaviours have been seen (Sayama, 2010, 2012a,b). Many swarms could be identified that have a distinct biological look to them: cells, amoebas, diatoms abound. Fig 5.3 shows a number of stills of a two species swarm that shows emergent persistent oscillation behaviour. It is tempting to see the dynamics of the so-called *swarm chemistry* as a simple model for the real life counterparts of these forms.

The simplest swarms in swarm chemistry are homogeneous swarms. As swarm chemistry treats velocities differently to the Reynold's Boids algorithm the range of behaviours also differs. A limited palette of behaviours manifest. Particles tend to aggregate into one or more long lived structures. These are roughly spherical and may be hollow i.e. presenting as a spherical shell. Alternatively the particles may simple disassociate and become a cloud of non interacting particles. Fig 5.2 shows examples of these homogeneous swarms. It should be noted that these visualisations are not necessarily to the same scale.

5.3 Cell division like behaviour in swarm chemistry

Hiroki Sayama's Swarm Chemistry website (Sayama, 2015), lists many sets of species that result in interesting behaviours. Discovery of these behaviours has been chiefly driven by manual and evolutionary methods. Software to explore Swarm Chemistry is also available. Many of the listed behaviours have been located by third parties manually exploring the parameter space.

The author located a set of particle species that result in a novel behaviour that may

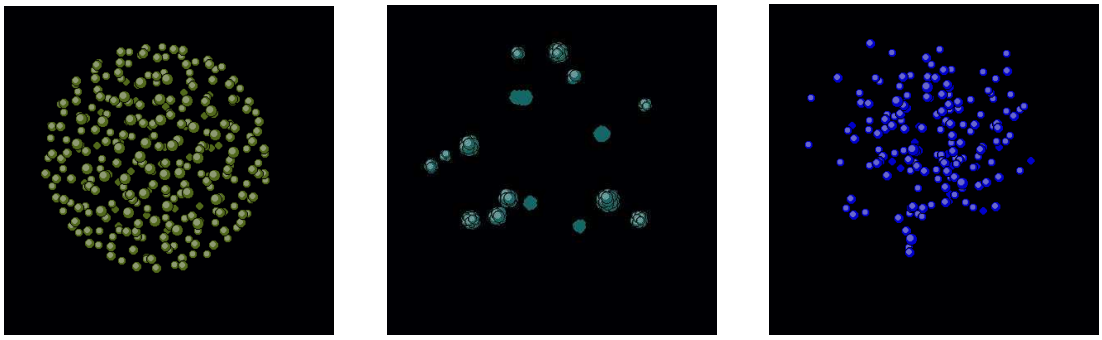


Figure 5.2: Some homogeneous swarms. Single species in Swarm Chemistry manifest a limited set of behaviours: a spherical cluster, multiple clusters, disassociated clouds of particles. NB Scale differs between images.

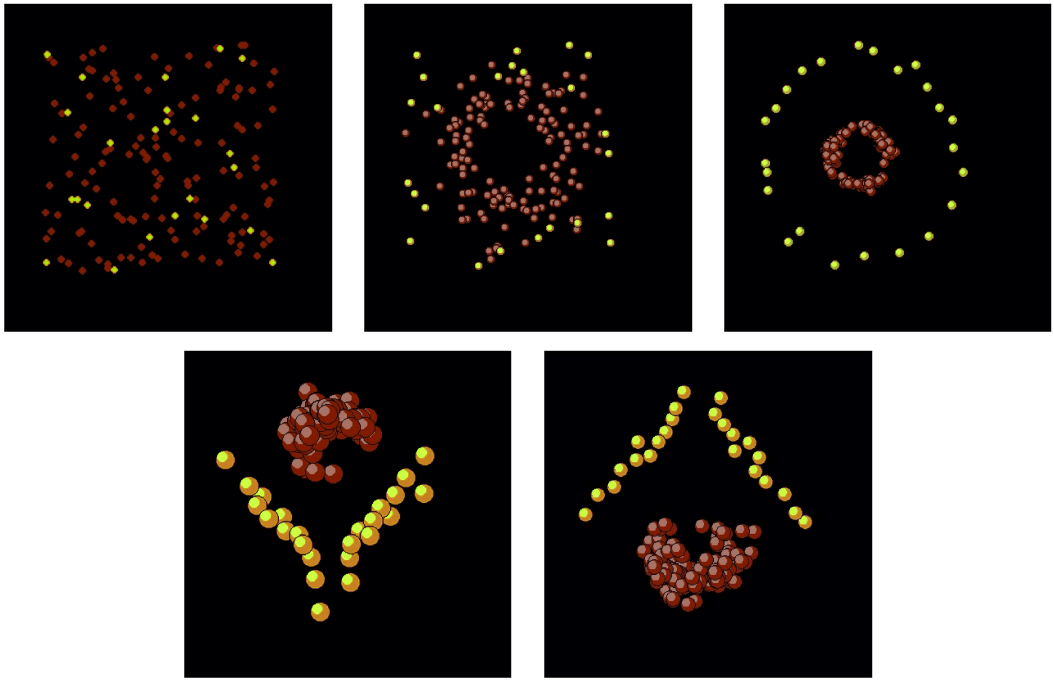


Figure 5.3: Heterogeneous Swarm Chemistry behaviours. Complex behaviours exist within the Swarm Chemistry system. Here a two species swarm shows an oscillator behaviour. (From top left) an initial mix of two species, shown yellow and red, spatially separate, and then create a persistent oscillatory motion.

be described as cell-division-like, shown in figure 5.5. The swarm is constructed from particles of two species. In our visualisations these are displayed as red and yellow particles. The swarms we examine typically have many more yellow particles than red. The position and velocity states of all the particles are randomly initialised, but with the positions lying closely together to assure that interactions will occur according to the rules presented above. Whilst initially mixed the two particle species quickly separate, with the red particles forming a ring about the (approximate) middle of a spherical cluster made of the yellow species. Depending on the specifics of the parameter values chosen and the size of the two populations this arrangement can exist for a considerable time (many hundreds of iterations). At some point the red ring appears to contract, causing the yellow cluster to split in two. The two clusters of yellow particles tend to move apart and the red particles rejoin one of those clusters.

The remainder of this chapter covers the investigations exploring the nature and robustness of this behaviour. Specifically:

- We develop some tools that allow us to quantify and track the behaviour.
- We examine and characterise the cell-division-like behaviour.
- We explore to what extent the behaviour is affected by the total size of the swarm and the populations of each subspecies.
- We examine differences and similarities in the behaviour when exhibited in two and three dimensional environments.
- We vary the parameter values of the two species to see the effect of such changes have on the emergence of our target behaviour.
- We show that the behaviour can survives the addition of a third species.
- Finally we make extensions to the Swarm Chemistry model that enable us to extend the behaviour to show repetitions in the cell division.

5.3.1 Quantification

We have said that a particle species is defined by eight parameters (c_1 through c_5, neighbourhood radius, speed and maximum speed). A swarm may be made of particles of many species and in varying numbers. Any specific swarm is thus defined as a point in a large parameter space. Even if we had tools to assist in the automatic detection of

novel behaviours in any given heterogeneous swarm in this system it would remain a highly onerous task to explore all portions of the parameter space. A simple method of exploration is to perform random searches and hope to locate islands of interest. Swarm Chemistry's originator, Hiroki Sayama, had his students and third parties do just this. He provides software to generate swarms, watch their behaviours and mix interesting species together. I have also followed this approach. When one locates a behaviour of interest, we can examine it by making small changes to the parameter values to assess the robustness of the behaviour to such parametric changes. When dealing with a smaller portion of the parameter space in this manner it becomes more tractable to use quantifications and tools to track the onset and cessation of a specific behaviour. The tools we have used include measuring the swarm density. Changes in density in a swarm as it evolves indicates a sudden shift in its spatial distribution that may be indicative of a change of state. We also use measures based on entropy as a measure of structure in our swarms, and clustering analysis to support more detailed investigations.

5.3.1.1 Density and entropy

Structure in an evolving swarm can be most usually be seen in how particles tend to agglomerate. Often groups or clusters form, on other occasions the particles will disassociate. In our swarms we can calculate the average density of particles. This density measure differentiates single clusters from both dispersed swarms and multiple groups: single clusters show a higher density. We note that this may not always be true: a large hollow single cluster may be less dense than multiple groups that are close together. A second measure, a spatial entropy, allowed differentiation between multiple groups and dispersed swarms. It has been suggested (Bonabeau et al., 1999) that a spatial entropy can be defined as

$$H = -\sum_{k} P(k) \log P(k) \tag{5.10}$$

where $P(k)$ is the fraction of particles found in patch k. H decreases as clusters form. We used patches that are always cubes of side 0.1 times the maximum extent of the swarm i.e. the minimal cube containing the swarm is split into 1000 patches. Two similar treatments are made in (Batty, 1974) and (Wolfram, 1984).

We also make use of the Kullback-Leibler divergence to provide a measure of our swarm's *distance* from a swarm with the same number of particles that is evenly

distributed through the same minimal cube as the target swarm. The Kullback-Leibler divergence is defined:

$$D_{KL} = \sum_k P(k) \log \frac{P(k)}{Q(k)} \tag{5.11}$$

where P is the distribution of the particle positions and Q is the distribution of an evenly dispersed swarm. Note that since Q is evenly distributed, we have simply $D_{KL} = \log \frac{1}{Q(k)} - H$. For the cell division like behaviour D_{KL} thus increases when the swarm has divided into separate clusters.

5.3.1.2 Clustering

The cell-division-like behaviour swarms start as a single cluster. As divisions occur there are increasing numbers of sub-swarms. All particles in the model are individually identifiable but the swarm has no knowledge of sub-swarm structure. We can track individual trajectories but have no means to locate sub-structure in the swarm that seems obvious to the observer. We have made use of the k-means clustering algorithm in order to identify which particles are in which sub-swarm. K-means clustering takes as an input a specific number (k) clusters that the algorithm is to find. It results in the data partitioned into these k clusters. Given a set of n particle positions $\{x_1, x_2, ..., x_n\}$, the algorithm attempts to partition the particles into k sets $S = \{S_1, S_2, ..., S_k\}$, such that we minimise the within cluster sum of the squares:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{5.12}$$

where μ_i is average position of the points in S_i.

Critically this method needs to know the number of clusters to find. For small numbers of clumps the author found that this may be estimated by counting the minima in a histogram of the inter-particle distances.

5.4 Results

5.4.1 Single species characterization

A homogeneous swarm appears to exhibit behaviours drawn from a fairly limited palette of possible behaviours. We note four behaviours: full dispersal, cluster or

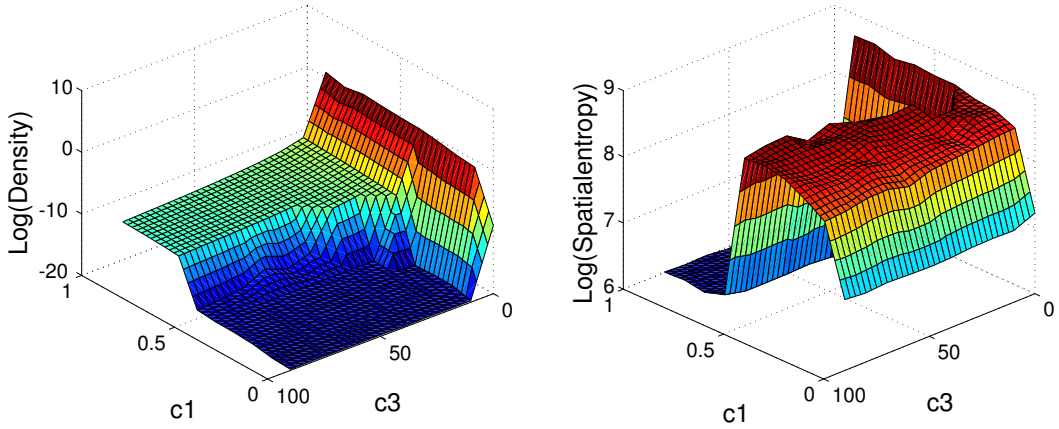


Figure 5.4: Density and entropy measurements for a homogeneous swarm as a function of its cohesion (c_1) and avoidance (c_3) parameters. The logarithm of each measure is plotted in order to squash the vertical extent of the surface plots as the range of values extends over several decades.

sphere, multiple groups, and one we call a point swarm. A point swarm is one where all particles collapse toward a single point, i.e. the cohesion force appears to dominate all other interactions. We mention it here for completeness only. Examples of the remaining three behaviours are shown in Fig 5.2. In full dispersal the particles separate and move apart. There is little or no tendency to aggregate. In a cluster the particles form a sphere (or approximate sphere) or shell of a sphere. Multiple groups are simply a multiple version of the last form. Point swarms are seen for swarm parameters where the avoidance value is at or near zero. This results in all particles collapsing to a single point. This state tends to not show as a sphere or a point. Instead the particles, which all exist in a tiny spatial volume, show as an irregular cluster of particles that jump about. Particles have discretised speeds so at each update a particle tends towards the average position of the cluster, but the step size is larger than the size of the cluster, thus the particles are unable to actually occupy a single point.

We find that the four single species states can be classified by the density and spatial entropy (or Kullback-Leibler divergence). By sweeping through the parameter space of the cohesion and avoidance parameters it was possible to find regions of each of the four swarm types. The spatial entropy and densities were measured for each. A visual check of the final state of the swarm was also made. The measures were plotted against the parameters to generate the surface plots shown in Fig. 5.4. The types of behaviour seen in these homogeneous swarms can be summarised by their location in density-entropy space as seen in Tab. 5.1.

	Entropy Low	Entropy High
Density low	Dispersed	Multiple groups
Density High	Single cluster	Point swarm

Table 5.1: Summary of behaviours in homogeneous swarms as a function of density and entropy

spc	rad	spd	msp	c_1	c_2	c_3	c_4	c_5
1 (yellow)	20.5	1.94	20.7	1	1	18.6	0.05	1
2 (red)	300	15.58	37.08	1	0.05	9.11	0.47	0.61

Table 5.2: Parameter values for the cell division like behaviour swarms. Headings are: spc = swarm species, rad = neighbourhood radius, spd = normal speed, msp = maximum speed, c_1 = cohesion, c_2 = alignment, c_3 = avoidance, c_4 = whim, c_5 = speed control.

5.4.2 Cell division behaviour species

We present two species of swarms that individually formed single clusters (multiple groups if their populations were large enough), but in combination result in a cell division like behaviour. Typical stages of this are shown in Fig. 5.5. The values for the parameters used in these swarms are shown in Tab. 5.2.

When displaying the particles we colour code them using their parameters c_1, c_2 and c_3 as red, green and blue components. Here, species 1 displays as a yellow colour and species 2 as a red colour (n.b. if viewed in black and white copy the red particles appear slightly darker). Clearly, the displayed colours change if we alter these parameter values. For descriptive convenience we choose to describe the two swarms as the yellow and the red swarms respectively. Typically the swarm population was constructed with many more yellow particles than red particles (ratios varying from about 5:1 to 10:1 are variously used). This heterogeneous swarm is initialised such that the particles are close enough together for interactions to occur. The two species are well mixed throughout this initialisation volume. The early evolution of the swarm is for the two species to separate. The yellow ones form a central core, the red ones form an outer shell. The yellow core tends to elongate and the red ones form a toroid roughly positioned centrally along the yellow core. The toroid squeezes and separates the yellow core into two parts. These move apart, with the red particles forming a cluster in between. At some point the reds are drawn into one of the yellow clusters and the process repeats. In the basic configuration the repetition only occurs within the

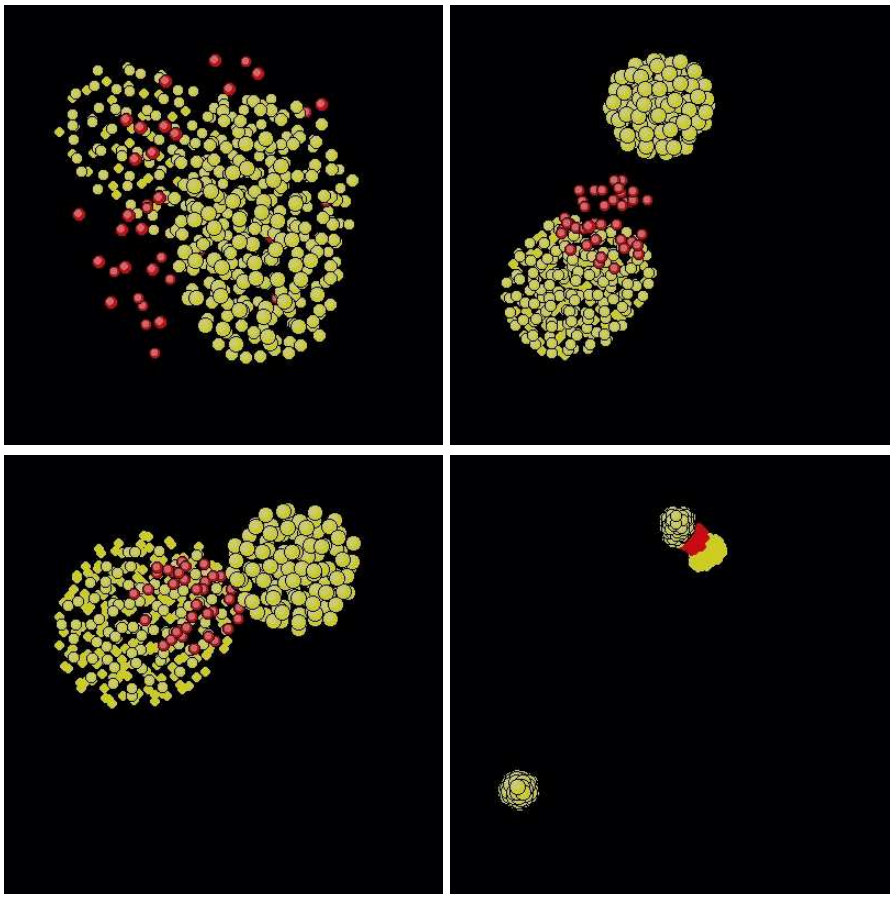


Figure 5.5: Typical evolution of cell division in our swarm. Top left shows red particles as a toroid about the yellows. Top right shows the yellow swarm divided in two with a separate red swarm. Bottom left has the reds rejoining the larger yellow cluster. Finally bottom right, the process repeats and is shown at a larger scale.

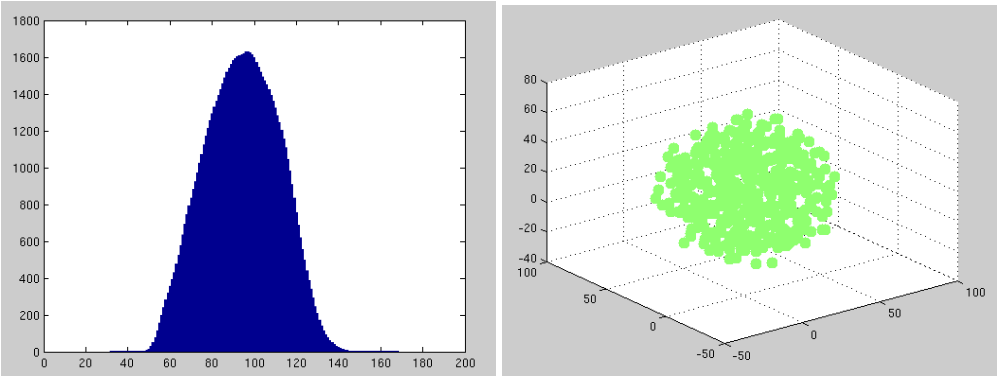


Figure 5.6: Left image shows the smoothed histogram of pairwise particle separations in a single cluster. No minima are detected implying a single cluster. K-means clustering locates particles in this (trivial) instance and assigns them a single colour.

cluster comprised of both yellows and red particles.

To explore how the swarm splits we employed k-means clustering to allow us to locate the particles within each sub-swarm. This algorithm requires as an input the number of clusters to locate. Our swarms typically consist of several hundreds of particles. It is not too onerous to calculate all the pairwise particle separations. If we consider the histogram of these values we can use the number of minima to estimate the number of clusters. A single swarm consisting of a single cluster shows a distribution of pairwise separations that is roughly Gaussian, thus having no minima. We estimate the number of clusters as one more than the number of minima. It is necessary to smooth the histogram (say by convolving the raw histogram with a Gaussian distribution). Fig 5.6 shows a visualisation of a single cluster: with its smoothed histogram of particle separations and the cluster coloured to show particles assigned to the clustered sub-swarms. Fig 5.7 shows the same data for the swarm now in three parts.

As a process this is not error free, but for the avoidance of mistakes the process may be guided by a user. Fig 5.8 shows such an error, but for the purpose of analysing what is occurring in these divisions the mechanism proves to be robust.

This technique is applied to the cell-division-like behaviours seen in swarms with a range of populations of yellow and red particles. Visually the clustering can be corroborated to assure no errors in clustering were made. In each case the division of the yellows is asymmetric (see Tab. 5.3) i.e. the divided parts were of unequal size. In all cases the red particles were always seen to rejoin the larger group.

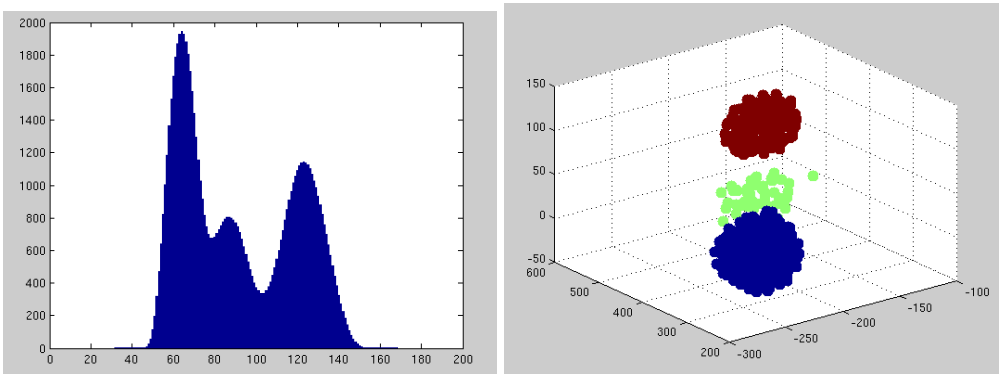


Figure 5.7: Left image shows the smoothed histogram of pairwise particle separations in a recently divided swarm. Two minima are detected implying three clusters. K-means clustering locates particles in each of these three clusters and assigns them to differently coloured groups.

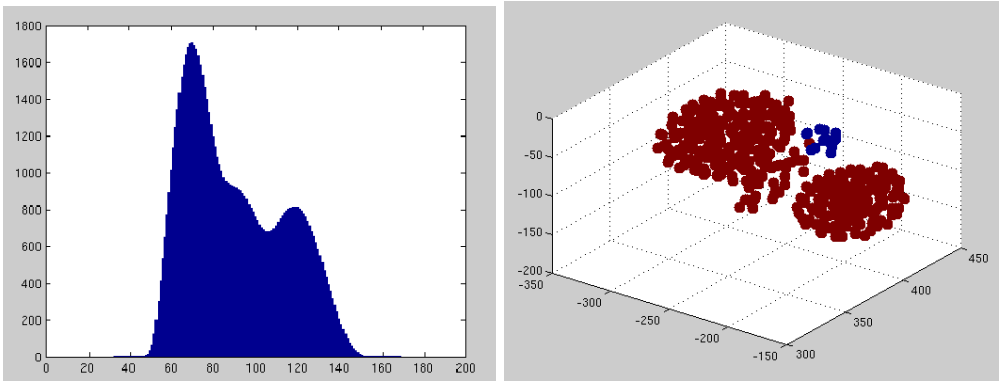


Figure 5.8: Cluster detection procedure fails here. The division is still in process, so the histogram of pairwise separations does not clearly show two minima. Having incorrectly predicted only 2 clusters, the k-means algorithm makes a poor job of identifying the clusters.

Yellow population	Red population	Larger yellow cluster's population	Smaller yellow cluster's population	Ratio
300	50	174	126	1.38
300	70	177	123	1.44
300	90	157	143	1.10
350	50	211	139	1.52
350	70	219	131	1.67
350	90	214	136	1.57
400	50	327	73	4.48
400	70	277	123	2.25
400	90	243	157	1.55

Table 5.3: Pre and post division populations. The populations of initial species were varied (yellows = 300, 350, 400, reds = 50, 70, 90). Following the division the number of particles in each cluster were determined via the k-means clustering algorithm. Yellow division was always asymmetric. Subsequently the red population was always observed to rejoin the larger of the two yellow groups.

5.4.2.1 CDL species on their own - Yellow

What is the nature of each of the two species in our swarm? The division process we see when they are combined may derive in part from some aspect of their solo behaviours. Homogeneous swarms were examined above and found to exhibit a limited range of behaviours. To confirm that our species conformed to this pattern we took each species and generated homogeneous swarms of different sizes. Each swarm was run for 1000 iterations (long enough for apparent evolution of behaviours). Each investigation was repeated ten times to assure no special position initialisation dependence had been observed. First yellow swarms constructed of particle populations of 35, 350, 1000 and 3500. The three smaller swarms showed single stable (approximately) spherical clusters. During a single execution the 1000 particle swarm was seen to *eject* a single yellow particle. With a population of 3500 the swarm frequently (about 50% of the runs) broke away from a spherical symmetry: contorting, extending, and splitting. Examples of final particle positions are shown in Fig 5.9. Figure 5.10 shows some typical stages in the division of the larger such a swarms.

5.4.2.2 CDL species on their own - Red

We repeated the previous study with homogeneous swarms constructed from our red particles. Again populations of 35, 350, 1000, 3500 were used. On this occasion

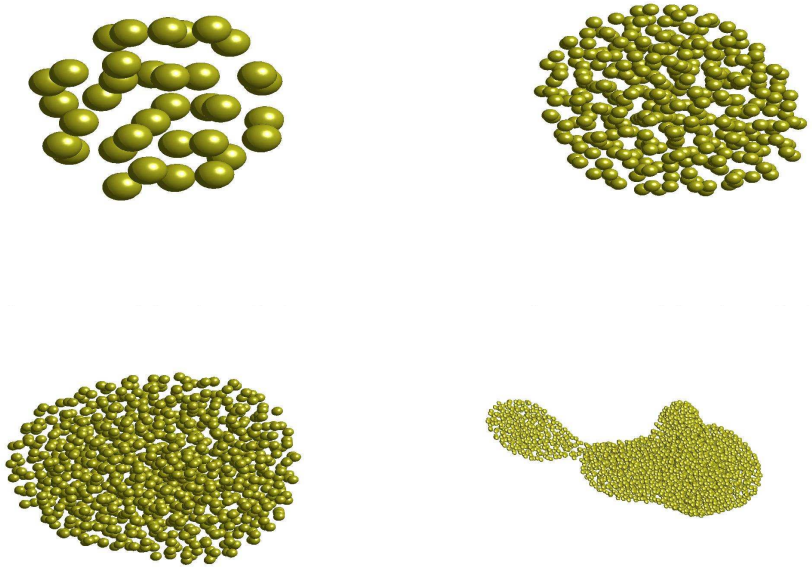


Figure 5.9: Homogeneous yellow particle swarms of population sizes (top left) 35, (top right) 350, (bottom left) 1000, (bottom right) 3500 show that larger swarms diverge from the approximately spherical cluster shape. The largest swarms tending to divide.

all swarms showed stable single clusters. Resultant swarms are shown in Fig 5.11. This different behaviour may follow from consideration of neighbourhood radii of the different particles. The red particles interact over larger distances: having a neighbourhood radius parameter of around 300 units compared to the yellow particleâs radius of around 20 units. Many more particles are thus contributing to the cohesion of the swarm. Additional single executions of swarms with populations of 10000 and 35000 showed that the swarm remained a coherent single cluster.

Of the two constituent species of our cell division swarm it is only the yellow ones that show inclination to divide on their own. This is only seen in considerably larger swarms and the stretching and twisting that the yellow homogeneous swarm undergoes as division occurs is markedly different from the cell-division-like behaviour described earlier.

5.4.3 Comparison with 2D swarm chemistry

Hiroki Sayama's initial explorations were conducted in a two dimensional space. Structures in such a world are necessarily circles and rings rather than the toroids

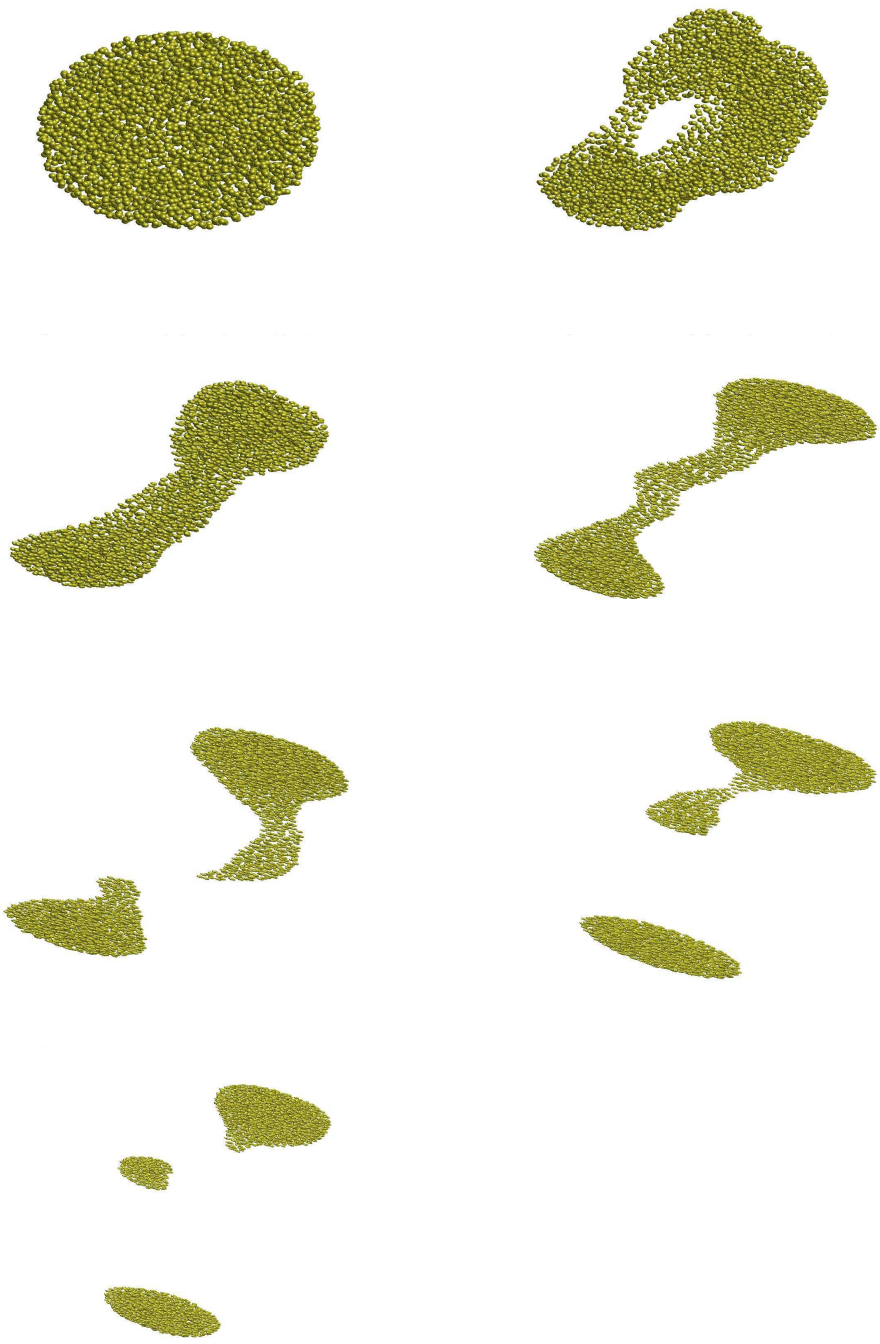


Figure 5.10: Example of population 3500 yellow swarm showing spontaneous division. Sequence begins top left.

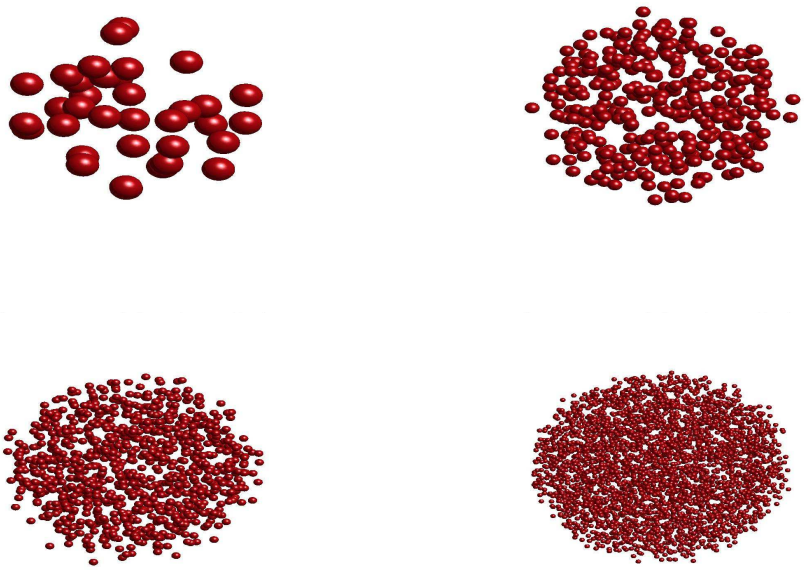


Figure 5.11: Homogeneous red particle swarms of population sizes (top left) 35, (top right) 350, (bottom left) 1000, (bottom right) 3500 show that all swarm sizes result in spherical cluster shapes.

and spheres our swarm manifests. As one increases the dimensionality of the space of this swarm system it becomes possible for more particles to exist within a given neighbourhood radius. This simply follows from the amount of the space within a radius r of a point changing from the area of a circle to the volume of a sphere. The kinetic forces acting on a given particle will therefore also be different. We thus expect our swarm to behave differently depending upon the dimensionality of the space our swarm inhabits.

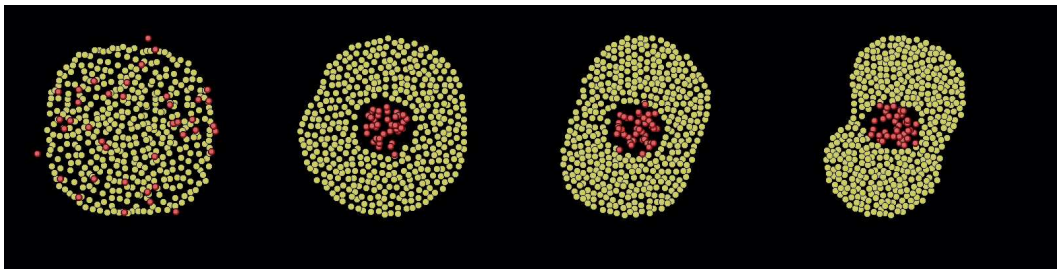
We explored the differences in 2D and 3D behaviour of our swarms. In the first instance we kept the parameters identical and observed the behaviour of the swarm in two dimensions. A form of cell division behaviour still occurred. As previously described the three dimensional division has the red particles forming a ring or toroid about the yellow particle swarm. These appear to squeeze this core until division occurs. The separated parts then travel apart. In the two dimensional case the red particles travel to the inside of a yellow circle of particles and from the *inside out* cause division to occur. The post division dynamics are different. It is notable that the separate parts do not travel apart, nor do the red particles get drawn back into one of the yellow clusters. See Fig. 5.12 for example images.

An *outside in* division, somewhat similar to what we have described in 3D, was achieved via modification of both species' parameters. As parameters have been changed, the particles no longer appear as red and yellow but as magenta and cyan respectively (shown in Fig. 5.12). We will continue to use 'red' and 'yellow' and names to refer to the minority ring forming species and the majority central core forming species respectively. The red particles form a ring around the yellow circle and squeeze it until division occurs. Again the separate parts do not travel apart. It is possible that reintegration of the red particles with one of the yellow clusters would occur if the swarm was left to run. It is also possible that with further parameter modification a recipe may be found that results in the split parts separating. The converse is true: we shall see later that there exist swarms in 3D that show dynamics akin to the cell divisions seen in 2D (see Section 5.4.5.2).

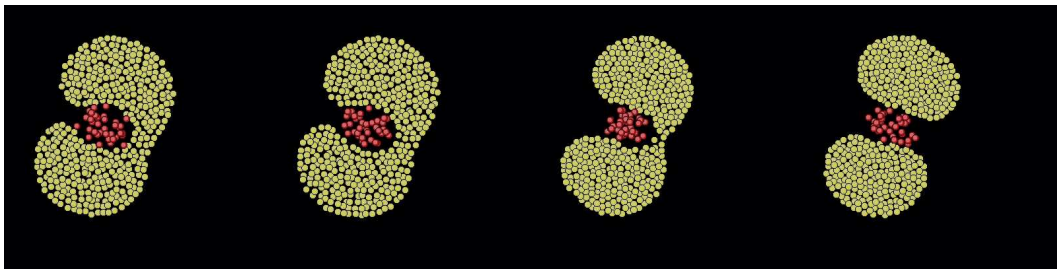
5.4.4 Robustness under population dynamics

5.4.4.1 Yellow versus red populations

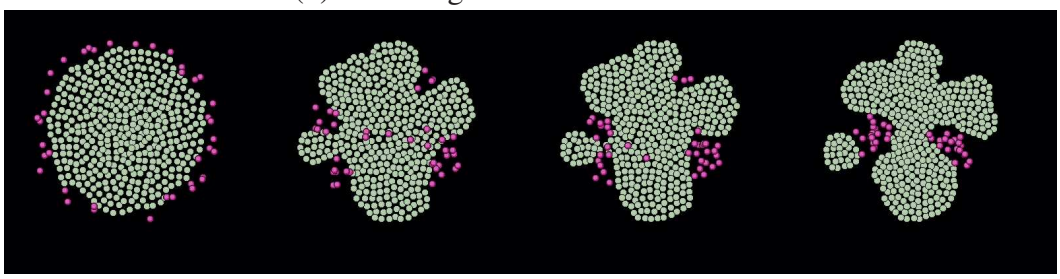
The cell division like behaviour swarms are made of two types of particle. These can be varied independently to explore the robustness of the observed behaviour on the size of their constituent sub-swarm sizes. The observed behaviour is suggestive that the red particle ring squeezes the yellow core at least encouraging separation. It seems reasonable that too few red particles or too many yellows may interfere with this. Using variations of the sub-swarm populations allow the determination the limits on the cell division like behaviour. Each run lasted for 2000 time ticks. The density and entropy measures were captured at the end of each run. For confirmation the final state of the swarm was captured as an image. Yellow populations were varied over a range from 100 to 550 in steps of 50, and red population over the range 10 to 90 in steps of 10. Fig. 5.13 shows the density and entropy measures as a surface plot for all combinations of these populations. Cell division occurs in regions of low density (blue on left hand plot) and high entropy (red on right hand plot). We see that the cell division behaviour extends over a wide range of populations. Very low red or high yellow populations tend to never show cell division. The line between division and no division is noisy. We assume this is due to variability in starting position of particles and/or the arbitrary duration of each run. Both assumptions may be examined.



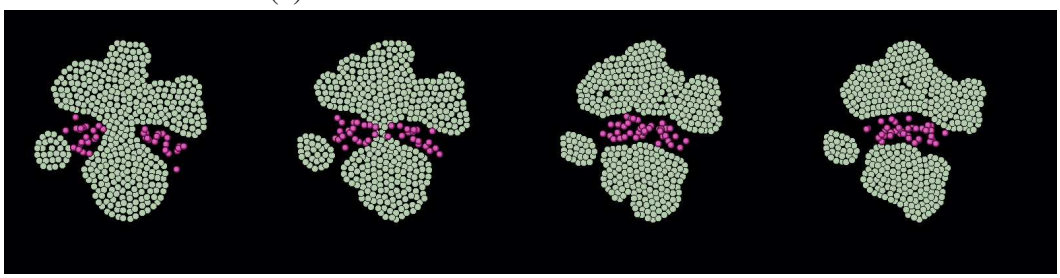
(a) Initial evolution of inside out division



(b) Final stages of inside out division



(c) Initial evolution of outside in division



(d) Final stages of outside in division

Figure 5.12: 2D cell division. Upper two rows show an 'inside out' division in eight frames. Initial mixture separates to an inner core of red particles, and an outer ring of the more numerous yellow particles. Gradually the red particles *eat* their way out, whilst the yellow population develops a narrowing. Finally the two halves of the yellow population pull apart. In the lower two rows we show an 'outside in' division in eight frames. A pair of species with modified parameters recapture something of the original cell division behaviour. As the parameters have changed, the displayed colours have been altered accordingly. The 'yellows' appear as cyan and the 'reds' as magenta. The reds circle the yellows. Gradually they constrict the larger sub-swarm, pinching it and eventually causing division.

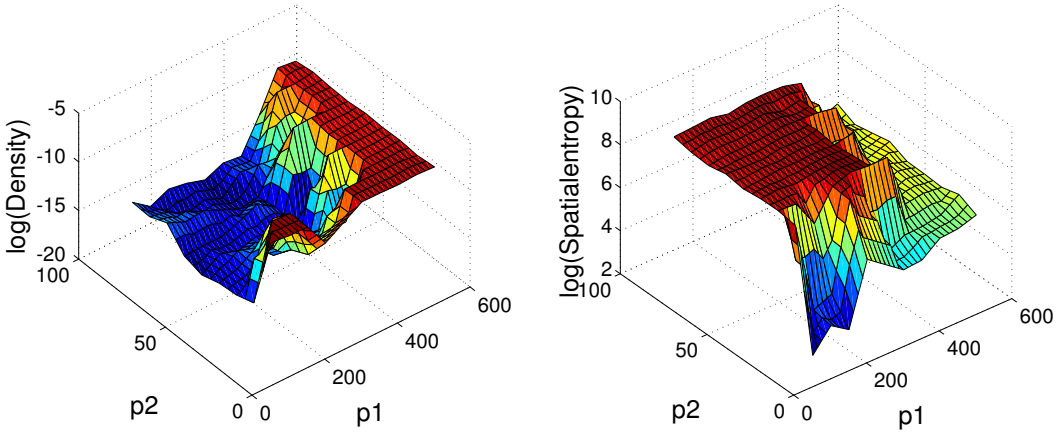


Figure 5.13: Density and entropy measurements for a heterogeneous swarm as a function of its yellow and red populations (p_1 and p_2). The logarithm of each measure is plotted in order to squash the vertical extent of the surface plots as the range of values extends over several decades.

5.4.4.2 Effect of initial particle position

The effect of the initialisation positions of a swarm's particles upon the onset of cell-division-like behaviour can be explored by executing multiple runs of similarly constituted swarms. Here, the red population is fixed at 50 particles, and we vary the yellow population. This represents a slice through the population space of potential swarms of the kind we are examining. Yellow populations are varied from 300 to 600 with a step size of 25 particles. For each swarm we allow five executions where the initial positions of the particles are different for each run. With the yellow population below 375 division always occurred. With yellow populations above 450 divisions never occurred. Thus the extreme ends of population appear to result in a definite outcome within the time of each run. However, in the range between (yellow populations between 375 through 450) the presence or absence of division is not certain. The KL divergence and the density (averaged over the 5 runs) are summarized in Fig. 5.14. The density measure is low in divided swarms, high in undivided swarms. The Kullback-Leibler measure inverts this logic. The region where division or its lack is uncertain shows intermediate values for both measures. Either measure can thus provide a clear signal for the onset of our target behaviour. Table 5.4 shows the final positions of each population mix (up to yellow population equal to 500) at the end of each run.

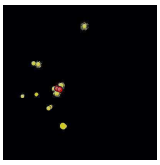
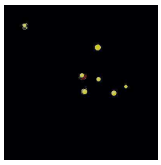
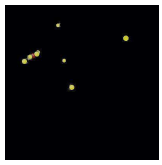
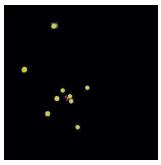
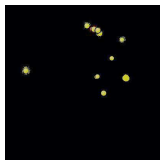
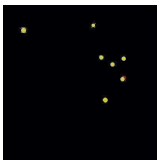
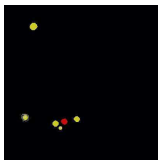
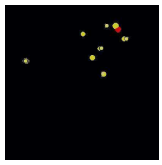
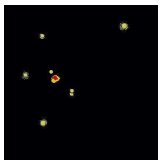
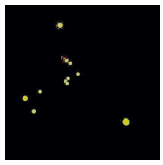
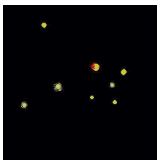
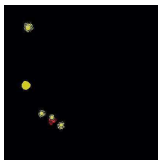
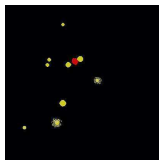
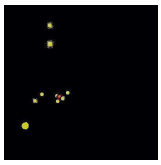
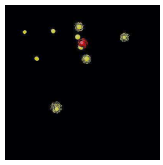
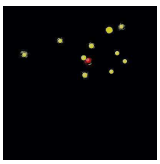
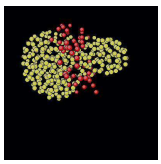
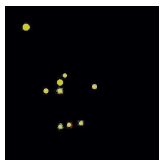
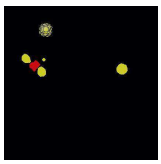
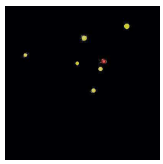
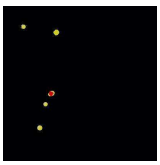
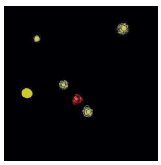
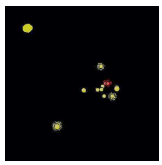
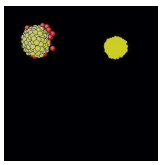
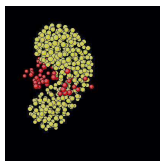
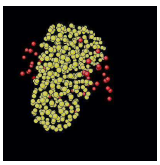
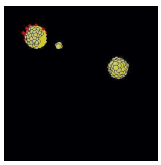
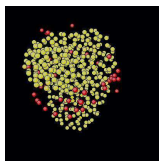
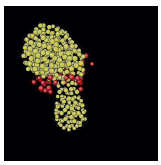
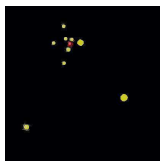
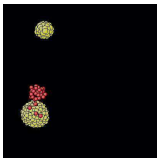
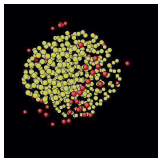
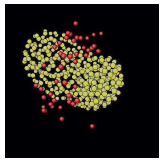
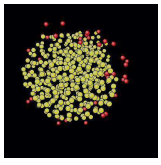
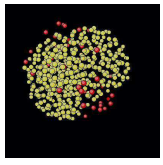
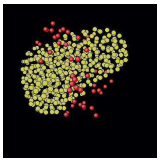
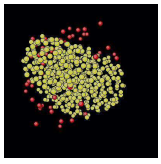
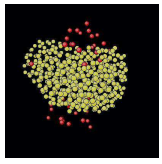
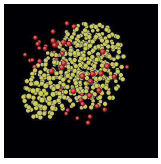
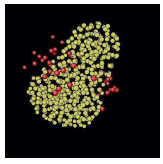
Yellow population	Run 1	Run 2	Run 3	Run 4	Run 5
300					
325					
350					
375					
400					
425					
450					
475					
500					

Table 5.4: Population dependence on cell division behaviour. Red and yellow populations are varied. The spatial state of the final frame of each run of each population mix is shown (after 2000 time ticks). Red population is set to 50 particles, yellow population is varied from from 300 to 500 particles in steps of 25). There were five repetitions.

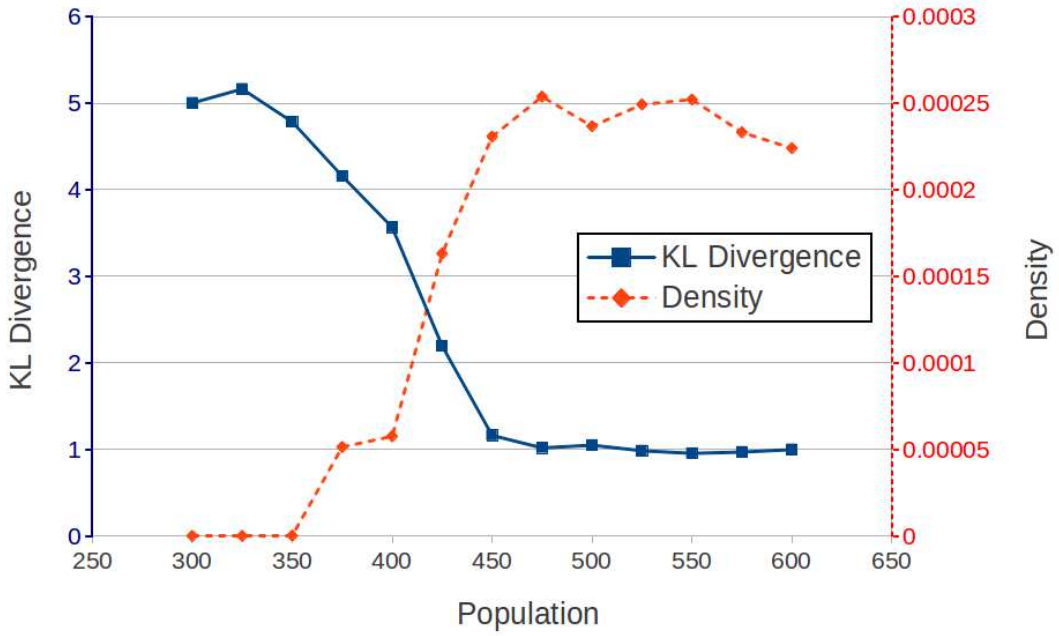


Figure 5.14: Density and Kullback-Leibler divergence measures as function of yellow population after 2000 time ticks. For a fixed population of red particles (50), we vary the population of the the yellow swarm (from 300 to 600).

5.4.4.3 Effect of lengthening run time

Certain population mixes have been shown to apparently never divide, others have shown that propensity to divide is dependent, at least to some extent, on the initial positions of the particles in the swarm. Clearly a claim that a swarm would never divide is not experimentally provable but we may extend the number of iterations that the swarm runs for in order to test whether an undivided swarm may of some occasions simply require more time. We repeated the previous investigation but allowed the model to run five times longer: now 10000 steps. There remains a range where division is uncertain. This region occurs for larger yellow populations. Yellow populations less than 425 always result in division. Those greater than 475 never divide. Populations between these limits may divide. Fig. 5.15 confirms this observation in that the step up in density occurs at higher yellow populations. Figure 5.5 shows the final positions of each population mix (up to yellow population equal to 500) at the end of each run.

Swarms that feature too few red particles may never divide. Similarly if the population of the yellow sub-swarm is increased then the resultant swarm appears to become increasingly stable. A small number of ad-hoc tests were done using much larger yellow sub-swarms and longer duration executions. These suggested that with relatively small red swarms the whole swarm may be unable to divide. However when

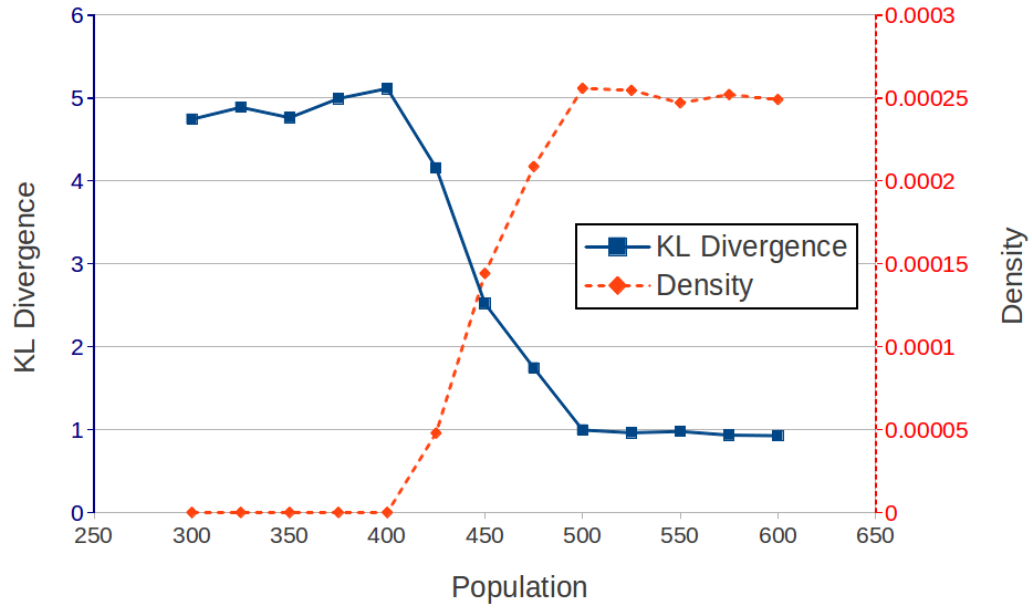


Figure 5.15: Density and Kullback-Leibler divergence measures as function of yellow population after 10000 time ticks. For a fixed population of red particles (50), we vary the population of the yellow swarm (from 300 to 600).

the red population was increased (to 180) the swarm which largely appeared to be stable would occasionally eject a small cluster of yellow particles. This suggests that such a swarm may slowly lose yellow particles until the remaining yellow cluster is small enough to show the normal division behaviour.

5.4.5 Robustness under parameter variation

As a full search of the parameter space is currently too onerous, a different approach was chosen. Single parameters were varied whilst keeping the remaining parameters unchanged. This approach allows us to explore a slice through parameter space. The onset and cessation of the target cell division behaviour is established for each parameter slice that is looked at.

5.4.5.1 Variation of neighbourhood radius

The cohesion, alignment and avoidance interactions use the concept of neighbourhood radii. For a given particle, only those other particles that are within its neighbourhood radius are used to impact the state update. The impact of varying this neighbour radius is considered here. A swarm (with a yellow:red population mix of 300:50) known to show cell-division-like behaviour is used throughout. The neighbour radius

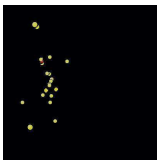
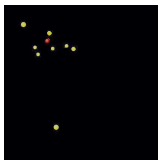
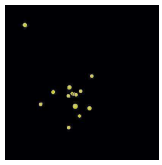
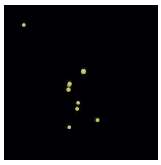
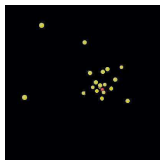
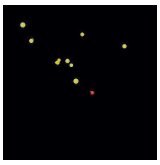
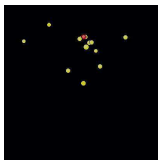
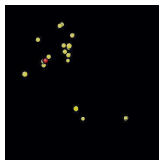
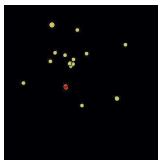
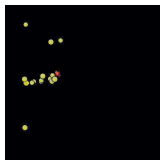
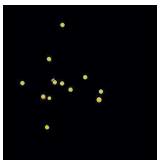
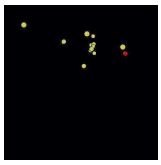
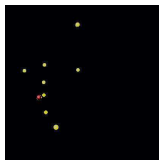
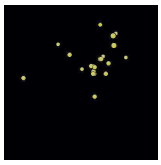
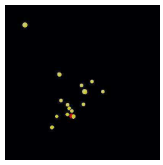
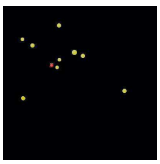
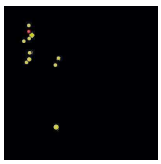
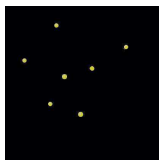
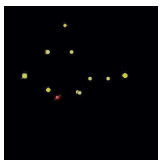
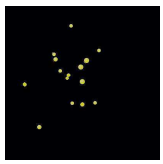
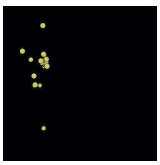
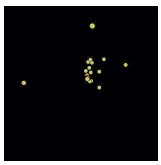
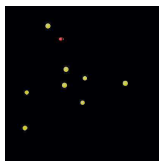
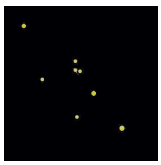
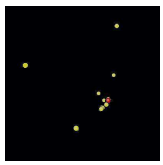
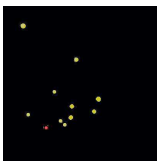
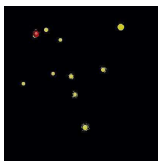
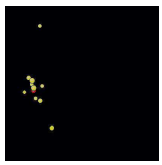
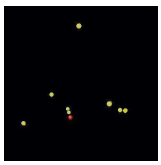
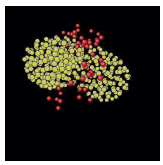
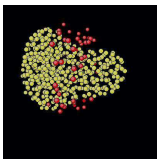
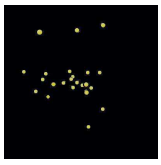
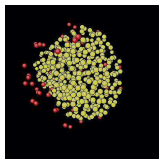
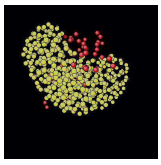
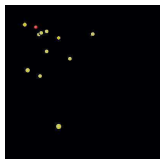
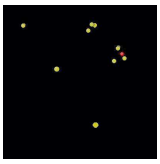
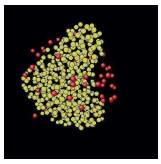
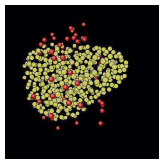
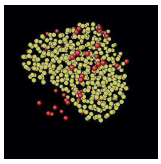
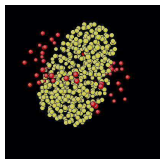
Yellow population	Run 1	Run 2	Run 3	Run 4	Run 5
300					
325					
350					
375					
400					
425					
450					
475					
500					

Table 5.5: Population dependence on cell division behaviour. Red and yellow populations are varied. The spatial state of the final frame of each run of each population mix is shown (after 10000 time ticks). Red population is set to 50 particles, yellow population is varied from from 300 to 500 particles in steps of 25). There were five repetitions.

of each sub-swarm is varied independently. The red radius is swept from 50 to 300 in steps of 25, and the yellow radius is varied over the range 10 to 40 (step 10). Each swarm is run for 2000 time ticks. When the yellow particles' neighbourhood radius is greater or equal to 30 the red particles travel inside the yellow swarm rather than forming an external ring and do not cause the yellow swarm to split. When the neighbourhood radius is 10 the yellow swarm simply disintegrates rather than showing the more ordered division process. It thus appears that a yellow neighbourhood radius of 20 is a sweet spot for the cell division behaviour. A red radius of 125 and above appears to be necessary for division to occur. Additional tests with a finer gradation of yellow neighbourhood radii suggest that at a distance of around 10 there is a quick and catastrophic division of the yellows into many small clusters. The red particles form a single cluster apparently disinterested in joining the others. From about 13 to 25 cell division occurs with the behaviours already documented. The lower the yellow neighbourhood radius value the easier, or faster, the divisions appear to occur. Yellow radii of 28 and above lead to swarms where the red particles and yellows exist either as a single cloud or with the red particles held within the yellows. Fig. 5.6 shows example swarm states at the end of each run for a range of the radii combinations studied.

5.4.5.2 Variation of avoidance (c_3) and cohesion (c_1) parameters

The aim here, as with the previous sections, is to sweep across a slice of parameter space and examine how the emergent behaviour of our swarm changes. Here we look at two of the kinetic interaction parameters: avoidance and cohesion. First we test the behaviour variation due to changes in each sub-swarm's avoidance parameters. Red avoidance values are changed over the range 5 to 40, and yellow between 10 and 60. The ranges are chosen following earlier ad-hoc tests that suggest these ranges will contain the cell-division-like behaviour we are interested in following.

Subsequently we test the behaviour variation due to changes in cohesion values. Both the red and yellow cohesion values are varied over the range from 0.2 to 1.0. A number of different behaviours were noted. A swarm with a 300:50 yellow to red mix was used as before. A number of behaviours were apparent:

- Cell-division-like: Our target behaviour is characterised by the red ring squeezing the yellow until division occurs. The divided parts move apart, swiftly, with the red particles rejoining the larger group.
- No division: The swarm remains a coherent entity.

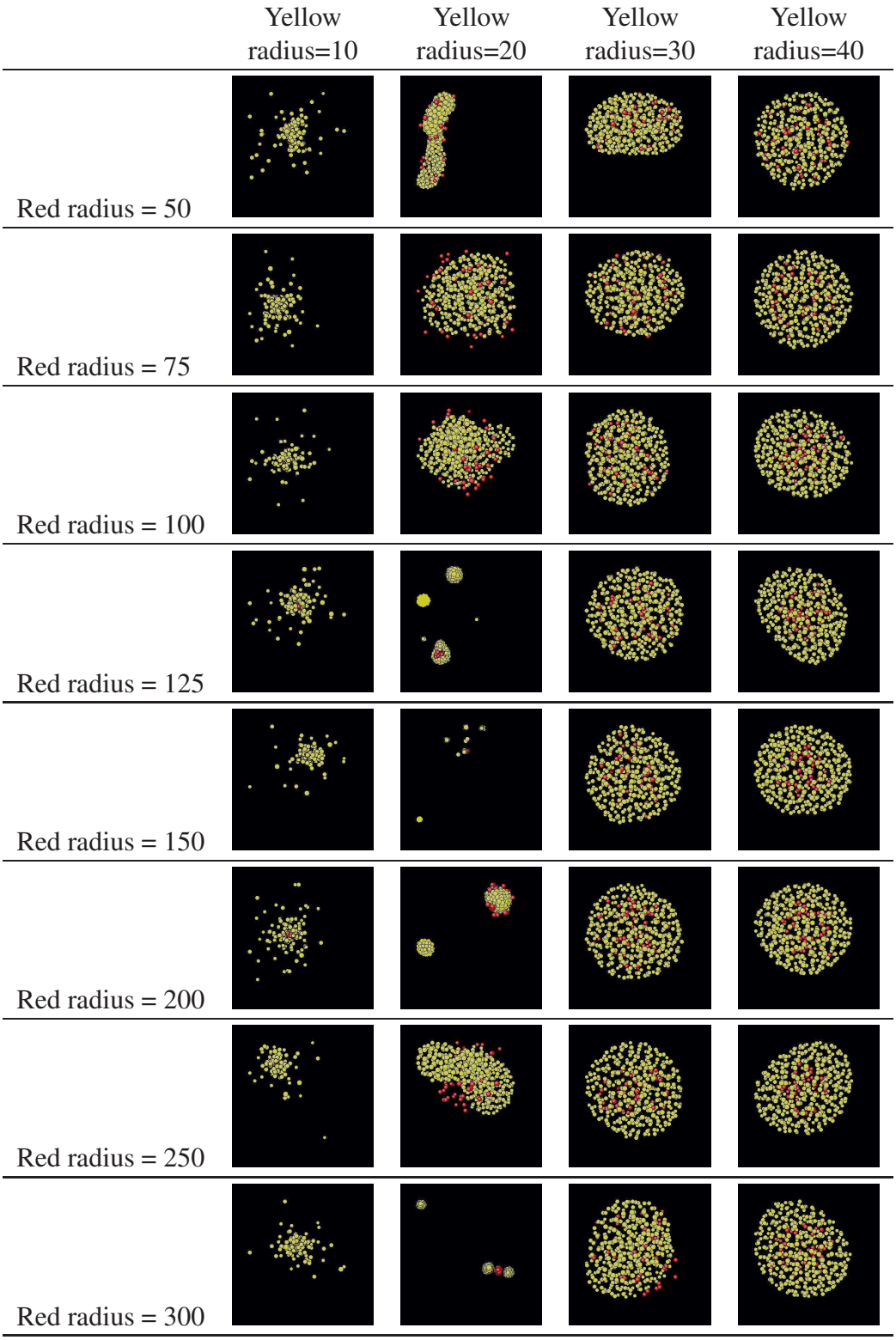


Table 5.6: Effect of red and yellow neighbourhood radii on division dynamics. Final spatial states of a number of sample runs are shown. Each run was of 2000 time ticks duration). The population of each species is held constant (300 yellow: 50 red), all parameters are kept constant except the neighbourhood radius of each species. A minimum radius of around 125 for the red species appears to be needed for cell division behaviour to manifest. The behaviour shows for a small but finite range of yellow radius values, centred around 20. Larger yellow radii do not yield division behaviours, smaller values lead to disorderly swarm disintegration.

- Yellow particles disintegrate: Swarm falls apart with no apparent encouragement from the red particles

- 2D-like division: Division may occur with reds squeezing from the outside or *eating out* from inside the yellow sub-swarm. Following division there is little or no separation of the split clusters and the red particles may not rejoin a yellow cluster.

Several of these behaviours, notably yellow particle disintegration and our cell-division-like behaviour, might not be clearly distinguishable via the use of measurements alone, so each run's categorization was confirmed by visual inspection. A single run of each permutation was made. All runs lasted 2000 time ticks. Table 5.7 shows the results for avoidance variation and Table 5.8 shows the results for cohesion variation.

Cell division behaviours exist over narrow ranges of both these parameters. Cell division behaviour of the sort we have been looking at is thus very sensitive to the values of both avoidance or cohesion parameters. It appears that for small yellow avoidance values ($c_3 \leq 40$) the red avoidance value needs to be around half that of the yellow value for any division to occur. Given the parameter set of the swarms, it appears that larger yellow cohesion values are needed to stop the yellow swarm from disintegrating. Perhaps above this level (around 0.6) the yellow swarm requires a greater 'pull' from the reds to begin to divide.

The cell-division-like behaviour exists on an edge between regions that show either no division or the more static 2D-like division behaviour. Within the avoidance parameter slice of this swarm system's parameter space it was possible to utilise our measures to distinguish cell-division-behaviour from the other behaviours. Thus the onset and cessation of our target behaviour could be explored in an automated manner. Sweeping through pairs of avoidance parameters with a finer granularity than was achieved with the manual checking was done. Here we are not interested in the subtleties of the type of division processes. Instead we simply detect any division, manifested by an increase in the KL divergence value for our swarm. We set a threshold of 2.5 for this, based on manual observations. We made increments in the avoidance parameter value for red particles from 5 to 40. For each red value the lower limit of the yellow avoidance parameter was estimated from the the earlier observations (Tab. 5.7). The swarm was run and if no division occurred in 2000 iterations, then the yellow avoidance parameter was incremented and the swarm rerun. This was repeated 10

Avoidance value	Yellow = 10	Yellow = 20	Yellow = 30	Yellow = 40	Yellow = 50	Yellow = 60
Red=5	3D	2D+	2D+	2D+	2D+	Y
Red=10	0	3D	2D+	2D+	2D+	2D+
Red=15	0	0	0	3D	3D	2D+
Red=20	0	0	0	0	3D	2D
Red=25	0	0	0	0	0	Y
Red=30	0	0	0	0	3D	Y
Red=35	0	0	0	0	0	Y
Red=40	0	0	0	0	3D	Y

Table 5.7: Division types as function of avoidance parameter, c_3, for a selection of the parameter variations tried. Categories are: '0' — No division seen, reds may form toroid around yellows. '3D' — Division seen, behaviour was characteristic of the standard 3D cell division. '2D' — Considered the same as 2D case: inside out split but clusters are largely static after split. '2D+' — as the 2D case but separate groups are more dynamic after split. Reds may be drawn in. 'Y' — Yellows disintegrate into small groups, reds form their own cluster.

Cohesion values	Yellow = 0.2	Yellow = 0.4	Yellow = 0.6	Yellow = 0.8	Yellow = 1.0
Red=0.2	Y	Y	Y	0	0
Red=0.4	Y	Y	3D	0	0
Red=0.6	Y	Y	0	0	0
Red=0.8	Y	Y	3D	0	0
Red=1.0	Y	2D	2D	3D	3D

Table 5.8: Division types as function of cohesion parameter, c_1, for a selection of the parameter variations tried. Categories are: '0' — No division seen, reds may form toroid around yellows. '3D' — Division seen, behaviour was characteristic of the standard 3D cell division. '2D' — Considered the same as 2D case: inside out split but clusters are largely static after split. 'Y' — Yellows disintegrate into small groups, reds form their own cluster.

times in order to gain some feel for the variability arising from particle initialisation. We plot this data in Fig. 5.16 as a box-plot as this nicely visualises the width of this division behaviour edge. This plot shows the same pattern: with low values of red avoidance the yellow avoidance needs to increase approximately linearly for division to occur. With higher red avoidance values the required yellow value appears broadly to be a constant. It is interesting to note that where these phases come together there appears to be a need to have a higher value for the yellow avoidance parameter value than either of the two zones would suggest. This is perhaps suggestive that there is more than one mechanism at play here: responsible for the upper and lower zones respectively. In the middle these mechanisms may be interfering and yielding the need for higher yellow avoidance values.

5.4.6 Robustness to adding other particle species

Throughout this chapter the swarm chemistry behaviour studied has been referred to by the phrase cell-division-like. This requires some explanation. In Hiroki Sayama's original swarm chemistry explorations structures and behaviours emerged that invited metaphorical descriptions. Concentric circles suggested to the viewer cell membrane and nucleus structures. It is easy to ascribe terms diatom, jellyfish etc. to other examples. These names are a useful shortcut in describing the emergent properties of different swarms. In what sense then do we apply the description *cell-division-like*? The swarm is a very simple one and and does not capture the complexity of what is seen in even a single biological cell, but that is the nature of a model: a simplification that captures an essence of the entity being modelled.

Lutkenhaus (2008) outlines the sort of division process we draw comparisons with here. Prokaryotes are (usually) single celled organisms. They are somewhat simpler than eukaryotes in that their nucleus is not encased in a membrane, and they lack other components seen in eukaryotic cells e.g. mitochondria. Prokaryotic cells divide via several mechanisms: binary fission, fragmentation or budding. The binary fission process is the mechanism to which we draw an analogy. Like many cellular mechanisms, this fission division process is mediated by a range of proteins. Processes in cells are frequently mediated by proteins. During the fission process a protein called FtsZ plays an important role. The FtsZ protein molecules migrate to the equator of the cell where they form a polymer chain referred to as the Z-ring. As the ring forms about the cell it constricts the cell initiating the division stage. Normally this

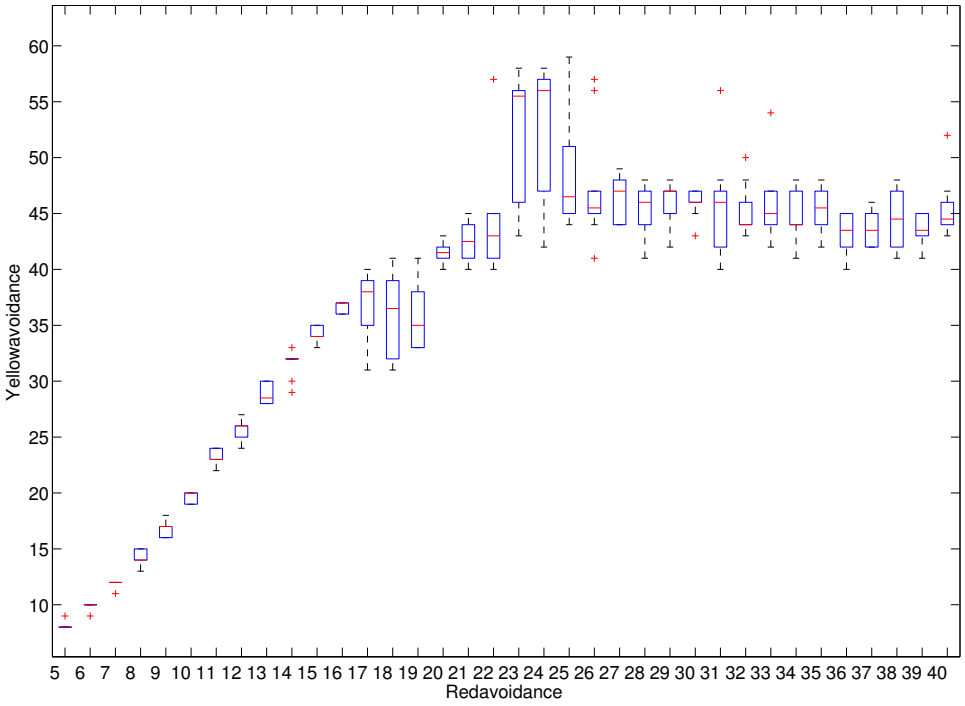


Figure 5.16: Detail of *edge* of division as function of avoidance parameter, c_3. Table 5.7 suggests that the edge between division and no division behaviours as one sweeps through combinations of avoidance parameters is where the 3D division behaviour occurs. Here we explore this edge. For each value of the red avoidance we scanned through the yellow avoidance parameters using the KL divergence measure (threshold set to 2.5 by observation) to capture when a combination led to a division behaviour. This was repeated 10 times for each red avoidance value to allow us to capture a feel for the variation in onset of division behaviours. We visualise this as a box-plot. Swarms that show no division exist below the plotted points.

spc	rad	spd	msp	c_1	c_2	c_3	c_4	c_5
3 (pink)	124.4	13.63	40	1	0.8	98.5	0.32	0.84

Table 5.9: Parameter values for the third species. When added to our normal pair of swarms cell division continued to occur. Headings are: spc = swarm species, rad = neighbourhood radius, spd = normal speed, msp = maximum speed, c_1 = cohesion, c_2 = alignment, c_3 = avoidance, c_4 = whim, c_5 = speed control.

Z-ring provides a scaffold that many other proteins can connect to and contribute to the process. However in some simple bacteria (those lacking a cell wall) those additional components are missing whilst the FtsZ proteins remain present. This suggests that the FtsZ protein on its own is sufficient to initiate binary fission. As the proteins polymerise they form filaments. These are usually are too short to construct a complete ring. It is thought that constriction on the cell arises via the lateral interactions between the filaments. In our model the ring of red particles forms spontaneously simply via the kinetic interactions between all particles in the swarm. Whilst cellular mechanisms are much more complex than our kinetic model, it is interesting to note filamentary interactions are suggested as a possible part of the process. In real cellular fission one suggested cellular division mechanism (Hale et al., 2001) describes how the oscillations of one protein (MinE) essentially defines the midpoint of the cell. This drives a second protein (MinD) to vacate that location. The MinD protein couples with a third protein (MinC) which acts to suppress the FtsZ protein from polymerising. Thus the cellular midpoint encourages the accumulation of the FtsZ protein and thus defines the location of division. Oscillatory motions have been shown (Sayama, 2012b). We show below that the cell division process is maintained even with the addition of an other species of particle. It may be possible to combine these in the future.

We succeeded in adding a third species to the cell-division swarm. The third species manifests as a shell of pink particles outside the other two swarms. Throughout the test the population was fixed at 92. The red and yellow populations were varied as before. The recipe for the third species is given in Tab. 5.9.

Figure 5.17 is an example frame from a three species swarm. The state at time tick 2000 is shown. The yellow sub-swarm population was 550. Previously two-species swarms with a yellow population of 550 resulted in a stable swarm showed no division behaviours. So whilst division is possible with the addition of other species the dynamics of the swarm is modified. It would appear that the inclusion of the third species results in a greater likelihood of cell division. In this swarm the pink particles

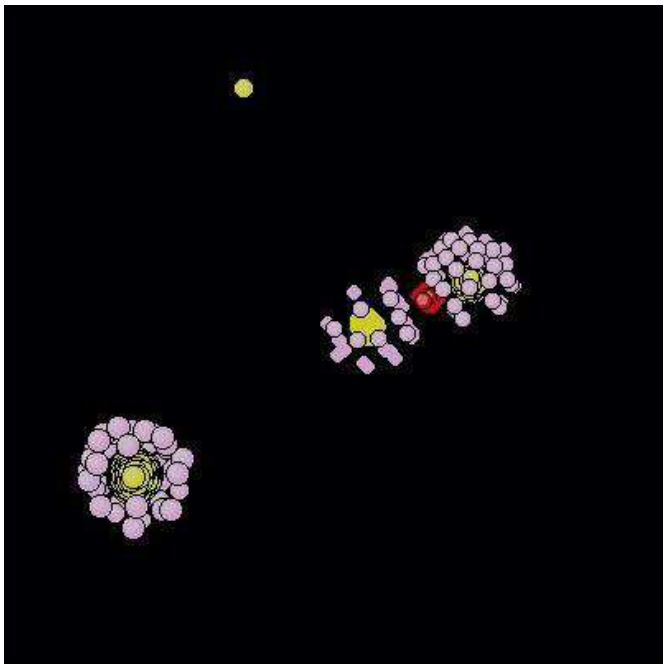


Figure 5.17: A swarm with three species still showing division behaviour.

are essentially acting as a catalyst on the cell-division-like process. We can repeat the population sweep through red and yellow sub-swarm sizes and measuring density and entropy as earlier. Figure 5.18 plots this. Divided swarms, again, show as lower density (higher entropy). Division occurs for larger yellow populations than without the pink particles throughout.

5.4.7 On origin of life models and the extensions allowing repeated division

Various structures and behaviours seen within Swarm Chemistry are reminiscent of biological forms. Obviously these similarities are in the eye of the beholder, the biological counterparts being much more complex in nature. Similarly the cell division process examined here is simpler than real division processes. However, it may serve as an analogy. Early cells would be subject to the push and pull of surface energies and ionic attractions and repulsions. Kinetic forces would play a role in their world. Simple mechanisms are thus interesting to explore to gain insight into early process possibilities.

Origin of life (OoL) models are attempts to explore mechanisms likely to been available to a pre-biotic Earth. They attempt to answer how an early Earth (around 3.5 billion years ago) replete with an array of simple minerals and compounds was able to

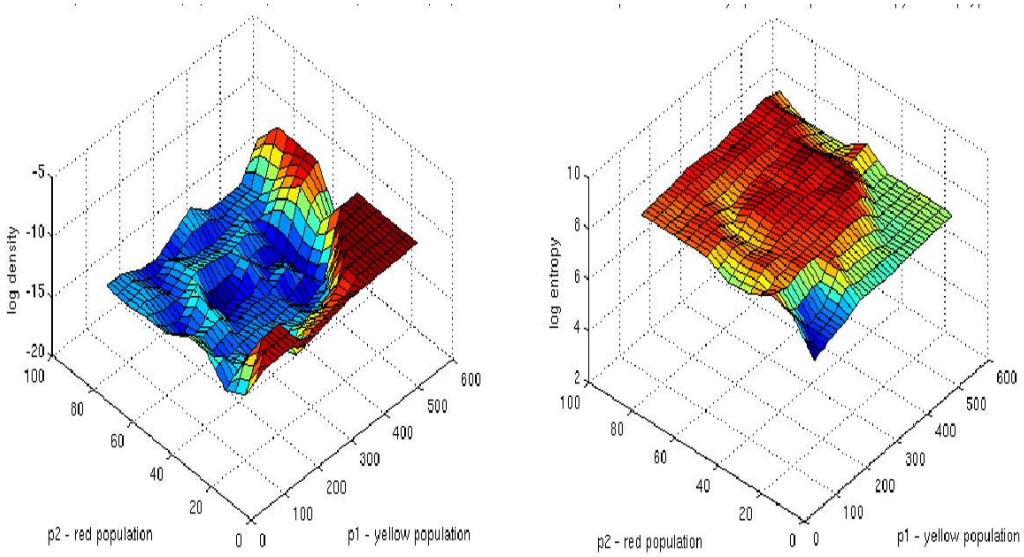


Figure 5.18: Density and entropy measurements for three species swarms. Red and Yellow populations are varied. The third, pink, species has a fixed population throughout. Cell division occurs for larger yellow populations than would have been the case without the addition of the pink species.

transition to one with complex lifeforms. How did chemistry become biology?

An initial problem is always one of definition: just what constitutes life? Proposed definitions seldom hold up to close scrutiny. A common approach is to suggest features that seem to be minimal requirements such as: metabolism, containment and heredity. A metabolism allows life to consume some form of fuel in order to maintain an ordered state against the universe's drive to disorder. Sooner or later this battle is lost and a living entity dies, it thus needs a means to procreate. In modern lifeforms an information store (frequently DNA) encodes a definition of the life-form and the processes required to make it. Earlier proto-lifeforms are likely to have involved less complex means of recreating themselves. Membranes containing chemical soups may have acted as early cells. The membranes preferentially retained their constituent molecules, admitting copies of the same molecules from the environment until the unit burst into two approximate copies. This scenario highlights the need for containment: chemical sets maintained in close proximity to promote activity and division processes to support some form of heredity.

Numerous suggestions and models have been proposed. Famously, the experiments of Miller and Urey in the 1950s showed that complex organic components could arise by sparking electricity through a mixture of simple gases: ammonia, methane, hydrogen and water vapour (Segré et al., 2001). This was suggested to mimic

the action of lightning in an assumed early Earth atmosphere. The problem remained that any complex organic molecules would be in very low concentrations and have a vanishingly low probability of further interactions unless a means to provide local concentrations i.e. some form of containment. Earlier, in the 1930s Oparin had suggested RNA-like molecules as the progenitors of modern life and proposed coacervates (small spherical droplets) as a suitable container (Lancet, 2003). Coacervates form spontaneously in certain dilute organic solutions. Hydrophobic interactions hold these small spherical droplets together. They possess osmotic processes allowing preferential transport of chemicals into the droplets.

It was this idea that Miller and Urey were exploring in their experiment. Other containers, for example, lipid bi-layers (appealing due to similarities with modern cell membranes) have been proposed over the years (Segré et al., 2001). Division of contained mixtures would then provide a means by which evolutionary selection could begin to take place. Such proto-replication would occur without the complex cellular mechanisms we currently see. Recent thoughts have studied life as sets of auto-catalytic equilibrium reactions within a container of some sort (Kauffman, 1996; Segré et al., 1998): autopoiesis arising purely from the self emergent properties of the system. All OoL theories tend to suffer from difficulties as a suitably high level of complexity is required before evolutionary mechanisms 'kick in' and drive the system to a level self-replicating complexity concomitant with life. Hypercycles, proposed by Eigen and Schuster (1978) suggests that interconnected sets of auto-catalytic sets can themselves be auto-catalytic. This provides, potentially, a mechanism whereby simple auto-catalytic sets of chemicals can lead to more complex systems. Further, hypercycles are invoked in the so called 'operator hierarchy' that see hierarchies in these interconnections leading to emergence on all levels of organisation (Jagers op Akkerhuis, 2008). There is a cross over between these OoL studies and some of the 'grand challenges' of artificial life (Bedau et al., 2000). This highlights a range of goals (both in vitro and in silico). A range of approaches are required: wet models; bottom up and top down approaches; and computational experiments. The aims are to explore (a) how something living arises from non-living components or systems, (b) what the limits of a living systems are, and (c) how life is related to mind machines and systems. An example of this crossover is the recent Los Alamos Bug project (Rasmussen et al., 2009) that aims to create a minimal protocell capable of replication. The model uses dissipative particle dynamics (DPD), essentially interactions between particles are considered as forces and particle acceleration derived from such force

interactions are the basis for the particle updates. These dynamics contains similar elements to those of swarm chemistry.

In our model, we saw that division is usually asymmetric. This would be potentially harmful for cell division, but may be of some use for early proto-replicators i.e. a simple mechanism that results in nearly even division of containers of chemical mixtures could be useful.

The cell division behaviour in the previous sections splits a cluster of yellow particles in two. Only one of those groups will subsequently divide again. This occurs as the red particles were shown to only associate with the larger cluster of yellow particles. In order for this division behaviour to be seen as a possible model for real world division we needed a mechanism that would allow any yellow cluster to potentially divide. Sayama (2012b) models each particle as expressing one parameter set drawn from a group of parameter sets. This group being termed a recipe. This formulation allowed a natural extension to evolutionary techniques to be applied. The expression of one parameter set in the recipe being driven by some fitness function. We choose a similar approach. Each particle possesses a recipe that contain both the red and the yellow parameter sets. Each particle may expresses one parameter set chosen from this recipe. We allow a small probability that any particle may change the parameter set it expresses. This is modelled as a biased equilibrium processes. Each yellow, on being chosen to pick a behaviour, will select changing to red with a 10% probability. Conversely each red if selected to consider which parameter set to express will change to the yellow parameter set with a 90% probability. This ensures a rough 9:1 mix in the population, but allows any cluster of yellow only particles to develop a red population.

This mechanism alone provides for each cluster to continue to divide over time. However, as groups do not tend to recombine the ultimate future for this approach is a dispersed swarm. We added a growth mechanism to allow clusters to increase in size. New particles would be created close to randomly chosen existing particles. This can be viewed as new particles being recruited from the environment. Fig. 5.19 shows some examples from a swarm that implements both the biased equilibrium and growth mechanisms. The swarm still tends to appear somewhat dispersed, however, there are still many groups that continue to divide.

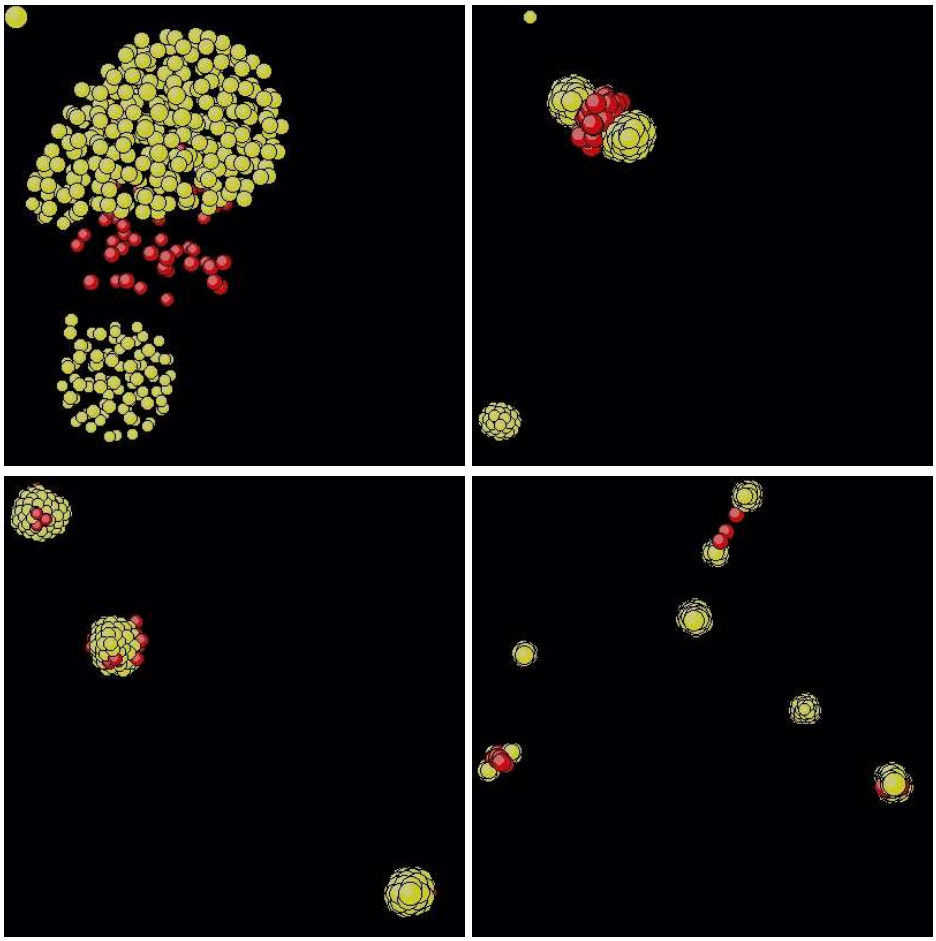


Figure 5.19: Repeated division. Top left shows first division. Top right shows the second division. Bottom Left shows multiple groups with red particles from the bias equilibrium process. Bottom right shows multiple divisions occurring.

5.4.7.1 KL divergence time evolution through repeated divisions

Earlier investigations had used Kullback-Leibler divergence as a measure to help us detect divisions in the swarm. With a repeating division this mechanism ceases to be so useful. We ran a growing swarm with the biased equilibrium process using the particle parameter sets as detailed in the previous sections. Figure 5.20 shows the Kullback-Leibler divergence measure (D_{KL}) as the repeated division swarm evolves. The swarm was run for around 6500 iterations. The numbers shown refer to the image frames captured, which in order to make them all five digits long start from 10000, so we run from 10000 to 16500. It also shows the swarm state at key moments in the experiment (typically frames occurring during sudden changes in the D_{KL} divergence measure). Initial divisions result in increasing D_{KL}, the later divisions tending to reduce the value. As we get more and more divisions the resultant swarm begins to look more like a dispersed swarm. Intuitively this suggests that the *distance* between the spatial distribution of the particles in the swarm and the evenly distributed comparator swarm used for our D_{KL} measure is reducing. The lowering of the value is therefore not a surprise. Separately calculating the measure for the yellow and red sub-swarms shows them to have similar values. This, again, is expected since the biased equilibrium process will cause the spatial distributions of the two sub-swarms to mimic each other. Following a division, a group of red particles will join the larger yellow group briefly making the red's distribution somewhat different from the yellow's. Subsequent stochastically driven changes in the expressed parameters sets will tend to even out these differences.

5.5 Discussion

We have demonstrated a repeating cell-division-like behaviour found within the system called swarm chemistry. This emerges from the repeated low level kinetic interactions between the particles in a heterogeneous swarm. Two particle types are required: individually these particles do not show the same behaviour. Prior to dividing, the red particles form a ring or toroid around the yellow sub-swarm. Without the yellow sub-swarm no such structure would occur. There are configurations where this appears a long lived phenomenon. Division occurs for a wide range of swarm sizes, but there appears to be a size above which the yellow swarm tends to stability. Some evidence was found that these large swarms may gradually lose yellow particles suggesting that

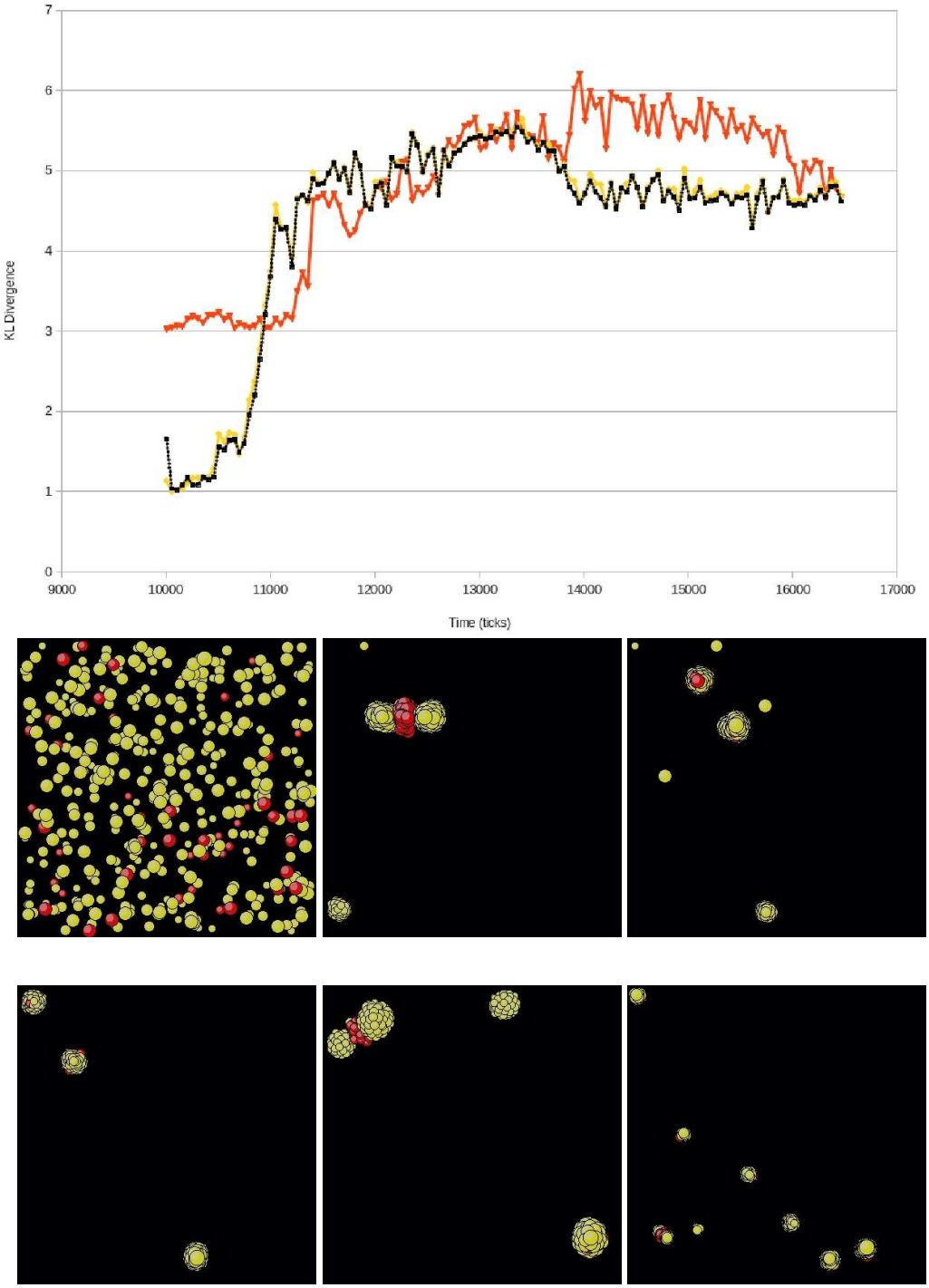


Figure 5.20: KL divergence time evolution of a repeating division swarm with key frames. Frame numbers are all five digits long, obtained by adding 10000 to the iteration number. So iterations 0 to 6500 correspond to frames 10000 to 16500. (top) The time evolution of the KL divergence measure for the combined swarms (black) and the two subspecies (red and yellow). (middle row, left to right) iterations 10000, 11046, 11404. (bottom row, left to right) iterations 13358, 13959, 16364

cell division may reappear if the swarm were to run for a greatly extended period. Following a division of the yellow cluster, the red particles were drawn to rejoin the larger of the two clusters and would lead to further divisions. The yellow cluster with no red particles would not divide. To achieve a swarm where all clusters would be capable of dividing it was necessary to add a biased equilibrium process to control the population split of the different particle types. This allowed all clusters to develop a suitable mix of the two particle types, but as there was no means for clusters to recombine the ultimate fate would be for the swarm is become increasingly dispersed. A growth mechanism was introduced, allowing small clusters to grow. Balancing the swarm growth and biased equilibrium process can be hard and the tendency for a swarm to fragment into smaller clusters remains. It would be appealing to improve the linkage between these mechanisms so that division would become more regularly periodic.

This division behaviour exists in a small but finite volume of the swarm chemistry parameter space. It was sensitive to the swarms' parameter recipes. The yellow neighbourhood radius needs to be in a narrow band. The red neighbourhood radius appears to have a lower limit, requiring larger values to show the division behaviour. The behaviour is present only across a narrow band of both avoidance and cohesion parameters. On one side of the band no division is observed. On the other side either an inside out division similar to that seen in 2D, or a spontaneous yellow disintegration that requires no interaction with the red particles, is observed. Locating other interesting behaviours that may exist in equally small volumes of this parameter space will require improved diagnostic tools. Density and Kullback-Leibler divergence were useful in guiding our manual observations for this specific behaviour, and were only able to assist rather than automate detection.

Additionally we have seen that the behaviours differ depending on whether the swarm is moving in two or three dimensional space. If the parameter values used in a 3D environment were used, unchanged, in a 2D environment then we observed an inside out division. This resulted in a relatively static set of divided groups. By modifying the parameter values used we were able to recapture the outside in division seen in 3D. This still failed to show the full dynamics seen in 3D. However, the fact that there are parameter mixes that show behaviour in 3D that matches that seen in 2D suggest the opposite may also be true.

The emergent behaviour has some similarities to prokaryotic Z-ring initiated binary fission. It was shown that additional particle species may be added to the swarm, which

do not inhibit the division process from occurring. Indeed the third species was shown to *catalyse* the swarm division. The system shown can be claimed to show containment (the particles exist in long lived groupings), heredity (via the combination of the use of parameter set recipes, the biased equilibrium process, and the fact that division represents replication), and an assumed metabolism (the energy needed for particles to speed up and slow down is assumed to be provided from the environment. As such this can be viewed as a simple model for Origin of Life theories.

Swarm Chemistry, as a system, has an elegant simplicity: from a few rules and the basic swarm heterogeneity emerge unexpected behaviours and structures. Natural extensions to the original rule set include the ability for particles to express one of a number of parameter sets. Sayama uses this approach to allow evolutionary mechanisms to be built into the system. I used this approach to support the biased equilibrium process used to allow repeated division. It would be interesting to extend the heterogeneity still further so that particles of one type were cognisant not just of the positions and velocities of their near neighbours but of the particular parameter sets that their neighbours were currently expressing. The desire for this stems from observing that the relationships between the various proteins that mediate prokaryotic binary fission contain complex interactions of a sort absent in swarm chemistry. It could be interesting to allow each particle species to have differing affinities with other species. For instance Species A would enforce the swarm chemistry rules fully when particles of type B are in its neighbourhood radius, but ignore those of type C if found at the same distance. Indeed having extended the species concept to include the idea of a recipe: one could envisage multiple parameter sets being used depending which species types were near. For example, species X might use parameter set 1 for all interactions with species Y particles, but would use parameter set 2 for particles of species type Z. It is not clear how these species to species affinities could be included in a similarly simple and elegant manner.

Structural analysis was driven by the position states of the particles. Position vectors fed the k-means clustering in order to track initial divisions of the swarm. Similarly the positions within the minimal cube containing the whole swarm was used to calculate the spatial entropy measure used to calculate the Kullback-Leibler divergence. Some investigations suggested that consideration of structure in the velocity state of the particles would be a useful direction to investigate. When a swarm chemistry swarm (either ours or many of those identified by third parties) develops over time what attracts the eye are the sudden shifts in behaviour: the to-ing and

fro-ing of an oscillator; the splitting of a cell; or the lurch of a chasing blob as it *senses* prey. We might expect that these points of interest would show changes or structure in the system's velocity space that might prove useful for automating the detection of such events. Some investigations have been begun. Figure 5.21 shows four different swarm types with plots of their Kullback-Leibler divergence, density, and mean speed measures. A homogeneous swarm's initial randomly initialised particles resolve themselves (in this case) to a spherical cluster. Initially this swarm has large variation in mean speed from one iteration to the next. Centered about the preferred speed of the red particles the kinetic interaction results in large swings in speed. As the spherical cluster forms the preferred speed remains the average, but the spread greatly reduced. It is as if the particles have found their place in the swarm where there is now minimal pushing and pulling on their neighbours. Something similar is seen in the jellyfish swarm, though the effect is less marked. The jellyfish swarm has three particle species which separate into four conjoined structures that move slowly, a hemispherical skirt oscillating slowly like the pulsing of a real jellyfish. There is still oscillation in the mean speed perhaps reflecting the slow oscillations of the jellyfish swarm. In the linear oscillator swarm we see an initial separation of its two particle species into concentric spheres, a linear to-ing and fro-ing starts up and after a brief period transitions into a rotary motion. Note: in two dimensions there is no third rotary phase. The mean speed plot shows these three phases. Changes in the spread of the mean speed again seem to suggest the onset of a new behaviour. Finally our cell-division-like behaviour swarm demonstrates similarly that a point of interest (the division) is marked by a change in the spread of the mean velocity. Here it appears as a spike, as the behaviour of interest is transitory. These results perhaps suggest that changes in the spread of a swarm's mean speed may indicate an event of interest in the evolution of the swarm. In the first three cases there appears to be an evolution from higher to lower spreads, indicating the onset of more stable behavioural regimes in the swarm. Our cell-division-like behaviour shows that increases in the mean speed spread can occur but perhaps only as a transient event in the system. Speed increases represent increases in energy of the swarm. It would be interesting to explore whether the tendency to decrease mean speed spread is a *law* of swarm chemistry.

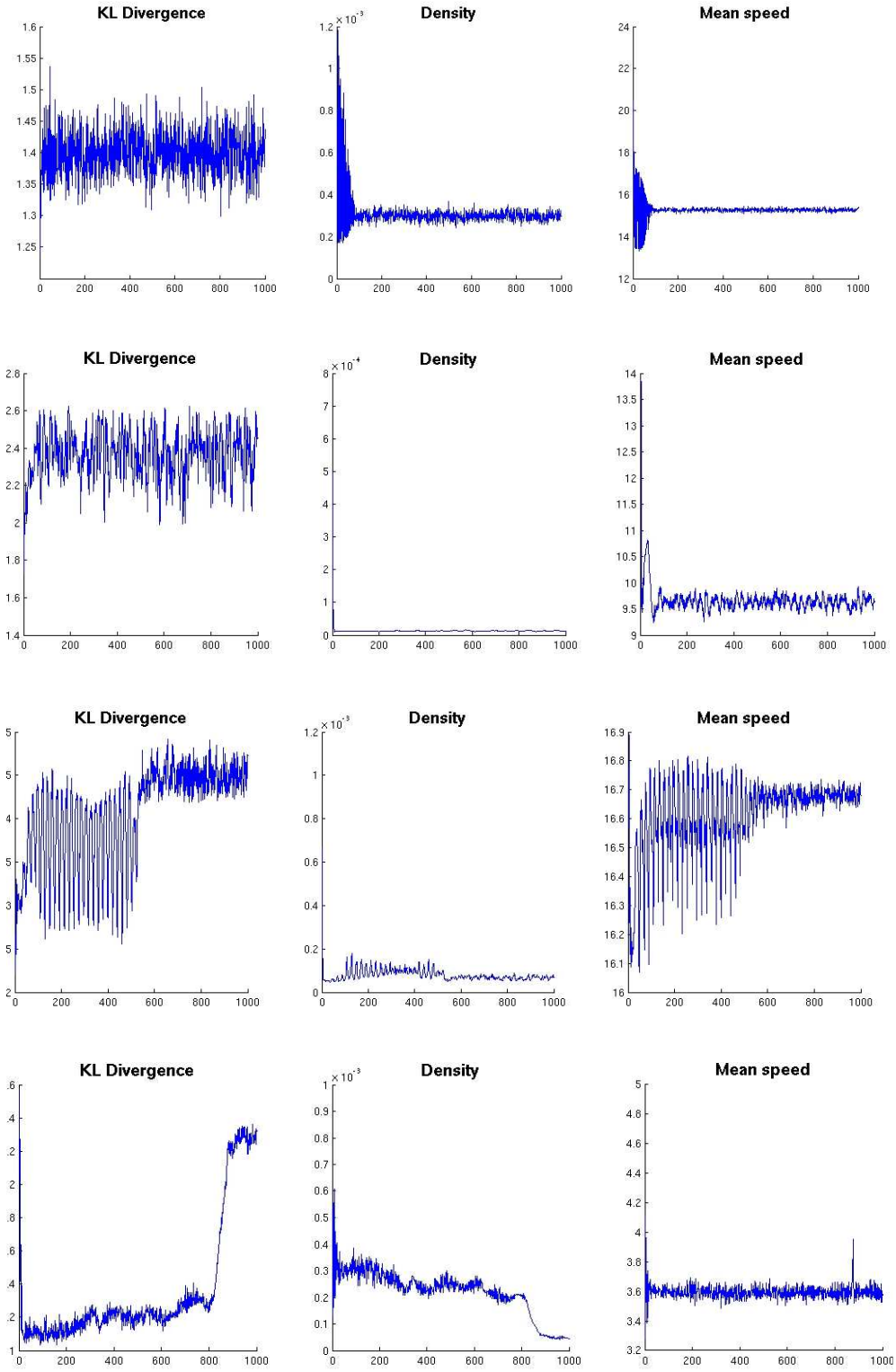


Figure 5.21: Changes in particle speeds may be of use to alert the user to changes in the behaviour of the swarm. Here we compare the D_{KL}, density and mean speed measures for various swarms. From top to bottom: a homogeneous swarm of our red particles, a jellyfish swarm, a linear oscillator, and our cell-division-like behaviour swarm showing its first division. The jellyfish and linear oscillator recipes were taken from Hiroki Sayama's website (Sayama, 2015)

.

Chapter 6

Contributions, future work and conclusions

Each of the preceding three chapters presented their own discussion and conclusions. In this chapter we summarise what we have presented and the novel findings that we have made.

6.1 Contributions and future work

6.2 Contributions

We have been concerned with exploring to what extent criticality can be shown to contribute to the emergence of behaviours in swarm systems. A chief motivation stems from the observation that in nature swarms can often appear ‘as of one mind’. A bird flock avoids a predator with a near instant change in direction of motion. Velocity changes appear to be correlated over distances much longer than local communication distances would suggest possible. This behaviour is reminiscent of the loss of length and time scales seen in critical systems. A few studies suggest real swarms may indeed be operating critically. This thesis studies swarm systems inspired by flocking behaviours and confirms the presence of that criticality. It is shown that criticality in swarm systems is beneficial.

At the start of this thesis we present four research questions which we can now answer.

6.2.1 Under what conditions does the particle swarm optimisation (PSO) algorithm exhibit criticality?

In answer to research question 1 chapter 3 shows that for PSO there exists a locus of the algorithm's parameter values that result in critical behaviour. On this locus the PSO swarm is poised between divergent and convergent states. PSO may be cast as a random dynamical system by rewriting the PSO update rules matrix form. As the swarm evolves its dynamics are determined by considering the infinite product of this stochastic matrix. The long term average behaviour of the swarm can be characterised by computing the Lyapunov exponent for the system. Exponents less than zero correspond to convergent swarms. Exponents greater than zero correspond to divergent swarms. We locate the locus of parameter values than correspond to stable swarms i.e. those with Lyapunov exponents equal to zero. This is shown in figure 3.5. Parameter pairs on this locus tune the system to the point of criticality.

This predicted locus of stability differs from the previous approaches. Empirical evidence is presented to support the correctness of the random dynamical system approach.

The location of the curve of criticality is shown to also vary depending upon the balance of α_1 and α_2 parameters, and with the distance between personal and global best locations for the particles.

6.2.2 Does the PSO swarm acting critically allow it to act optimally?

Empirically we look to see where in parameter space optimal results occur for many problem functions. A common pattern was observed over many problem functions. For negative values of ω there was a linear relationship between ω and α. For positive ω the relationship followed a curve. This shape matched the critical locus we predicted. We show, in answer to research question 2 that as the number of iterations executed increases the location of optimal solutions approaches our predicted critical parameter curve.

6.2.3 Can this knowledge be utilised in a practical manner?

We observe that real world problems tend to present as a black box function. The ability of the PSO swarm to locate solutions is dependent upon the nature of the

problem space, how long we will run the algorithm for and where in the parameter space we choose the controlling parameter values. Given knowledge of the first two items we can (at least in theory) intelligently choose how sub-critical we *need* the swarm to be to obtain optimal results. In practice this can still be hard. Thus a novel algorithm, CriPS, is proposed in chapter 4. This is shown to require no specific setting of parameter values and avoids stagnation. This is achieved by using a measure of the swarm's diversity within the problem space. Changes in this measure are used as a feedback signal to modify the PSO parameters. This results in the parameter values travelling across a line in PSO's parameter space that traverses the curve of stability. If the swarm is converging the parameters are adjusted to make the swarm behave super-criticality i.e. more like to diverge. If the swarm is diverging the parameters are adjusted to make the swarm behave sub-criticality i.e. more like to converge. The swarm is shown not to stagnate as a result.

CriPS is shown to outperform other simple PSO variants. In comparison with other metaheuristics this new algorithm performed more modestly. When trialled using the functions from the CEC2013 competition CriPS was shown to outperform all other algorithms for a single function. The overall performance may reflect the limitations of PSO as an algorithm rather than this new approach.

6.2.4 Can we show that other swarm systems also exhibit criticality?

In chapter 5 we show that critical behaviour in swarm systems is not restricted to PSO. A repeating cell-division-like behaviour is demonstrated within the system called swarm chemistry. This emerges from the repeated low level kinetic interactions between the particles in a heterogeneous swarm. This division behaviour exists in a small but finite volume of the swarm chemistry parameter space. It appears to sit on an edge of the parameter space between no division behaviours and markedly less dynamic division behaviours. And yet the behaviour was robust to population changes and even the addition of more species of particle.

This behaviour emerges from the repeated low level kinetic interactions between the particles in a heterogeneous swarm. There are configurations where this appears a long lived phenomenon. The process has analogies with the prokaryotic Z-ring initiated fission process and origin of life theories.

6.3 Future work

6.3.1 Future: PSO as dynamical system

Likely possibilities for future additional work include:

- As PSO runs it samples cost function values from the problem space. These can be used to determine the nature of the problem space, such that the parameter values of the algorithm can be automatically set to the values that will be optimum for (a) the problem being solved and (b) the remaining iteration budget? Recent work on geometry (Mersmann et al., 2011) and information (Malan and Engelbrecht, 2014) based measures of problem spaces may be a useful starting point for this. Alternatively the statistics of shifts in personal and global bests or indeed the diversity measurement feedback explored in the CriPS algorithm may be informative. Correctly performed this should allow PSO to tune itself to problem functions automatically so as to perform optimally.
- Other metaheuristics that employ stochasticity can be analysed as random dynamical systems. Does such analysis predict similar critical regions in their parameter space? If so, do empirical results in such regions show optimal behaviour?

6.3.2 Future: CriPS a near critical swarm

Likely possibilities for future additional work include:

- Practical investigation into the best value for the ε parameter in CriPS. A specific value would be nice, but it is more likely that a meta-process should determine the swarm diversity bursting behaviour and automatically adjust the ε value in order to tune the bursting to ensure near criticality.
- CriPS performs less well than the CMA-ES approach that won the CEC2013 competition. This may be due to PSO being inherently less good than CMA-ES to solve the type of problems in that competition. The winning algorithm still required a restart strategy to handle the problem of premature convergence. Application of swarm diversity feedback to this and other metaheuristics may be a useful and automatic means to avoid stagnation.

- The swarm diversity measure may be a useful diagnostic that could be employed to select critical PSO parameters directly. Indeed given the understanding of PSO criticality it may be that CriPS is potentially acting counter productively, i.e. that it is moving the parameter values away from the optimal critical region. Where it is performing well it is perhaps due to the average swarm behaviour being critical (in a pseudo- or quasi-critical) manner (de Andrade Costa et al., 2015). Future work should look at whether including both these processes is counter productive.

6.3.3 Future: Cell-division-like behaviour in Swarm Chemistry

Likely possibilities for future additional work include:

- Extensions of this model to naturally allow a better balance to the repeated division process. This requires that the growth and biased equilibrium processes are tuned. It would be more in keeping with the nature of these swarm studies that these processes should be driven by local information only.

- Nature studies have shown long correlation lengths in velocity changes within swarms. This appears to be the case in the behaviour examined here, but it would be useful to measure this explicitly for a range of swarm recipes that exhibit interesting behaviour.

6.4 Conclusions

As discussed in Section 2.8.3 there exist studies that suggest that motions in swarms are critical. Additionally it has been theorised that Lévy flights can provide efficient foraging strategies for certain animals. This thesis shows that criticality also plays a beneficial role in model swarm systems. The supporting evidence is constructed from (a) arguments around whether such systems feature characteristics that are known to be needed for systems to show criticality and (b) evidence in the form of system analysis or evidence of critical behaviours.

The systems examined feature characteristics that make it feasible for them to manifest critical behaviours. Each swarm is made of many units that act on local information. This is explicit in the Swarm Chemistry model. Each particle in the swarm is only aware of particles within a certain radius of it, and the swarm's full

extent may be much larger than this distance. In the PSO case, this is also true. On each iteration a particle experiences the cost function values sampled at own location. Whilst it retains a memory of its best location and receives swarm wide knowledge of the location of new global best solutions, these updates occur infrequently.

In critical systems a separation of time-scales is required for self organised criticality to exist. This may be connected with the existence of thresholds. The sandpile model explicitly encodes a threshold, such that the slow addition of grains releases fast flowing avalanches. In swarm chemistry there exists an interplay of push and pull forces, of velocity alignments, accelerations, decelerations and even a ability to randomly ignore the rules (occasionally). The effect on one particle accumulates over several iterations. Given these effects and the feedback in the system as each particle (within range) influences and is influence by its neighbours, it is hard to predict the accrued effects over a number of iterations. It is reasonable expect that thresholds may be breached. A particle pushed away remains within a radius of influence for a number of iterations. Once beyond that radius its influences vanishes. What we observe are discrete events: points in the swarm's evolution where the behaviour abruptly changes. A group divides, or changes direction, or expands suddenly. The appearance of these points suggests that a threshold has been passed.

In PSO there is a separation of time-scales due to shifts of personal and global best positions. Typically PSO makes these new discoveries rarely. When a new best location (personal or global) is found, the particle experiences a different force. This can be thought of as an external perturbation. The resultant evolving dynamic of the particle's future motion is the system's relaxation. However it is from the stochastic update mechanism that this swarm's criticality is derived. This is explicitly shown in Chapter 2. Analysis of the properties of the infinite product of the update matrices of PSO cast as a random dynamical system allows us to determine the locus of parameter values that result in a stable swarm. Within the locus the swarm will converge, outside it will diverge. Optimality is achieved when the swarm behaves critically. As such this flocking inspired swarm system confirms that swarms may utilise critical points to their benefit. In nature a few studies show that swarms can exhibit the characteristic loss of length scales in their velocity distributions. The cuckoo search (PSO derived) algorithm has employed the Lévy flight heavy tail distribution search strategy suggested to exist in forage strategies of many animals. In Cavagna et al. (2010) they discuss the role of noise in achieving criticality. In the case of starlings the noise is derived from their errant estimation of neighbour velocities and is tuned to

a point that results in a critical group response. Assuming the group response confers a survival benefit on the genes that give rise to this level of perceptual error, then evolutionary pressures will select for it. In our systems we lack this evolutionary pressure. The sources of noise may thus require us to tune the system. In PSO the stochastic update rules provide the noise and the $\omega : \alpha$ parameters the tuning. The Swarm Chemistry system is clearly tuned via its particles' parameters. Noise is less explicit. The Swarm Chemistry model includes a *whim* parameter, which provides for a small stochastically driven chance of particles to ignore their update rules. However, a greater source of noise may derive from the random initialisation of particle states.

In ferromagnetic material this is well known: a critical temperature results in the system being tuned to criticality. In model systems such as Per Bak's sandpile model there is no explicit tuning. Instead self organised criticality is invoked as a mechanism. The system falls into an absorbing state which is also its critical state. In swarms we don't require it to be self organised criticality so long as there is a benefit to the group.

The CriPS algorithm performs better than or equal to standard PSO. This is most likely due to its ability to avoid stagnation. Some configurations show approximate power-law distributed changes in swarm diversity suggesting that it can operate in a critical or sub-critical manner. As a mechanism it may be better employed with other metaheuristics.

Swarm Chemistry was shown to contain a cell division like behaviour. It is noted that this may be of interest to origin of life theories as the model is a very simple kinetic interaction system. The parameter set that results in this behaviour is a narrow but finite region of the system's parameter space, suggestive of a critical *edge*. The role of criticality in such mechanisms may have more to do with the nature of the dynamics. Other division behaviours seen see swarms divide sluggishly at best, often with no continued separation. Here, the two parts tend to continue to move apart after division. This suggests that the direction of movement in each half is rapidly transmitted and agreed by the particles in each sub-swarm. This is analogous to the loss of length-scale seen in dipole moment propagation in the Ising model.

Appendix A

Empirical PSO results: Performance of unconstrained PSO over CEC 2013 function set

In 2013 The IEEE Congress of Evolutionary Computation provided a set a problem functions to test metaheuristic algorithms. This CEC2013 set of function provided a range of challenges. The original intent was to provide a wide range of challenges to competing algorithms in order to judge which performed best. Here, we use the same set of functions in order to explore the true behaviour of particle swarm optimization (PSO). As discussed earlier we use a simple unconstrained implementation of PSO. The problems, here, are all defined in 10 dimensions, we use a 25 particle swarm and execute 2000 iterations: equivalent to 50000 function calls. We execute 100 repetitions of this set up for each ω:α pairing. We vary ω in the range -1.25 to 1.25 with increments of 0.1. We vary α in the range 0.125 to 6.0 with increments of 0.25. For each function we show two views of the results: the ω:α that result in the 5% best results; and a contour display of the best fitnesses obtained. We overlay the predicted critical curve. In all cases best results lie within this.

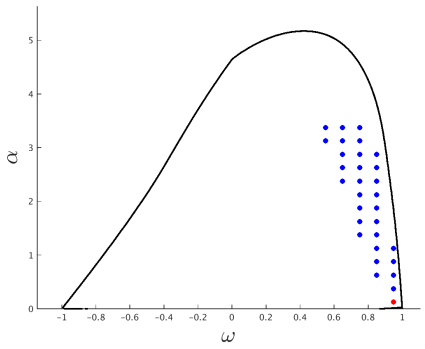
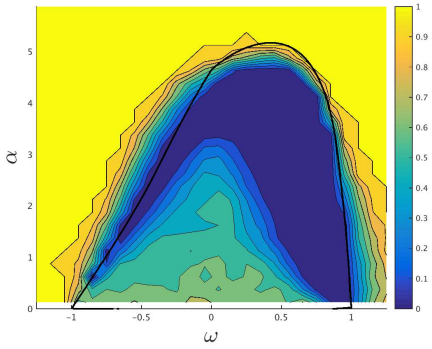
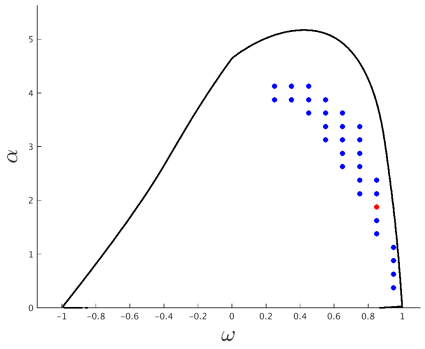
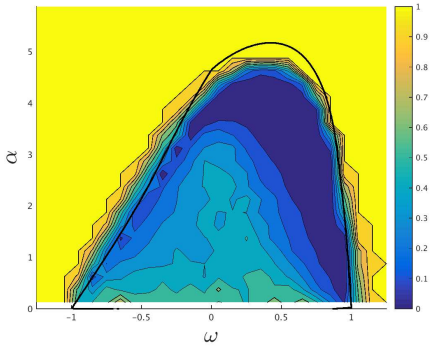
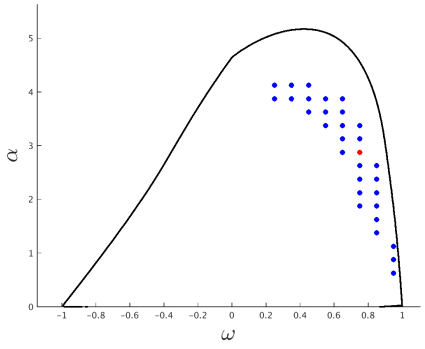
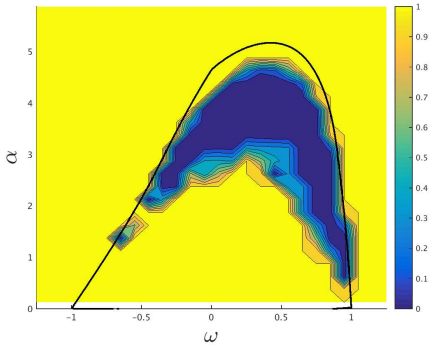
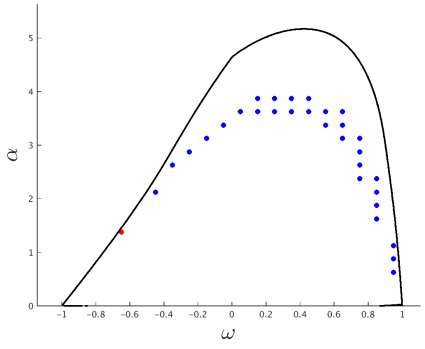
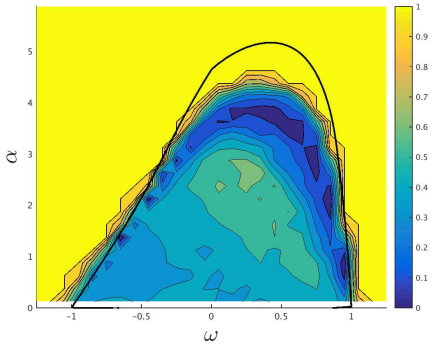
F1		
F2		
F3		
F4		

Table A.1: PSO performance against CECE2013 functions 1 through 4. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

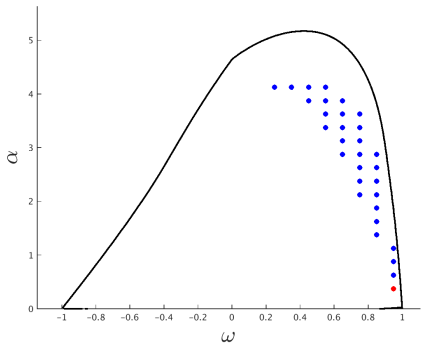
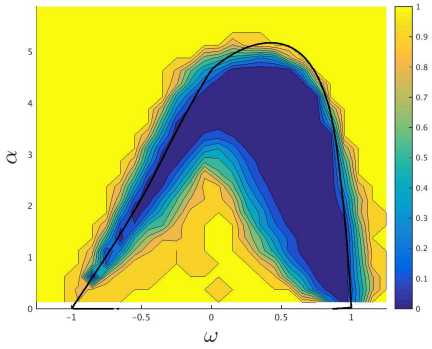
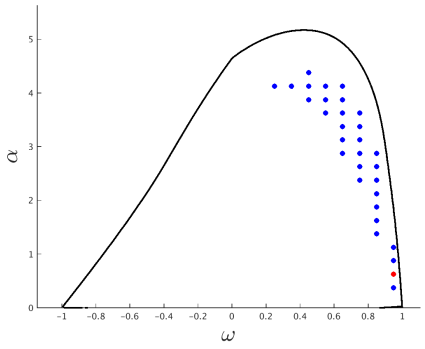
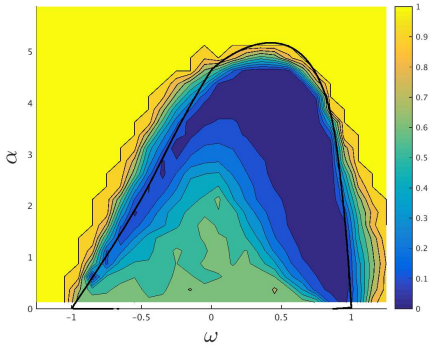
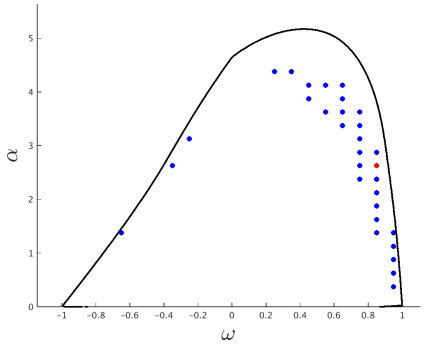
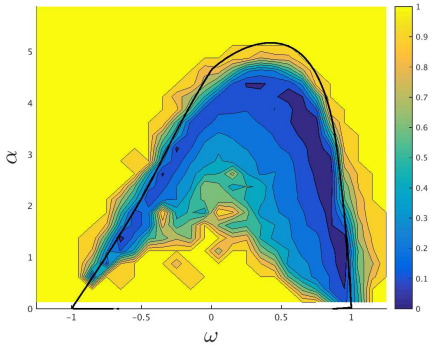
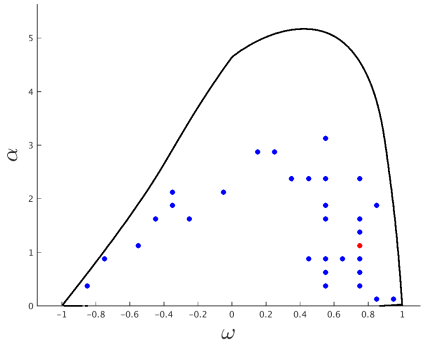
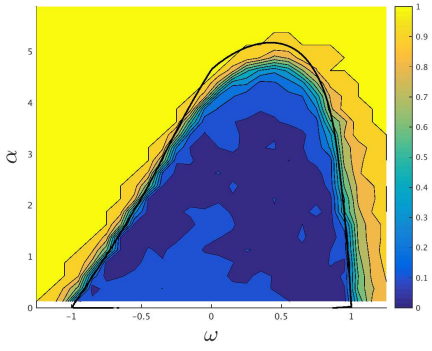
F5		
F6		
F7		
F8		

Table A.2: PSO performance against CECE2013 functions 5 through 8. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

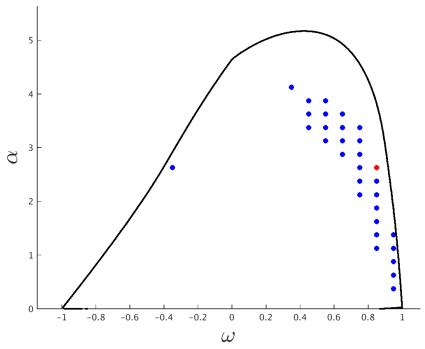
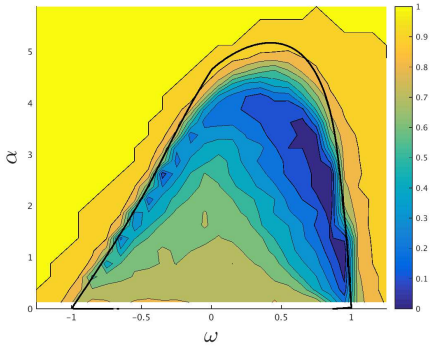
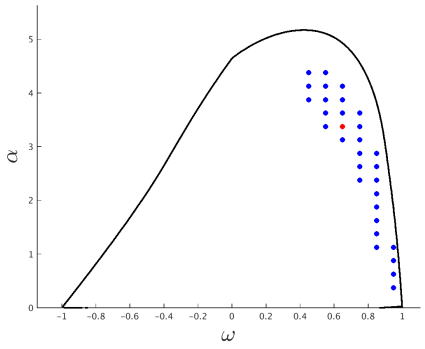
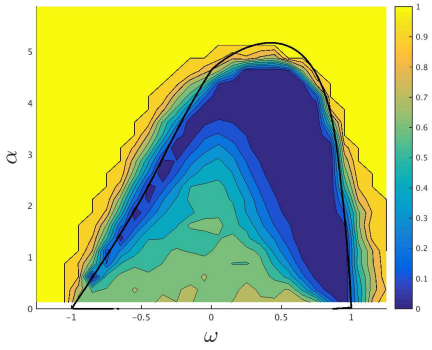
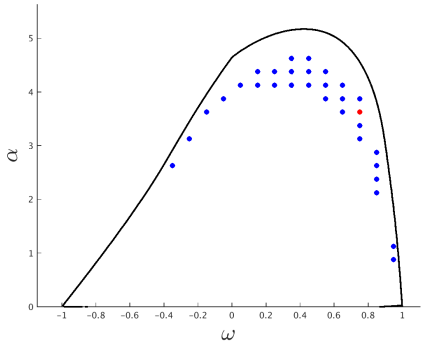
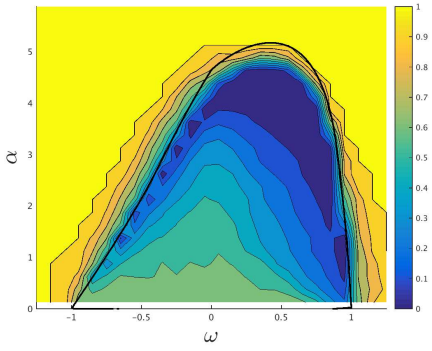
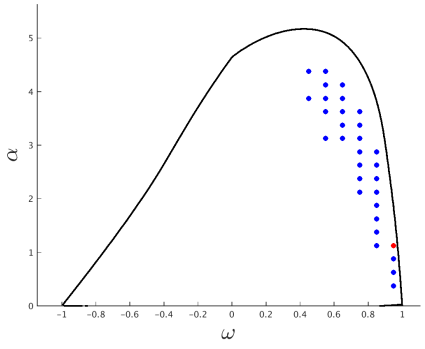
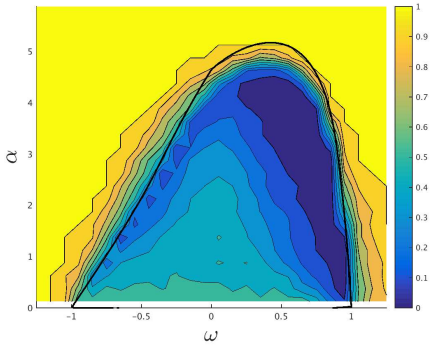
F9		
F10		
F11		
F12		

Table A.3: PSO performance against CECE2013 functions 9 through 12. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

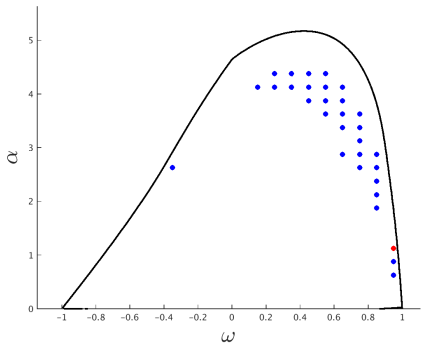
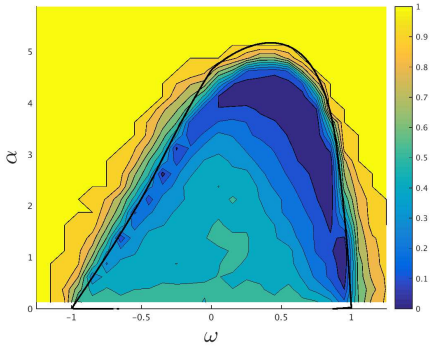
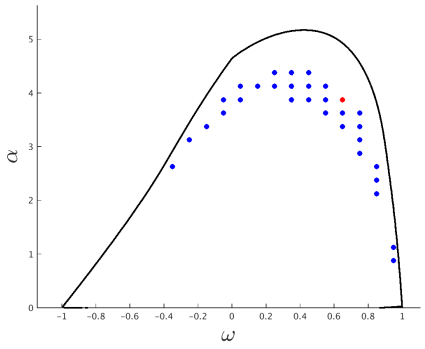
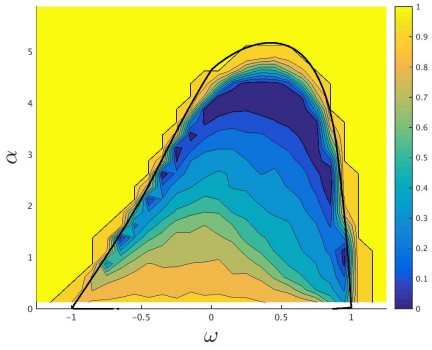
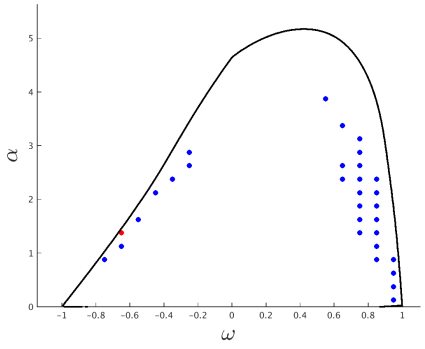
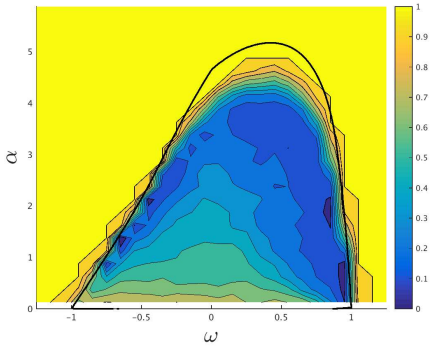
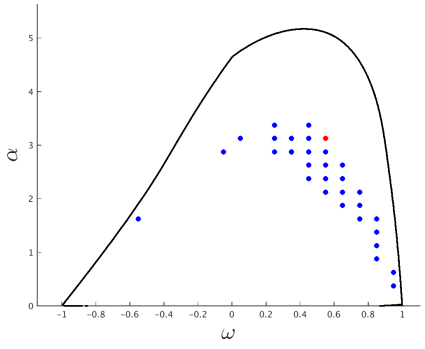
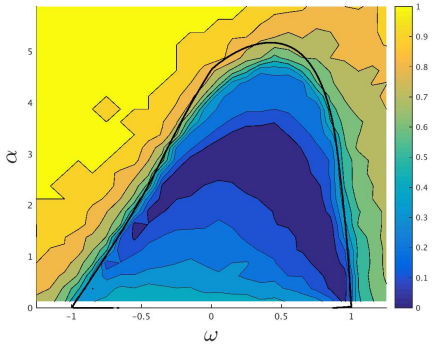
F13		
F14		
F15		
F16		

Table A.4: PSO performance against CECE2013 functions 13 through 16. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

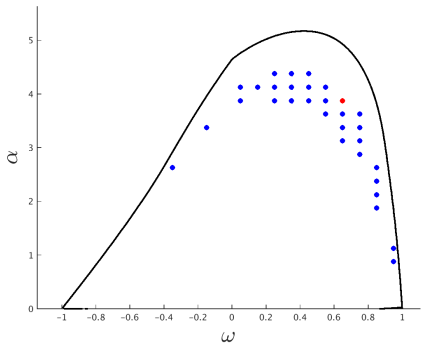
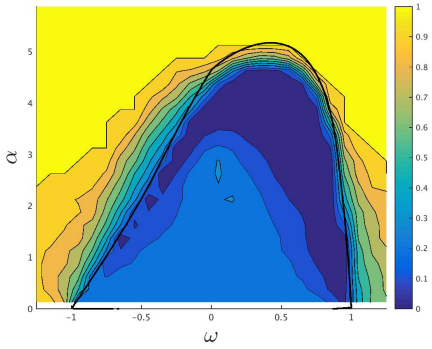
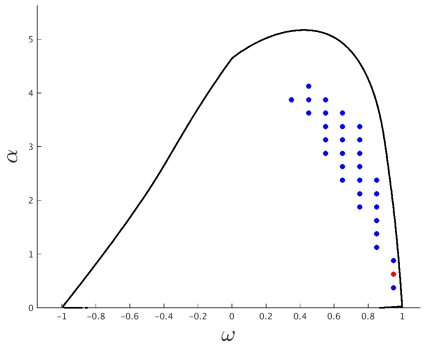
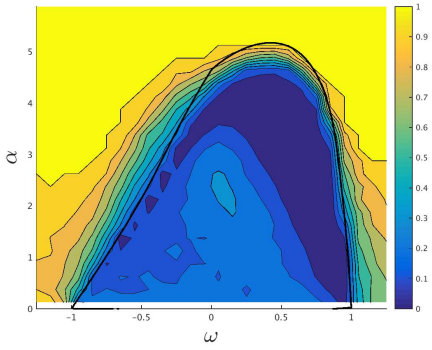
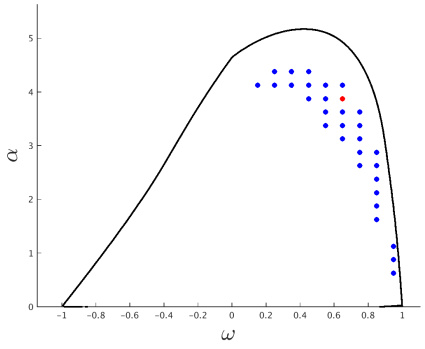
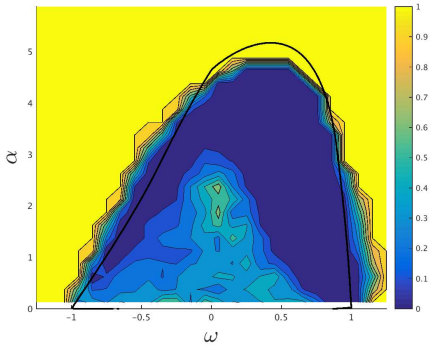
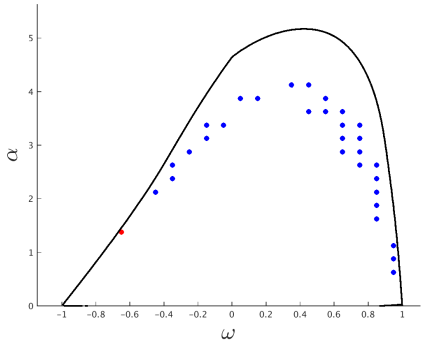
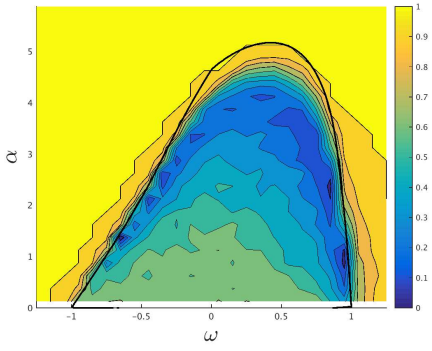
F17		
F18		
F19		
F20		

Table A.5: PSO performance against CECE2013 functions 17 through 20. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

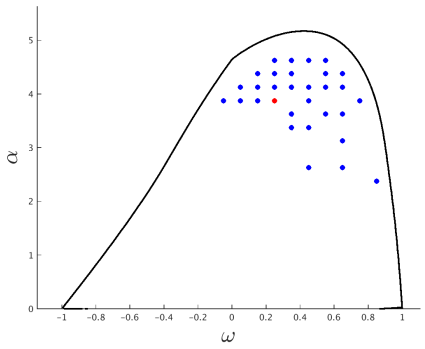
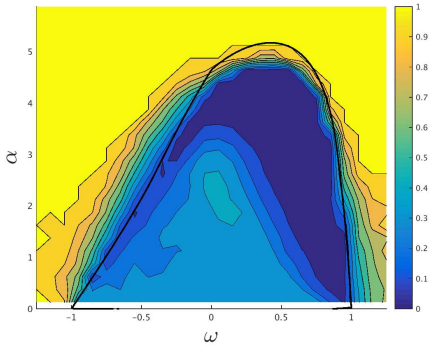
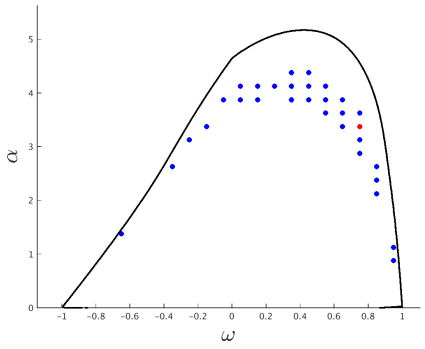
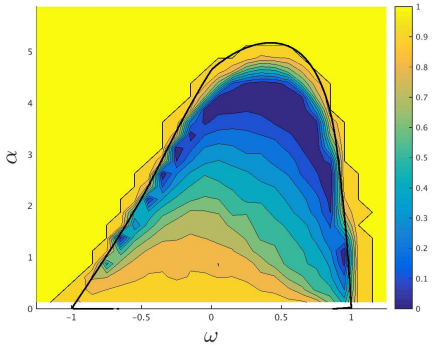
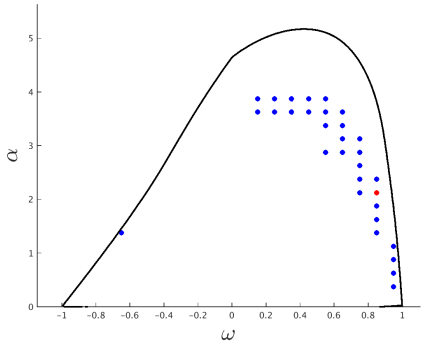
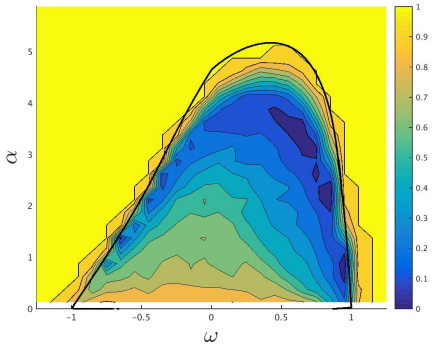
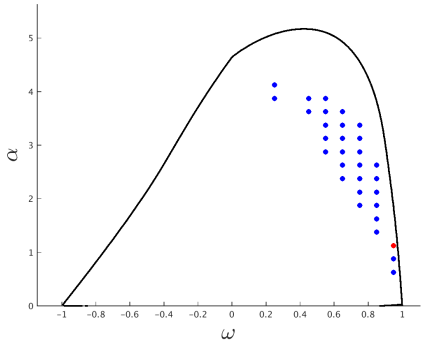
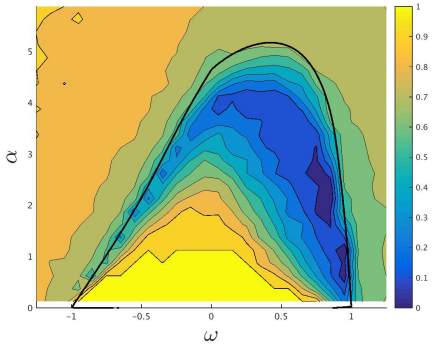
F21		
F22		
F23		
F24		

Table A.6: PSO performance against CECE2013 functions 21 through 24. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

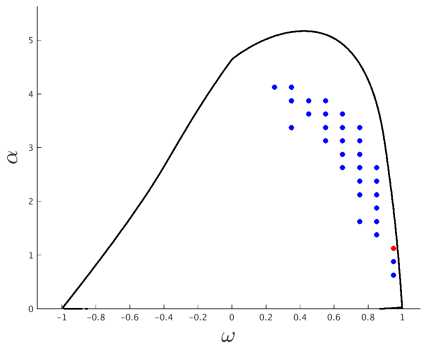
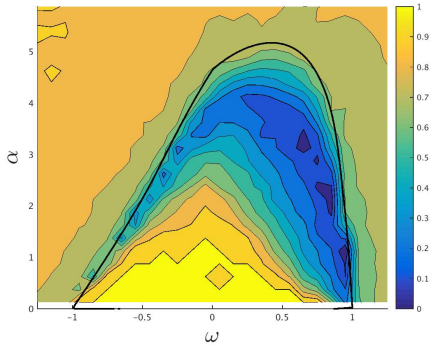
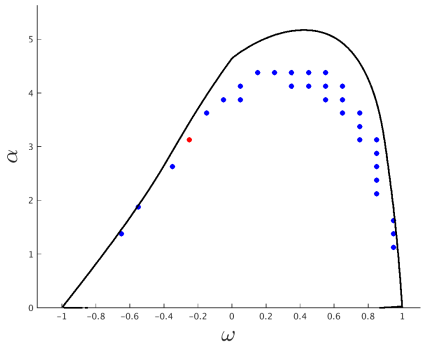
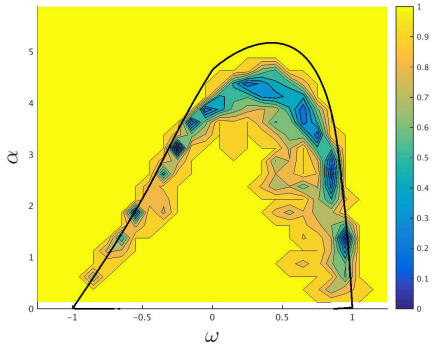
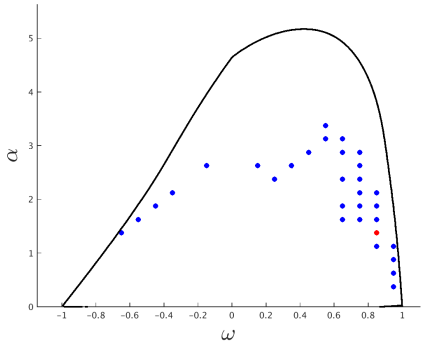
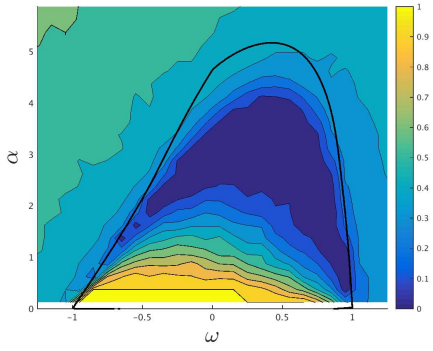
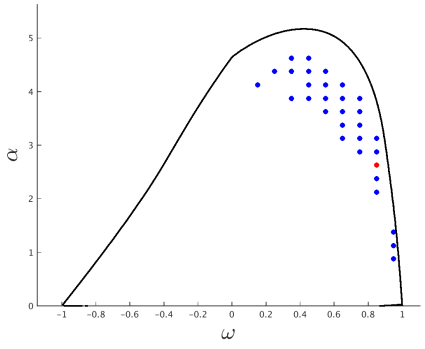
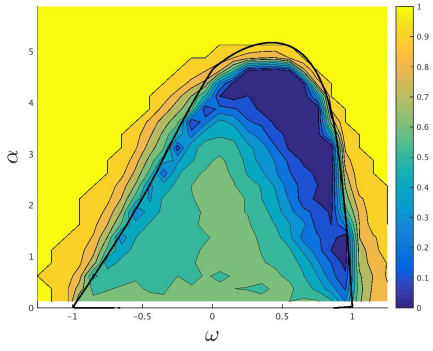
F25		
F26		
F27		
F28		

Table A.7: PSO performance against CECE2013 functions 25 through 28. Left hand figures show where the 5% best performing ω:α parameter pairs are located. Right hand figures provide a contour representation of performance over the whole ω:α parameter plane. The theoretical critical curve is overlaid. In all cases a 25 particle unconstrained PSO algorithm is used with 100 repeated executions.

Appendix B

Empirical PSO results: Performance over CEC 2013 function set as a function of $\left|\text{p-g}\right|$ distance

When running the PSO algorithm against the CEC2013 function set we measured the distance between the position of the global best location g and the centroid of the set of personal best positions p_i. This value at each parameter pair, was averaged over the length of the run. Subsequently we average over all 100 runs. Here, we plot the average $|$ p-g $|$ distance at each parameter pair.

We plot (a) the theoretical loci of criticality as derived earlier both for swarms where $g = p_i$ (dotted curve) and $g \neq p_i$ (solid curve), (b) The best 5% parameter pairs (small circles), and (c) a contour plot of the average $|$ p-g $|$ distances. We note that outside the theoretical locus there are pairs of parameters that will have led to expanding swarms. These are likely to have resulted in early termination of the algorithm (due to the swarm size or velocity exceeding our threshold).

For each function and parameter pair the PSO algorithm was executed 100 times. Each execution was allowed to continue for 2000 iterations. The swarm used was 25 particles in size. The particles were initialised in a $[-100, 100]^D$ volume.

The 5% of parameter pairs that gave rise to the lowest average cost values are plotted with small circles. These are filled blue, except for the single best location which is filled red. In general the set of these best locations lies on or just below the critical curve for the $g = p_i$ case (shown with a dotted curve).

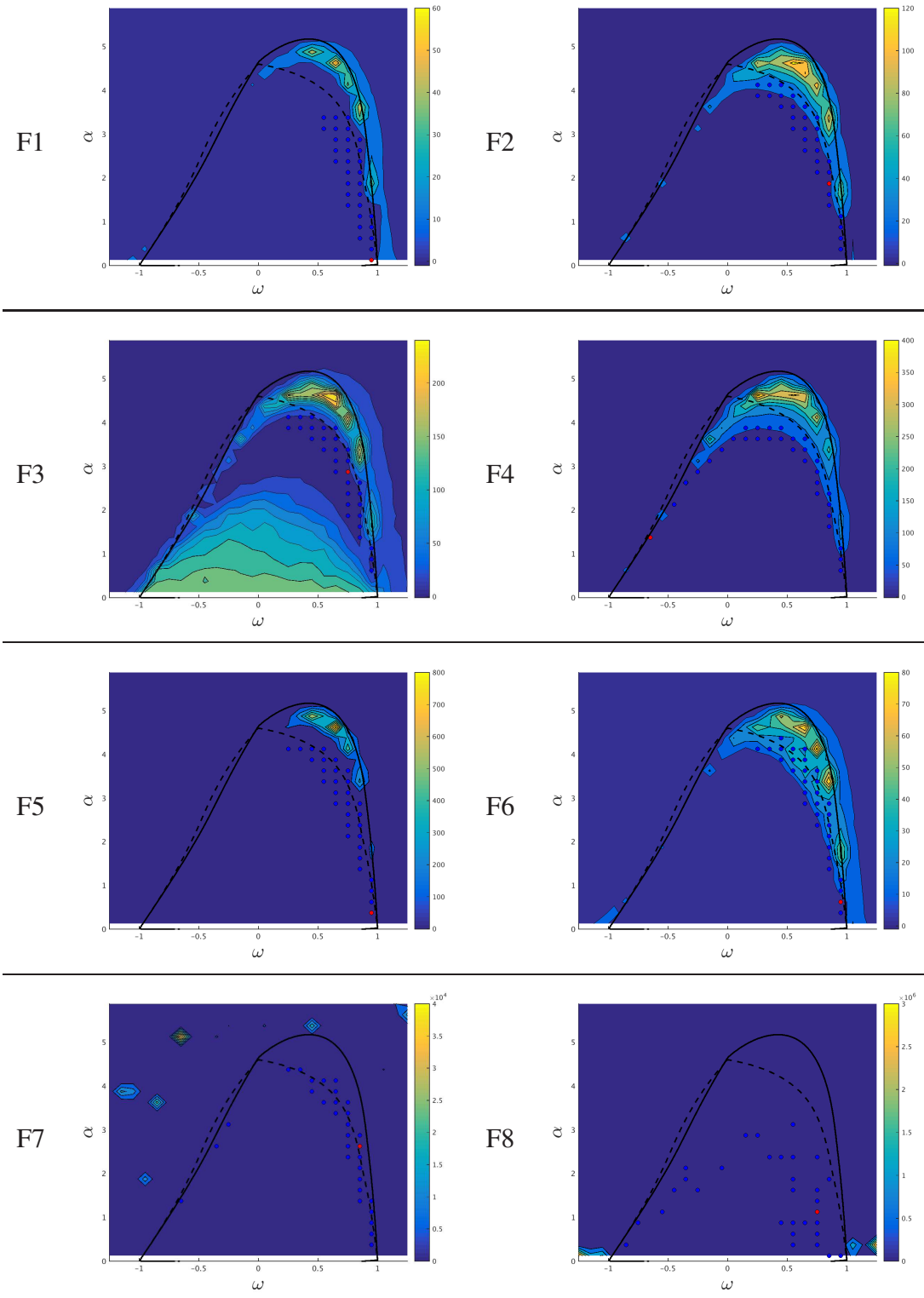


Table B.1: Average | p-g | distance against CEC2013 functions 1 to 8. The average distance between global best location, g, and the centroid of the particles' personal best locations, p_i, is average over all iterations and runs at each $\alpha : \omega$ pair for each function.

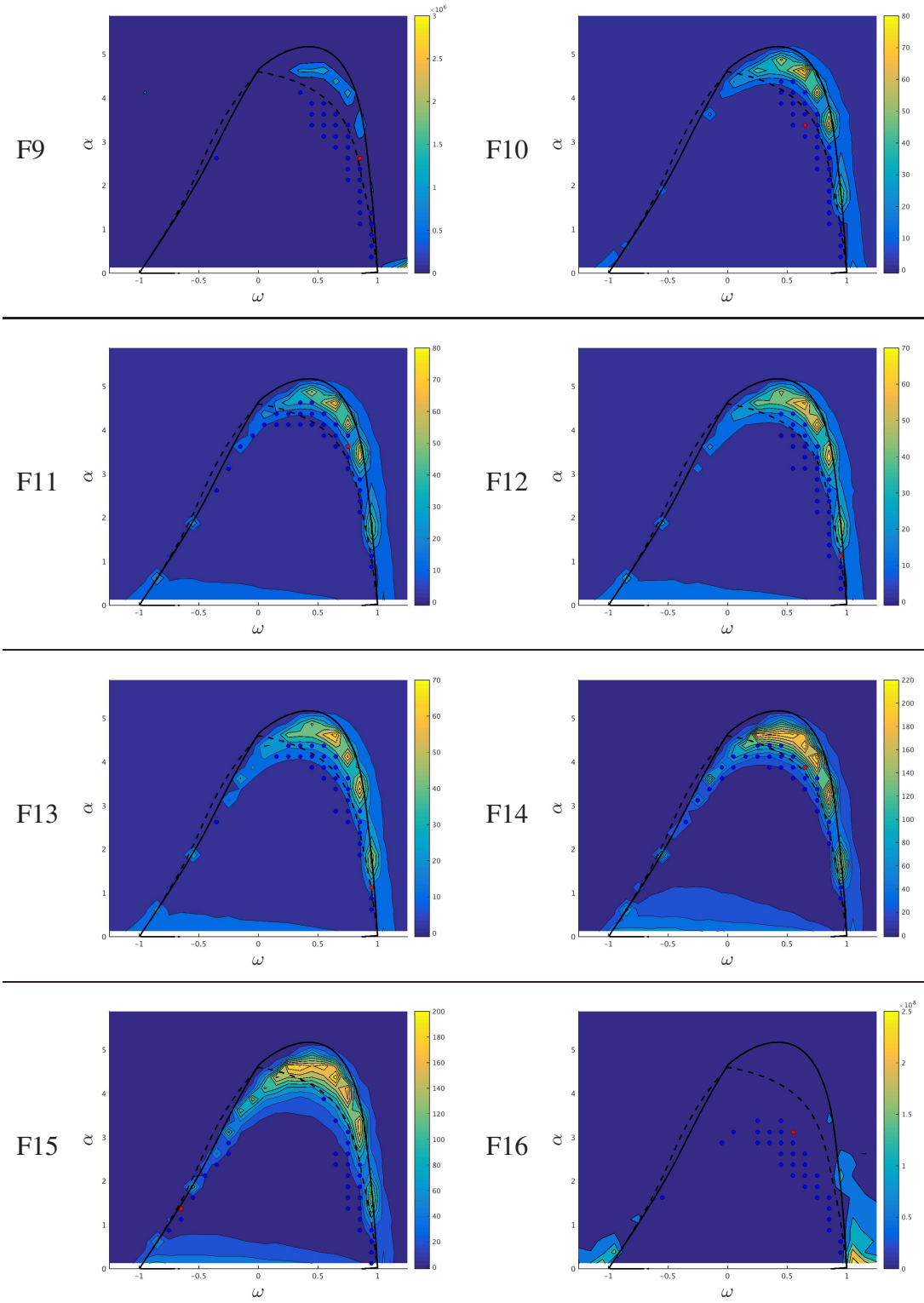


Table B.2: Average | p-g | distance against CEC2013 functions 9 to 16. The average distance between global best location, g, and the centroid of the particles' personal best locations, p_i, is average over all iterations and runs at each $\alpha : \omega$ pair for each function.

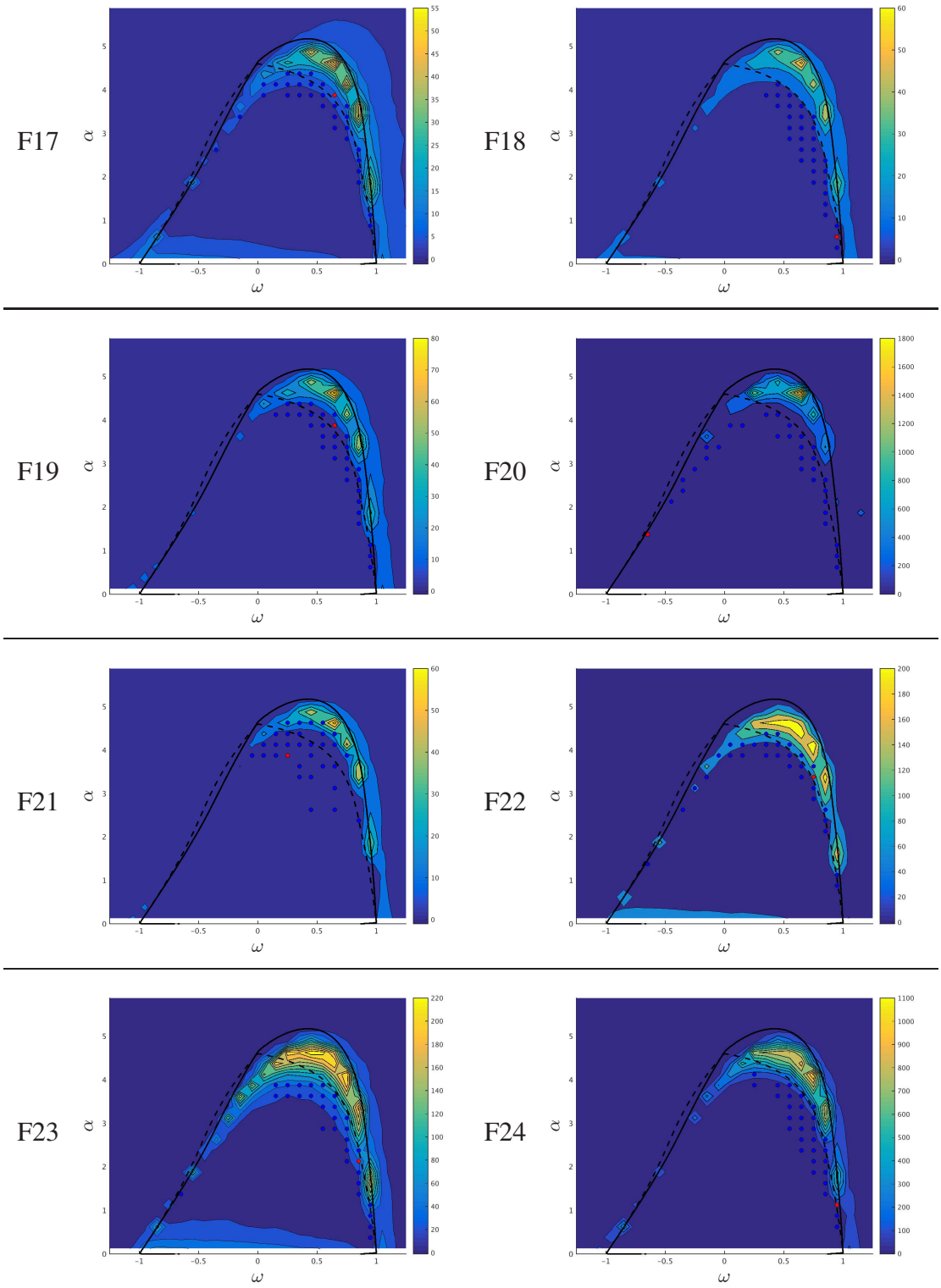


Table B.3: Average | p-g | distance against CEC2013 functions 17 to 24. The average distance between global best location, g, and the centroid of the particles' personal best locations, p_i, is average over all iterations and runs at each $\alpha : \omega$ pair for each function.

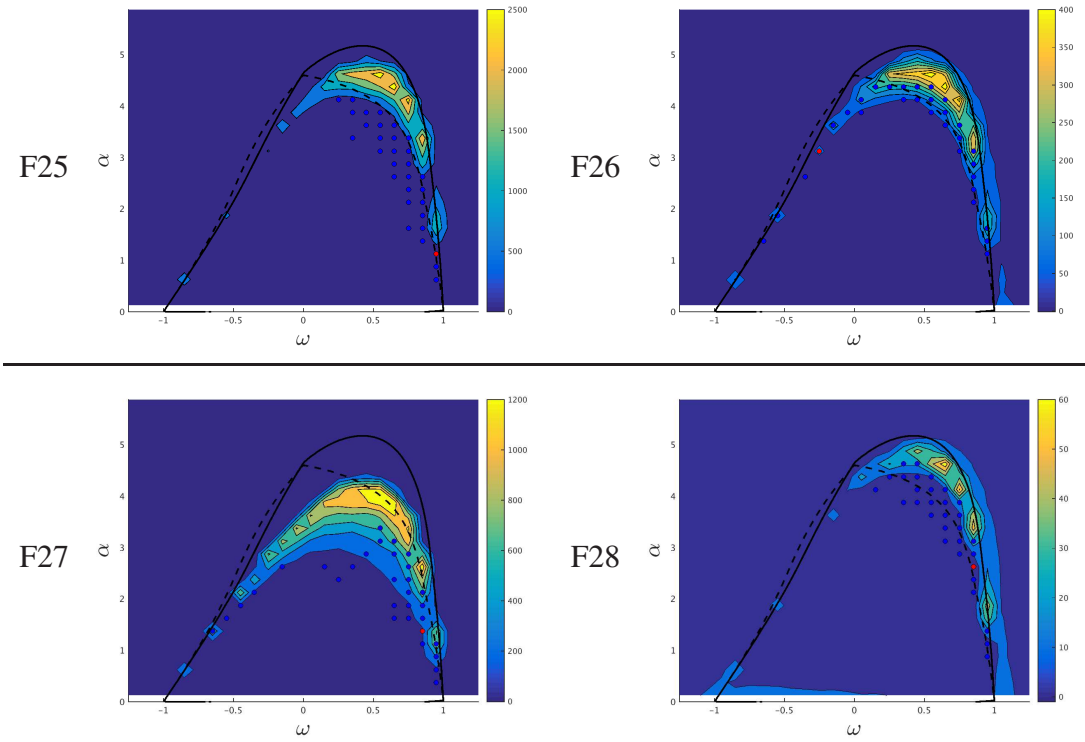


Table B.4: Average | p-g | distance against CEC2013 functions 25 to 28. The average distance between global best location, g, and the centroid of the particles' personal best locations, p_i, is average over all iterations and runs at each $\alpha : \omega$ pair for each function.

Appendix C

CriPS comparison results with selected CEC2013 metaheuristic competitors

The full comparative results been the CriPS algorithm and those competitor algorithms from the CEC2013 competition are presented in the tables below. The CEC2013 protocol was followed and the mean values for each dimensionality:function pair was ranked across all algorithms. The ordering was then determined by consideration the of mean ranking over all problems.

Function	CriPS	NBIPOP-aCMA-ES	NIPOP-aCMA-ES	CDE:b6e6rl	DE-APC	GA	Papa
F1	1.62e-07 (7.0)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)
F2	9.38e+04 (6.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	1.82e-04 (5.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	2.42e+06 (7.0)
F3	5.18e+07 (6.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	3.94e-01 (5.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	2.21e+08 (7.0)
F4	2.24e+03 (6.0)	1.00e-08 (3.0)	1.00e-08 (3.0)	1.00e-08 (3.0)	1.00e-08 (3.0)	1.00e-08 (3.0)	1.34e+04 (7.0)
F5	2.68e-06 (7.0)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)
F6	1.01e+01 (6.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	4.23e+00 (5.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	1.64e+01 (7.0)
F7	2.23e+01 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	2.96e-03 (3.0)	2.05e-02 (4.0)	1.49e+00 (5.0)	4.70e+01 (7.0)
F8	2.03e+01 (1.0)	2.03e+01 (2.0)	2.04e+01 (3.0)	2.04e+01 (5.0)	2.05e+01 (6.0)	2.06e+01 (7.0)	2.04e+01 (4.0)
F9	5.33e+00 (4.0)	2.32e-01 (1.0)	2.54e-01 (2.0)	1.78e+00 (3.0)	9.96e+00 (6.0)	1.05e+01 (7.0)	6.81e+00 (5.0)
F10	9.04e-01 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	3.51e-02 (3.0)	1.06e-01 (4.0)	1.55e-01 (5.0)	1.06e+01 (7.0)
F11	1.74e+00 (5.0)	3.64e-01 (4.0)	2.67e-01 (3.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	1.99e+00 (6.0)	1.71e+01 (7.0)
F12	1.99e+01 (6.0)	2.38e-01 (2.0)	7.80e-02 (1.0)	8.07e+00 (3.0)	1.49e+01 (5.0)	1.19e+01 (4.0)	2.34e+01 (7.0)
F13	2.97e+01 (5.0)	4.84e-01 (2.0)	2.54e-01 (1.0)	9.57e+00 (3.0)	2.74e+01 (4.0)	3.40e+01 (6.0)	4.32e+01 (7.0)
F14	1.46e+02 (6.0)	1.15e+02 (2.0)	1.40e+02 (5.0)	7.10e-02 (1.0)	1.15e+02 (3.0)	1.39e+02 (4.0)	5.24e+02 (7.0)
F15	8.23e+02 (4.0)	1.58e+02 (2.0)	1.29e+02 (1.0)	6.99e+02 (3.0)	1.64e+03 (7.0)	1.46e+03 (6.0)	1.06e+03 (5.0)
F16	7.03e-01 (2.0)	1.20e-01 (1.0)	1.05e+00 (5.0)	9.48e-01 (4.0)	1.35e+00 (6.0)	1.88e+00 (7.0)	7.29e-01 (3.0)
F17	1.49e+01 (6.0)	1.13e+01 (4.0)	1.10e+01 (3.0)	1.01e+01 (1.0)	1.03e+01 (2.0)	1.35e+01 (5.0)	2.77e+01 (7.0)
F18	2.79e+01 (4.0)	1.13e+01 (2.0)	1.08e+01 (1.0)	2.49e+01 (3.0)	4.05e+01 (6.0)	4.81e+01 (7.0)	3.43e+01 (5.0)
F19	6.45e-01 (3.0)	5.25e-01 (2.0)	6.58e-01 (4.0)	3.38e-01 (1.0)	1.23e+00 (6.0)	9.16e-01 (5.0)	1.59e+00 (7.0)
F20	3.10e+00 (4.0)	2.73e+00 (3.0)	2.42e+00 (2.0)	2.31e+00 (1.0)	3.59e+00 (6.0)	4.13e+00 (7.0)	3.59e+00 (5.0)
F21	3.79e+02 (5.0)	1.53e+02 (1.0)	3.51e+02 (4.0)	3.83e+02 (6.0)	3.00e+02 (2.5)	3.00e+02 (2.5)	3.92e+02 (7.0)
F22	2.70e+02 (4.0)	1.75e+02 (3.0)	1.47e+02 (2.0)	1.98e+01 (1.0)	7.70e+02 (7.0)	3.25e+02 (5.0)	6.50e+02 (6.0)
F23	1.08e+03 (4.0)	1.74e+02 (1.0)	1.97e+02 (2.0)	7.05e+02 (3.0)	1.87e+03 (7.0)	1.83e+03 (6.0)	1.27e+03 (5.0)
F24	2.16e+02 (4.0)	1.20e+02 (1.0)	1.49e+02 (2.0)	2.04e+02 (3.0)	2.25e+02 (6.5)	2.25e+02 (6.5)	2.18e+02 (5.0)
F25	2.13e+02 (4.0)	1.77e+02 (1.0)	1.97e+02 (2.0)	2.00e+02 (3.0)	2.25e+02 (6.0)	2.26e+02 (7.0)	2.15e+02 (5.0)
F26	1.98e+02 (5.0)	1.11e+02 (1.0)	1.24e+02 (2.0)	1.55e+02 (3.0)	2.00e+02 (6.5)	2.00e+02 (6.5)	1.71e+02 (4.0)
F27	4.32e+02 (4.0)	3.17e+02 (1.0)	3.51e+02 (3.0)	3.25e+02 (2.0)	5.52e+02 (6.0)	5.60e+02 (7.0)	4.65e+02 (5.0)
F28	3.61e+02 (6.0)	2.49e+02 (1.0)	2.93e+02 (3.0)	2.92e+02 (2.0)	3.00e+02 (4.5)	3.00e+02 (4.5)	4.96e+02 (7.0)

Table C.1: Comparative results between the CriPS algorithm and selection of algorithms in the CEC2013 competition. Objective functions evaluated in 10 dimensions. Values shown are mean fitness over 51 runs and ranking score in parentheses.

Function	CriPS	NBIPOP-aCMA-ES	NIPOP-aCMA-ES	CDE:b6e6rl	DE-APC	GA	Papa
F1	6.52e-06 (7.0)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)
F2	9.26e+05 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	7.00e+04 (3.0)	6.29e+05 (4.0)	7.12e+05 (5.0)	1.34e+07 (7.0)
F3	5.67e+08 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	4.36e+03 (3.0)	8.09e+07 (4.0)	4.24e+08 (5.0)	1.94e+09 (7.0)
F4	7.69e+03 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	1.81e-02 (3.0)	4.13e+00 (4.0)	5.90e+00 (5.0)	4.47e+04 (7.0)
F5	3.66e-05 (7.0)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)
F6	3.31e+01 (5.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	5.25e+00 (3.0)	1.71e+01 (4.0)	6.97e+01 (6.0)	7.77e+01 (7.0)
F7	1.24e+02 (6.0)	2.31e+00 (1.0)	4.05e+00 (2.0)	2.44e+01 (3.0)	8.56e+01 (4.0)	1.32e+02 (7.0)	1.18e+02 (5.0)
F8	2.09e+01 (1.5)	2.09e+01 (5.0)	2.09e+01 (3.0)	2.09e+01 (4.0)	2.11e+01 (6.5)	2.11e+01 (6.5)	2.09e+01 (1.5)
F9	3.07e+01 (4.0)	3.30e+00 (2.0)	2.82e+00 (1.0)	2.86e+01 (3.0)	4.18e+01 (6.0)	4.24e+01 (7.0)	3.31e+01 (5.0)
F10	2.02e-01 (5.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	1.91e-02 (3.0)	1.99e-01 (4.0)	2.12e-01 (6.0)	1.18e+01 (7.0)
F11	2.33e+01 (5.0)	3.04e+00 (3.0)	1.03e+00 (2.0)	1.00e-08 (1.0)	1.89e+01 (4.0)	5.37e+01 (6.0)	1.65e+02 (7.0)
F12	1.74e+02 (6.0)	2.91e+00 (2.0)	6.56e-01 (1.0)	8.36e+01 (5.0)	5.17e+01 (3.0)	6.27e+01 (4.0)	2.15e+02 (7.0)
F13	2.59e+02 (6.0)	2.78e+00 (2.0)	9.31e-01 (1.0)	1.14e+02 (3.0)	1.39e+02 (5.0)	1.29e+02 (4.0)	3.29e+02 (7.0)
F14	1.19e+03 (4.0)	8.10e+02 (3.0)	7.17e+02 (2.0)	2.37e-02 (1.0)	4.51e+03 (7.0)	2.28e+03 (5.0)	2.61e+03 (6.0)
F15	4.57e+03 (4.0)	7.65e+02 (2.0)	6.70e+02 (1.0)	4.66e+03 (5.0)	7.79e+03 (7.0)	5.93e+03 (6.0)	4.39e+03 (3.0)
F16	1.65e+00 (3.0)	4.40e-01 (1.0)	2.48e+00 (5.0)	1.98e+00 (4.0)	3.11e+00 (6.0)	3.64e+00 (7.0)	1.32e+00 (2.0)
F17	7.80e+01 (5.0)	3.44e+01 (3.0)	3.42e+01 (2.0)	3.04e+01 (1.0)	7.12e+01 (4.0)	8.83e+01 (6.0)	2.43e+02 (7.0)
F18	1.79e+02 (6.0)	6.23e+01 (2.0)	5.40e+01 (1.0)	1.72e+02 (5.0)	9.04e+01 (3.0)	1.56e+02 (4.0)	2.57e+02 (7.0)
F19	5.23e+00 (5.0)	2.23e+00 (2.0)	2.41e+00 (3.0)	1.84e+00 (1.0)	3.96e+00 (4.0)	1.10e+01 (6.0)	2.41e+01 (7.0)
F20	1.43e+01 (5.5)	1.29e+01 (2.0)	1.34e+01 (3.0)	1.18e+01 (1.0)	1.41e+01 (4.0)	1.45e+01 (7.0)	1.43e+01 (5.5)
F21	3.16e+02 (4.0)	1.92e+02 (1.0)	2.41e+02 (2.0)	2.97e+02 (3.0)	4.43e+02 (6.5)	4.43e+02 (6.5)	3.30e+02 (5.0)
F22	9.77e+02 (4.0)	8.38e+02 (3.0)	5.73e+02 (2.0)	1.23e+02 (1.0)	5.46e+03 (7.0)	2.10e+03 (5.0)	3.25e+03 (6.0)
F23	5.62e+03 (6.0)	6.67e+02 (1.0)	6.67e+02 (2.0)	5.01e+03 (4.0)	8.00e+03 (7.0)	5.59e+03 (5.0)	5.00e+03 (3.0)
F24	2.96e+02 (4.0)	1.62e+02 (1.0)	2.91e+02 (3.0)	2.52e+02 (2.0)	3.08e+02 (7.0)	3.04e+02 (6.0)	2.97e+02 (5.0)
F25	3.20e+02 (6.0)	2.20e+02 (1.0)	2.79e+02 (3.0)	2.75e+02 (2.0)	3.04e+02 (4.0)	3.08e+02 (5.0)	3.27e+02 (7.0)
F26	3.48e+02 (5.0)	1.58e+02 (1.0)	2.51e+02 (4.0)	2.10e+02 (2.0)	4.04e+02 (7.0)	3.81e+02 (6.0)	2.46e+02 (3.0)
F27	1.15e+03 (4.5)	4.69e+02 (1.0)	8.70e+02 (2.0)	1.01e+03 (3.0)	1.34e+03 (6.0)	1.36e+03 (7.0)	1.15e+03 (4.5)
F28	1.00e+03 (6.0)	2.69e+02 (1.0)	3.00e+02 (3.5)	3.00e+02 (3.5)	3.00e+02 (3.5)	3.00e+02 (3.5)	2.08e+03 (7.0)

Table C.2: Comparative results between the CriPS algorithm and selection of algorithms in the CEC2013 competition. Objective functions evaluated in 30 dimensions. Values shown are mean fitness over 51 runs and ranking score in parentheses.

Function	CriPS	NBIPOP-aCMA-ES	NIPOP-aCMA-ES	CDE:b6e6rl	DE-APC	GA	Papa
F1	3.25e-05 (7.0)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)	1.00e-08 (3.5)
F2	2.08e+06 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	3.23e+05 (3.0)	8.27e+05 (4.0)	1.05e+06 (5.0)	1.59e+07 (7.0)
F3	3.40e+09 (6.0)	1.82e+01 (1.0)	1.88e+01 (2.0)	8.61e+06 (3.0)	6.39e+07 (4.0)	8.44e+08 (5.0)	5.06e+09 (7.0)
F4	1.24e+04 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	2.32e-01 (3.0)	5.97e+00 (4.0)	3.45e+01 (5.0)	5.43e+04 (7.0)
F5	1.19e-04 (5.0)	1.00e-08 (2.5)	1.00e-08 (2.5)	1.00e-08 (2.5)	7.67e+05 (6.0)	8.15e+05 (7.0)	1.00e-08 (2.5)
F6	7.01e+01 (5.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	4.34e+01 (3.0)	4.38e+01 (4.0)	9.71e+01 (7.0)	9.70e+01 (6.0)
F7	6.87e+01 (3.0)	4.97e+00 (1.0)	1.17e+01 (2.0)	8.26e+01 (5.0)	7.21e+01 (4.0)	1.04e+02 (6.0)	1.28e+02 (7.0)
F8	2.11e+01 (1.5)	2.11e+01 (4.0)	2.11e+01 (3.0)	2.11e+01 (5.0)	2.12e+01 (6.5)	2.12e+01 (6.5)	2.11e+01 (1.5)
F9	3.03e+01 (3.0)	7.22e+00 (2.0)	6.88e+00 (1.0)	5.67e+01 (4.0)	7.50e+01 (6.0)	7.77e+01 (7.0)	6.15e+01 (5.0)
F10	1.29e+00 (6.0)	1.00e-08 (1.5)	1.00e-08 (1.5)	3.54e-02 (3.0)	2.22e-01 (4.0)	4.39e-01 (5.0)	2.99e+01 (7.0)
F11	6.86e+01 (4.0)	5.50e+00 (3.0)	1.99e+00 (2.0)	1.00e-08 (1.0)	7.32e+01 (5.0)	1.05e+02 (6.0)	3.53e+02 (7.0)
F12	4.38e+02 (6.0)	5.37e+00 (2.0)	1.37e+00 (1.0)	1.89e+02 (5.0)	9.95e+01 (3.0)	1.59e+02 (4.0)	4.41e+02 (7.0)
F13	5.80e+02 (6.0)	7.59e+00 (2.0)	1.48e+00 (1.0)	2.53e+02 (4.0)	2.35e+02 (3.0)	3.63e+02 (5.0)	6.38e+02 (7.0)
F14	2.09e+03 (4.0)	1.38e+03 (3.0)	1.26e+03 (2.0)	3.65e-02 (1.0)	1.08e+04 (7.0)	6.98e+03 (6.0)	5.06e+03 (5.0)
F15	8.93e+03 (4.0)	1.55e+03 (2.0)	1.35e+03 (1.0)	9.27e+03 (5.0)	1.52e+04 (6.0)	1.64e+04 (7.0)	8.51e+03 (3.0)
F16	2.35e+00 (4.0)	8.78e-01 (1.0)	3.37e+00 (5.0)	2.30e+00 (3.0)	3.85e+00 (6.0)	4.34e+00 (7.0)	2.07e+00 (2.0)
F17	1.73e+02 (5.0)	5.74e+01 (2.0)	5.77e+01 (3.0)	5.08e+01 (1.0)	2.03e+02 (6.0)	1.67e+02 (4.0)	6.01e+02 (7.0)
F18	4.70e+02 (6.0)	1.34e+02 (1.0)	1.94e+02 (3.0)	3.28e+02 (4.0)	1.41e+02 (2.0)	4.35e+02 (5.0)	6.33e+02 (7.0)
F19	1.39e+01 (5.0)	4.46e+00 (2.0)	4.47e+00 (3.0)	3.43e+00 (1.0)	1.19e+01 (4.0)	1.49e+01 (6.0)	1.28e+02 (7.0)
F20	2.40e+01 (6.0)	2.25e+01 (2.0)	2.30e+01 (3.0)	2.15e+01 (1.0)	2.39e+01 (5.0)	2.43e+01 (7.0)	2.37e+01 (4.0)
F21	8.45e+02 (5.0)	1.98e+02 (1.0)	3.65e+02 (2.0)	4.60e+02 (3.0)	1.13e+03 (6.5)	1.13e+03 (6.5)	7.58e+02 (4.0)
F22	2.50e+03 (4.0)	1.45e+03 (3.0)	1.02e+03 (2.0)	3.60e+01 (1.0)	1.17e+04 (7.0)	9.47e+03 (6.0)	6.50e+03 (5.0)
F23	1.16e+04 (5.0)	1.71e+03 (2.0)	1.19e+03 (1.0)	9.78e+03 (3.0)	1.54e+04 (6.0)	1.63e+04 (7.0)	1.02e+04 (4.0)
F24	3.91e+02 (5.0)	2.40e+02 (1.0)	3.70e+02 (3.0)	3.33e+02 (2.0)	3.94e+02 (6.0)	3.96e+02 (7.0)	3.83e+02 (4.0)
F25	4.61e+02 (7.0)	2.48e+02 (1.0)	3.65e+02 (3.0)	3.64e+02 (2.0)	3.91e+02 (4.0)	3.96e+02 (5.0)	4.44e+02 (6.0)
F26	4.28e+02 (5.0)	1.96e+02 (1.0)	2.88e+02 (2.0)	3.28e+02 (3.0)	4.91e+02 (6.0)	4.96e+02 (7.0)	4.26e+02 (4.0)
F27	2.05e+03 (5.0)	7.28e+02 (1.0)	1.90e+03 (3.0)	1.73e+03 (2.0)	2.22e+03 (6.0)	2.27e+03 (7.0)	2.03e+03 (4.0)
F28	2.03e+03 (4.0)	4.00e+02 (1.5)	5.72e+02 (3.0)	4.00e+02 (1.5)	3.40e+03 (5.0)	3.42e+03 (6.0)	4.02e+03 (7.0)

Table C.3: Comparative results between the CriPS algorithm and selection of algorithms in the CEC2013 competition. Objective functions evaluated in 50 dimensions. Values shown are mean fitness over 51 runs and ranking score in parentheses.

Condition	CriPS	NBIPOP-aCMA-ES	NIPOP-aCMA-ES	CDE:b6e6rl	DE-APC	GA	Papa
All data	4.96	1.95 (1)	2.35 (5)	2.88 (10)	4.86 (15)	5.51 (17)	5.50 (21)
10D data	4.86	2.05 (1)	2.55 (5)	2.98 (10)	4.64 (15)	5.13 (17)	5.79 (21)
30D data	5.09	1.95 (1)	2.25 (5)	2.84 (10)	4.95 (15)	5.48 (17)	5.45 (21)
50D data	4.95	1.84 (1)	2.25 (5)	2.80 (10)	4.98 (15)	5.91 (17)	5.27 (21)
F8-F28 only	4.63	1.87 (1)	2.37 (5)	2.67 (10)	5.24 (15)	5.82 (17)	5.40 (21)

Table C.4: Average ranking results for the CriPS algorithm and a selection of algorithms in the CEC2013 competition. Across all function:dimension pairings the mean results for each algorithm are ranked (1 through 7). These are averaged and the results shown in this table. The figures in parentheses are the positions occupied by the algorithms in the CEC2013 competition. The first line shows the results for all data. The ranking of the CriPS algorithm can be seen to be between the 15^{th} and 17^{th} best algorithms in the competition by this criteria. Lines two, three and four show the results when we consider the problem functions only in specific dimensionalities: CriPS performance appears similar in each case. Last line shows the same exercise but only considering a subset of functions. It should not be surprising that excluding functions that an algorithm does poorly on improves its apparent performance, but by excluding the 'simpler' functions we perhaps show that the CriPS strategy may be worth more in more complex problem spaces.

Appendix D

Detailed results from CriPS variations

D.1 CriPS algorithm variations and suitable statistical testing

Here we present some details relating to the performance of variations of the CriPS algorithm.

The algorithm is explored whilst varying the swarm population and with differing values for the ε parameter. In each case we generate a number of different sets of results. Each time we make a new comparison we run the algorithm 51 times, for each of the 28 functions. We execute only for the 10 dimensional problem space to limit (to some extent) the number of cases to be explored. As we are comparing multiple results sets, paired t-tests (or variants such as Welch's t-test) are not appropriate. ANOVA tests allow us to compare multiple results, but have certain assumptions that seem unlikely to be met e.g. normally distributed results. To explore this we can plot, for a single function, the fitness values achieved at the end of each run. As these values are often spread over several orders of magnitude we can also display them with the log of the fitness used. We can also plot the fitnesses in histogram form. Figure D.1 displays this data for the CEC2013 function 1 (sphere function). Fitness values achieved by the algorithm over the 51 repeat executions achieve results that vary over many orders of magnitude. The algorithm is capable on occasion of beating the competition's 'zero' target (fitness results less that 10^{-8} are considered to be zero). However, there are a few poor results that result in the linear results histogram to appear highly skewed. The logarithmic histogram arguably appears to have a more 'normal' appearance, so it may be that we could transform the data prior to applying an ANOVA

test. This picture, however, is not the same for all functions explored. Figure D.2 shows the equivalent data for CEC2013's composite function 8. With this function neither histogram appears normal. For this reason comparisons between multiple variants of our algorithm will use a non-parametric statistical test, the Kruskal-Wallis test.

D.1.1 The Kruskal-Wallis test

This can be thought of as a non parametric version of the classical one-way ANOVA test. It is a rank sum test for more than two groups comparing the medians of the groups to determine if the samples may be considered as having been drawn from the same population. Instead of ANOVA's assumption of normal distributions in the group populations the Kruskal-Wallace test assumes that the distributions are the same (but not necessarily normal). For our testing of CriPS variants where either population or ε are being varied the distributions appear to be similar but, as noted in section D.1, not always normal.

D.1.2 Results of CriPS with varying swarm population size

Using the CEC2013 protocol we reran the CriPS algorithm for 10 dimensional problems with different swarm populations. Populations of 10, 25, 50, 75, 100, 125, and 250 were used. The performances are shown in Figure D.3. Details of the mean and standard deviation values for all the functions and all the swarm sizes are in Table D.4. There are performance differences between the swarm size variants. We can use the Kruskal-Wallis test (see section D.1.1) to determine if there is any significance to these differences.

Figure D.4 shows some selected results of applying the Kruskal-Wallis test to the results for several functions. For consistency we always select the first group (swarm population = 10, shown in blue). Groups that are significantly different are highlighted in red. Groups that are not significantly different are displayed in black. One can visually assess other comparisons: significant differences result in the lines not overlapping. The best results are always the lines with the leftmost midpoints. Function 1 does better with swarms of 25 or 50, whereas for function 10 the largest swarm of 250 wins, but there seem to be no other functions that favour the largest swarm sizes (although several where it doesn't seem detrimental). Function 12 shows larger swarms performing best. A smaller swarm has necessarily less information about good locations to share, but it would appear broadly that there is no benefit of

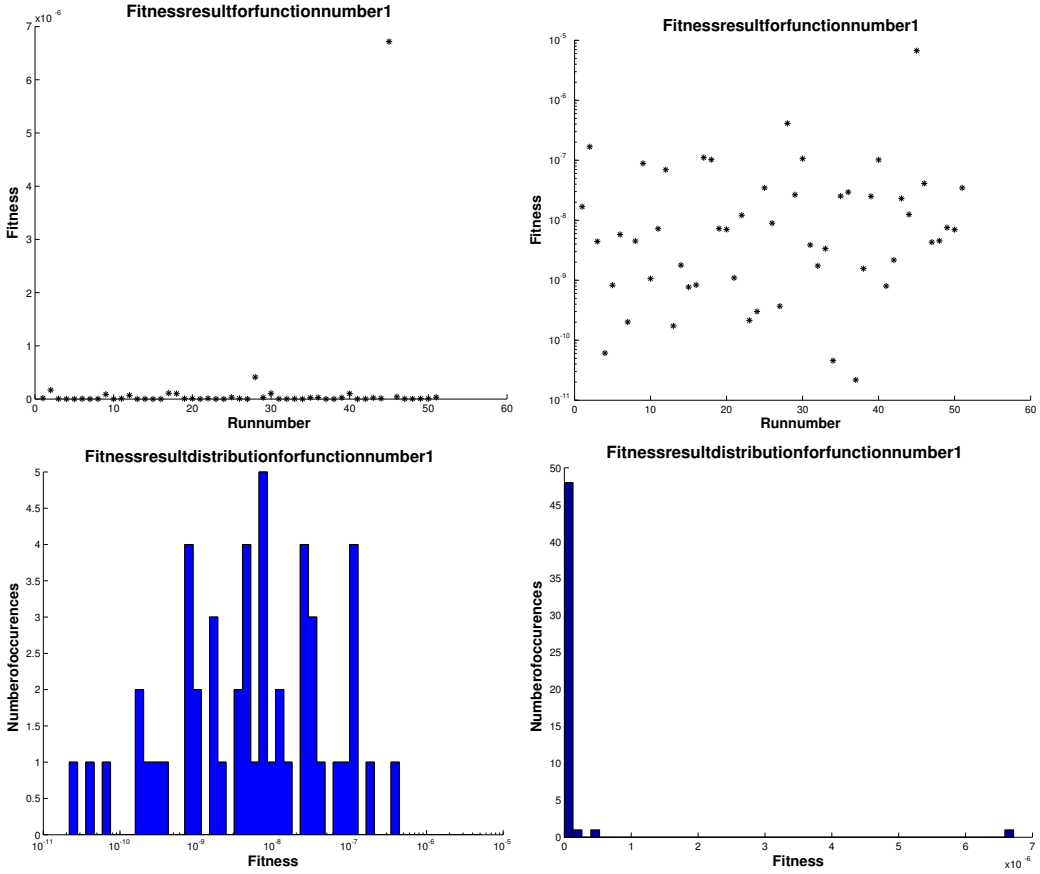


Figure D.1: Examples of final fitness distribution data achieved by the CriPS algorithm against CEC2013 function 1 (Sphere). From top left, clockwise: Individual final fitness values on linear scale; Individual final fitness values on log scale; histogram of final fitness values on linear scale; histogram of final fitness values on log scale.

increasing the swarm size above a modest 25 particles or so. We note that all these results are for 10 dimensional search spaces. Higher dimensionalities may present a differing picture.

D.1.3 Results of CriPS with varying ε value

We have seen that ε can be used to tune the statistical balance of exploratory and exploitative moves in the swarm. Initially we were concerned to use this to determine whether criticality in the swarm size changes could be engineered. If, instead, we wish solely to explore whether we can improve the results obtained we can use ε to modify the swarm dynamic. We run the CEC2013 tests on the ten dimensional objective functions, setting $\varepsilon \in \{0.025, 0.075, 0.125, 0.5, 0.75, 1.0, 1.5, 5.0\}$. Figure D.5 shows the performance of the algorithm for these epsilon values for 10 dimensional problems in the CE2013 challenge. The full mean and standard deviations are shown in

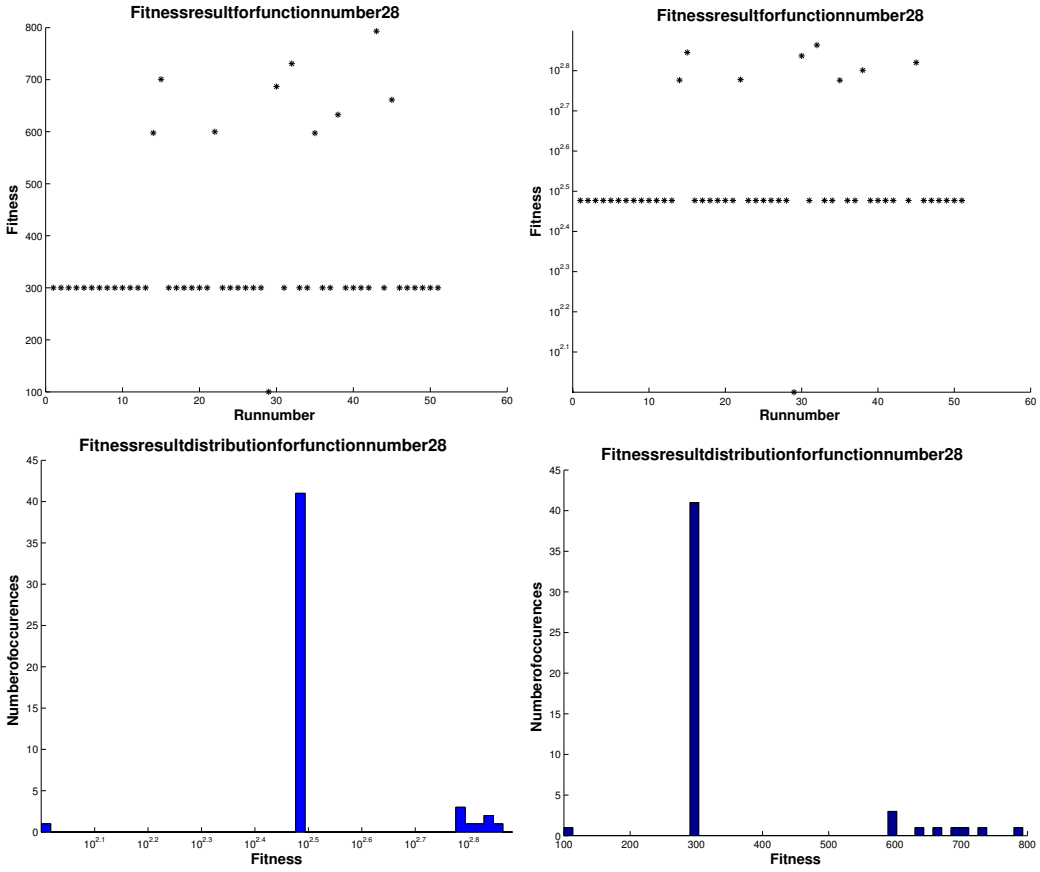


Figure D.2: Examples of final fitness distribution data achieved by the CriPS algorithm against CEC2013 function 28 (Composite function 8). From top left, clockwise: Individual final fitness values on linear scale; Individual final fitness values on log scale; histogram of final fitness values on linear scale; histogram of final fitness values on log scale.

Table D.5.

We can explore whether any of these differences are significant. Once again we use the Kruskal-Wallis test to make these comparisons. Figure D.6 shows exemplar plots of the results of Kruskal-Wallis tests run on the results for several of the functions. Looking at functions 1, 7 and 28 we see different ε values lead to better results. All the tests use p=0.05 as the significance level. These plots visualise the results of the KW tests. Each group mean is represented by a symbol, and the interval is represented by a line extending out from the symbol. Two group means are significantly different if their intervals are disjoint; they are not significantly different if their intervals overlap. No overriding picture emerges from the 28 functions. Smaller ε values often result in better results, but equally often all values generate similar results and occasionally (function 7) the larger ε appears to be best.

Function	Best	Worst	Median	Mean	Std
1	2.18e-11	6.72e-06	6.97e-09	1.62e-07	9.39e-07
2	1.14e+04	2.95e+05	8.02e+04	9.38e+04	7.47e+04
3	2.54e+01	1.27e+09	2.67e+06	5.18e+07	2.01e+08
4	1.31e+02	1.87e+04	1.21e+03	2.24e+03	3.41e+03
5	7.82e-09	2.57e-05	6.17e-07	2.68e-06	4.75e-06
6	2.18e-02	8.15e+01	9.81e+00	1.01e+01	1.68e+01
7	9.79e-01	8.32e+01	1.61e+01	2.23e+01	2.10e+01
8	2.02e+01	2.05e+01	2.03e+01	2.03e+01	8.12e-02
9	1.67e+00	9.08e+00	5.67e+00	5.33e+00	1.75e+00
10	8.87e-02	2.52e+00	7.67e-01	9.04e-01	5.77e-01
11	0.00e+00	9.95e+00	9.95e-01	1.74e+00	2.16e+00
12	5.97e+00	4.88e+01	1.79e+01	1.99e+01	1.10e+01
13	1.04e+01	5.38e+01	2.95e+01	2.97e+01	9.84e+00
14	3.42e+00	5.22e+02	1.33e+02	1.46e+02	1.20e+02
15	1.95e+02	1.63e+03	8.15e+02	8.23e+02	3.22e+02
16	1.59e-01	1.34e+00	6.48e-01	7.03e-01	2.86e-01
17	3.36e+00	2.76e+01	1.44e+01	1.49e+01	3.87e+00
18	1.38e+01	4.63e+01	2.70e+01	2.79e+01	8.56e+00
19	3.96e-02	1.47e+00	5.85e-01	6.45e-01	2.90e-01
20	2.19e+00	3.93e+00	3.13e+00	3.10e+00	3.91e-01
21	1.00e+02	4.00e+02	4.00e+02	3.79e+02	7.02e+01
22	3.26e+01	6.03e+02	2.54e+02	2.70e+02	1.44e+02
23	6.12e+01	1.70e+03	1.13e+03	1.08e+03	3.38e+02
24	2.01e+02	2.26e+02	2.16e+02	2.16e+02	5.89e+00
25	1.14e+02	2.25e+02	2.16e+02	2.13e+02	1.55e+01
26	1.07e+02	3.28e+02	2.00e+02	1.99e+02	7.30e+01
27	3.04e+02	6.27e+02	4.38e+02	4.32e+02	1.04e+02
28	1.00e+02	7.93e+02	3.00e+02	3.61e+02	1.48e+02

Table D.1: CriPS results under the CEC2013 test protocol. Problem space is 10 dimensional.

The value of CriPS is in avoiding stagnation, the algorithm still uses the standard PSO mechanisms to guide the particles towards a solution. It would therefore appear that so long as the mechanism of using changes in the swarm metric to update the PSO parameters is employed broadly comparable solutions are available to the algorithm.

Function	Best	Worst	Median	Mean	Std
1	1.88e-07	8.71e-05	1.85e-06	6.52e-06	1.40e-05
2	3.41e+05	2.19e+06	8.48e+05	9.26e+05	4.57e+05
3	9.24e+06	3.14e+09	3.17e+08	5.67e+08	6.48e+08
4	6.28e+02	3.13e+04	7.21e+03	7.69e+03	5.44e+03
5	1.15e-06	3.30e-04	1.73e-05	3.66e-05	5.71e-05
6	7.30e-01	8.03e+01	1.61e+01	3.31e+01	2.78e+01
7	1.85e+02	1.85e+02	1.85e+02	1.85e+02	0.00e+00
8	2.08e+01	2.10e+01	2.09e+01	2.09e+01	6.73e-02
9	3.43e+01	3.43e+01	3.43e+01	3.43e+01	0.00e+00
10	3.62e-02	1.18e+00	1.50e-01	2.03e-01	2.14e-01
11	1.99e+00	7.88e+01	1.60e+01	2.33e+01	1.84e+01
12	8.76e+01	2.80e+02	1.60e+02	1.74e+02	5.25e+01
13	1.63e+02	3.75e+02	2.57e+02	2.59e+02	4.91e+01
14	2.01e+02	3.94e+03	1.01e+03	1.19e+03	6.60e+02
15	3.08e+03	6.32e+03	4.54e+03	4.57e+03	6.31e+02
16	7.91e-01	2.67e+00	1.66e+00	1.65e+00	4.54e-01
17	4.62e+01	1.68e+02	7.38e+01	7.80e+01	2.32e+01
18	1.15e+02	3.17e+02	1.67e+02	1.79e+02	5.12e+01
19	2.42e+00	1.46e+01	4.64e+00	5.23e+00	2.26e+00
20	1.14e+01	1.50e+01	1.45e+01	1.43e+01	7.03e-01
21	1.00e+02	4.44e+02	3.00e+02	3.17e+02	1.04e+02
22	3.89e+02	2.07e+03	8.79e+02	9.77e+02	4.31e+02
23	3.27e+03	7.61e+03	5.61e+03	5.62e+03	1.01e+03
24	2.57e+02	3.22e+02	2.97e+02	2.96e+02	1.39e+01
25	2.81e+02	3.45e+02	3.20e+02	3.20e+02	1.42e+01
26	2.00e+02	3.97e+02	3.73e+02	3.48e+02	6.51e+01
27	8.88e+02	1.36e+03	1.15e+03	1.15e+03	1.03e+02
28	1.00e+02	3.59e+03	3.00e+02	1.00e+03	1.05e+03

Table D.2: CriPS results under the CEC2013 test protocol. Problem space is 30 dimensional.

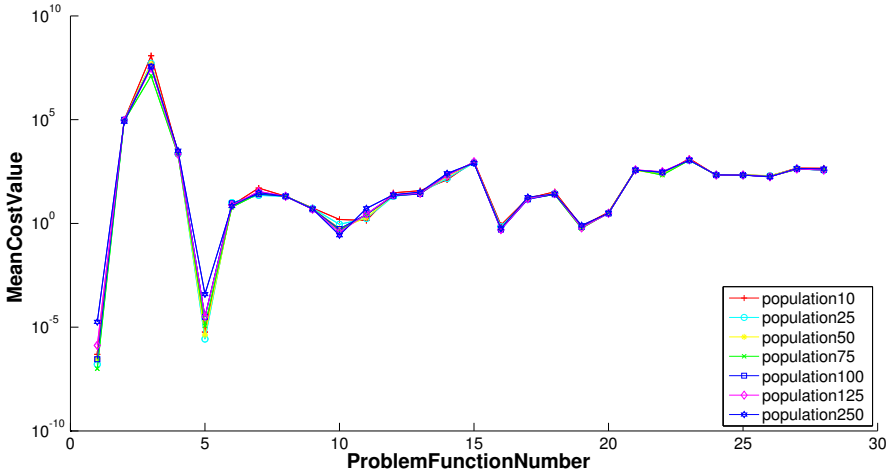


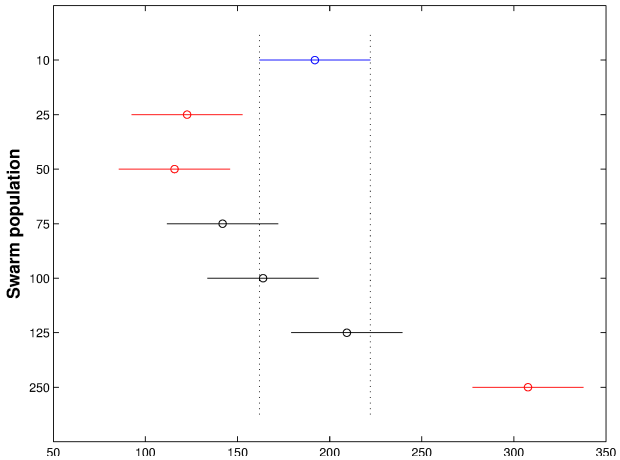
Figure D.3: CriPS performance with varying swarm populations. The CEC2013 competition protocol was used. Objective functions are 10 dimensional only. The figure shows the mean fitness achieved for each swarm size against each objective function.

Function	Best	Worst	Median	Mean	Std
1	1.66e-06	4.12e-04	1.59e-05	3.25e-05	6.05e-05
2	1.01e+06	4.21e+06	2.05e+06	2.08e+06	6.00e+05
3	1.48e+08	1.23e+10	2.34e+09	3.40e+09	2.88e+09
4	2.16e+03	2.79e+04	1.21e+04	1.24e+04	6.43e+03
5	4.23e-06	5.65e-04	7.64e-05	1.19e-04	1.10e-04
6	2.12e+01	2.26e+02	4.87e+01	7.01e+01	3.92e+01
7	2.12e-09	2.36e+02	2.88e-05	6.87e+01	8.26e+01
8	2.10e+01	2.18e+01	2.11e+01	2.11e+01	4.43e-02
9	5.27e-10	7.20e+01	9.96e-09	3.03e+01	3.26e+01
10	2.06e-01	4.50e+00	1.21e+00	1.29e+00	6.55e-01
11	8.01e+00	1.88e+02	5.22e+01	6.86e+01	4.62e+01
12	2.09e+02	7.14e+02	4.34e+02	4.38e+02	9.84e+01
13	3.78e+02	8.69e+02	5.77e+02	5.81e+02	1.14e+02
14	7.55e+02	5.64e+03	1.93e+03	2.10e+03	1.02e+03
15	6.86e+03	1.16e+04	8.97e+03	8.93e+03	1.12e+03
16	9.97e-01	3.70e+00	2.31e+00	2.35e+00	7.08e-01
17	1.05e+02	3.52e+02	1.53e+02	1.73e+02	5.68e+01
18	2.34e+02	9.25e+02	4.24e+02	4.70e+02	1.62e+02
19	4.71e+00	3.19e+01	1.31e+01	1.39e+01	5.66e+00
20	2.16e+01	2.45e+01	2.45e+01	2.40e+01	7.53e-01
21	2.00e+02	1.12e+03	1.12e+03	8.45e+02	3.63e+02
22	4.84e+02	5.14e+03	2.38e+03	2.50e+03	1.09e+03
23	7.93e+03	1.45e+04	1.20e+04	1.16e+04	1.55e+03
24	3.43e+02	4.40e+02	3.93e+02	3.91e+02	1.94e+01
25	4.29e+02	5.08e+02	4.59e+02	4.61e+02	1.93e+01
26	2.00e+02	4.90e+02	4.61e+02	4.28e+02	9.26e+01
27	1.66e+03	2.48e+03	2.04e+03	2.05e+03	1.70e+02
28	4.00e+02	6.77e+03	4.00e+02	2.03e+03	2.18e+03

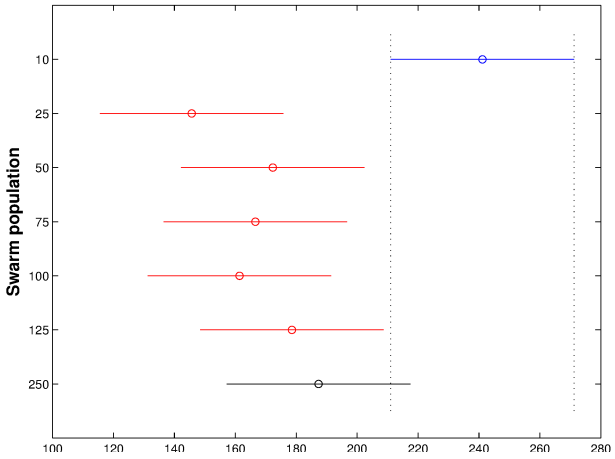
Table D.3: CriPS results under the CEC2013 test protocol. Problem space is 50 dimensional.

Function	size=10	size=25	size=50	size=75	size=100	size=125	size=250
1	4.91e-07 (2.40e-06)	1.62e-07 (9.39e-07)	3.00e-07 (1.57e-06)	1.00e-07 (2.73e-07)	2.84e-07 (6.21e-07)	1.32e-06 (4.25e-06)	1.79e-05 (3.71e-05)
2	9.60e+04 (7.97e+04)	9.38e+04 (7.47e+04)	9.51e+04 (7.53e+04)	8.76e+04 (7.21e+04)	9.79e+04 (9.04e+04)	9.04e+04 (8.40e+04)	8.07e+04 (5.55e+04)
3	1.20e+08 (4.15e+08)	5.18e+07 (2.01e+08)	4.93e+07 (1.52e+08)	1.31e+07 (3.40e+07)	3.59e+07 (1.05e+08)	3.05e+07 (6.56e+07)	3.57e+07 (8.38e+07)
4	2.58e+03 (2.97e+03)	2.24e+03 (3.41e+03)	2.25e+03 (2.02e+03)	2.02e+03 (2.34e+03)	2.63e+03 (2.51e+03)	2.20e+03 (2.08e+03)	3.14e+03 (2.17e+03)
5	5.81e-06 (1.35e-05)	2.68e-06 (4.75e-06)	4.17e-06 (1.28e-05)	1.16e-05 (3.26e-05)	3.13e-05 (1.29e-04)	3.20e-05 (5.33e-05)	3.92e-04 (6.57e-04)
6	7.55e+00 (1.15e+01)	1.01e+01 (1.68e+01)	7.42e+00 (1.11e+01)	6.02e+00 (4.78e+00)	9.32e+00 (1.43e+01)	7.50e+00 (8.23e+00)	6.88e+00 (1.18e+01)
7	5.06e+01 (3.50e+01)	2.25e+01 (2.10e+01)	2.71e+01 (2.89e+01)	2.70e+01 (2.23e+01)	2.56e+01 (1.62e+01)	3.48e+01 (2.44e+01)	3.03e+01 (1.93e+01)
8	2.03e+01 (7.16e-02)	2.03e+01 (8.12e-02)	2.03e+01 (7.54e-02)	2.03e+01 (8.14e-02)	2.03e+01 (6.24e-02)	2.03e+01 (6.74e-02)	2.03e+01 (8.61e-02)
9	5.53e+00 (1.61e+00)	5.33e+00 (1.75e+00)	5.03e+00 (1.49e+00)	5.26e+00 (1.63e+00)	4.83e+00 (1.70e+00)	4.85e+00 (1.46e+00)	5.02e+00 (1.53e+00)
10	1.55e+00 (1.22e+00)	9.05e-01 (5.77e-01)	5.70e-01 (4.08e-01)	4.59e-01 (3.47e-01)	5.40e-01 (3.88e-01)	3.80e-01 (2.35e-01)	2.73e-01 (1.86e-01)
11	1.38e+00 (3.70e+00)	1.74e+00 (2.16e+00)	1.89e+00 (1.37e+00)	2.61e+00 (2.38e+00)	2.69e+00 (2.17e+00)	2.80e+00 (2.54e+00)	5.23e+00 (3.73e+00)
12	2.98e+01 (1.17e+01)	1.99e+01 (1.10e+01)	2.25e+01 (1.04e+01)	2.17e+01 (1.01e+01)	2.14e+01 (9.97e+00)	2.31e+01 (1.13e+01)	2.40e+01 (1.12e+01)
13	3.79e+01 (1.38e+01)	2.97e+01 (9.84e+00)	2.75e+01 (9.95e+00)	2.79e+01 (1.07e+01)	2.63e+01 (1.18e+01)	2.85e+01 (1.21e+01)	3.20e+01 (1.25e+01)
14	1.26e+02 (1.23e+02)	1.46e+02 (1.20e+02)	1.83e+02 (1.59e+02)	2.11e+02 (1.19e+02)	2.23e+02 (1.20e+02)	1.99e+02 (1.65e+02)	2.55e+02 (1.41e+02)
15	9.73e+02 (2.82e+02)	8.23e+02 (3.22e+02)	8.63e+02 (2.89e+02)	8.70e+02 (2.96e+02)	8.28e+02 (2.85e+02)	9.21e+02 (3.08e+02)	8.16e+02 (2.40e+02)
16	8.34e-01 (3.16e-01)	7.03e-01 (2.86e-01)	5.73e-01 (2.56e-01)	5.56e-01 (2.68e-01)	5.12e-01 (2.05e-01)	5.04e-01 (1.90e-01)	5.75e-01 (2.33e-01)
17	1.63e+01 (6.03e+00)	1.49e+01 (3.87e+00)	1.59e+01 (3.69e+00)	1.60e+01 (4.44e+00)	1.47e+01 (3.36e+00)	1.55e+01 (3.65e+00)	1.83e+01 (5.06e+00)
18	3.34e+01 (9.89e+00)	2.79e+01 (8.56e+00)	2.87e+01 (7.89e+00)	2.46e+01 (7.33e+00)	2.54e+01 (8.34e+00)	2.74e+01 (7.82e+00)	2.64e+01 (7.09e+00)
19	7.21e-01 (5.74e-01)	6.45e-01 (2.90e-01)	6.19e-01 (2.06e-01)	5.78e-01 (1.84e-01)	6.70e-01 (2.42e-01)	6.18e-01 (1.76e-01)	7.99e-01 (4.00e-01)
20	3.47e+00 (4.94e-01)	3.10e+00 (3.91e-01)	3.15e+00 (4.62e-01)	3.22e+00 (4.53e-01)	3.05e+00 (4.90e-01)	3.07e+00 (6.19e-01)	3.12e+00 (4.58e-01)
21	3.83e+02 (5.56e+01)	3.79e+02 (7.02e+01)	3.59e+02 (8.77e+01)	3.88e+02 (4.32e+01)	3.73e+02 (6.96e+01)	3.81e+02 (6.01e+01)	3.71e+02 (6.73e+01)
22	2.42e+02 (1.93e+02)	2.70e+02 (1.44e+02)	2.34e+02 (1.66e+02)	2.16e+02 (1.73e+02)	2.91e+02 (1.63e+02)	3.11e+02 (1.99e+02)	2.97e+02 (1.88e+02)
23	1.33e+03 (4.05e+02)	1.08e+03 (3.38e+02)	1.14e+03 (3.42e+02)	1.11e+03 (3.11e+02)	1.13e+03 (3.25e+02)	1.17e+03 (3.54e+02)	1.13e+03 (4.44e+02)
24	2.11e+02 (2.78e+01)	2.16e+02 (5.89e+00)	2.15e+02 (1.33e+01)	2.14e+02 (1.50e+01)	2.13e+02 (1.10e+01)	2.15e+02 (5.40e+00)	2.18e+02 (5.44e+00)
25	2.19e+02 (6.96e+00)	2.13e+02 (1.55e+01)	2.13e+02 (1.35e+01)	2.12e+02 (1.42e+01)	2.12e+02 (1.23e+01)	2.14e+02 (6.13e+00)	2.13e+02 (1.31e+01)
26	1.93e+02 (7.46e+01)	1.99e+02 (7.30e+01)	2.02e+02 (7.16e+01)	1.85e+02 (7.19e+01)	1.86e+02 (6.19e+01)	1.78e+02 (4.57e+01)	1.74e+02 (3.79e+01)
27	4.62e+02 (1.02e+02)	4.32e+02 (1.04e+02)	4.23e+02 (1.08e+02)	3.99e+02 (1.07e+02)	4.14e+02 (1.00e+02)	4.21e+02 (9.87e+01)	4.48e+02 (9.81e+01)
28	4.54e+02 (2.28e+02)	3.61e+02 (1.48e+02)	3.78e+02 (1.70e+02)	3.93e+02 (2.06e+02)	3.78e+02 (1.66e+02)	3.73e+02 (1.67e+02)	4.29e+02 (2.14e+02)

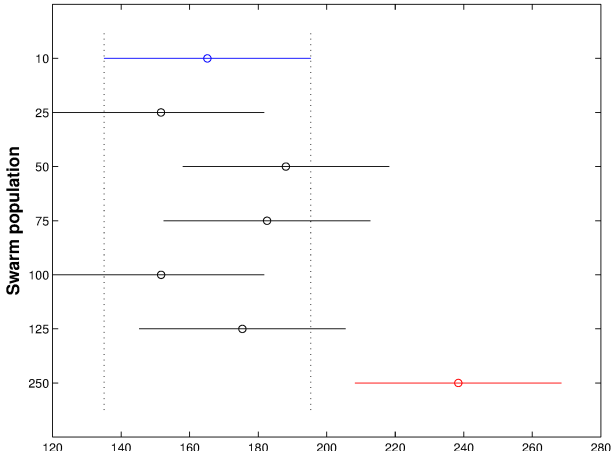
Table D.4: Mean and standard deviation (in parentheses) of CriPS running with a range of swarm sizes against the CEC2013 competition functions. Objective functions evaluated in 10 dimensions.



(a) Function 1



(c) Function 12



(c) Function 17

Figure D.4: CriPS performance with varying swarm population. The CEC2013 competition protocol was used. Objective functions are 10 dimensional only. The best population values are those whose data lines lie leftmost of each graph. All comparisons are made with the smallest population case (shown in blue). Black lines indicate results that overlap with this and are not significant. Red lines show results that appear to be significant. Mean fitness value achieved is shown on the horizontal axis.

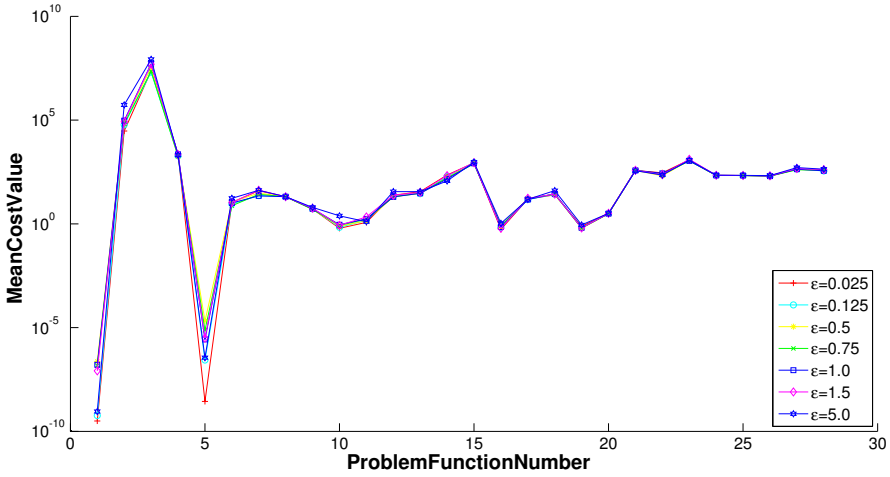
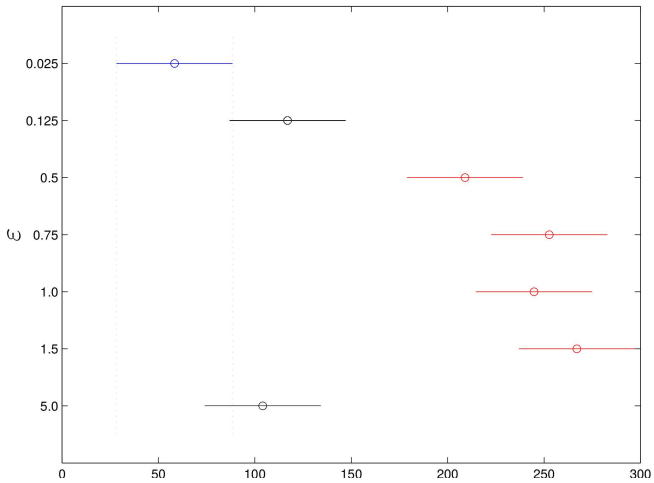


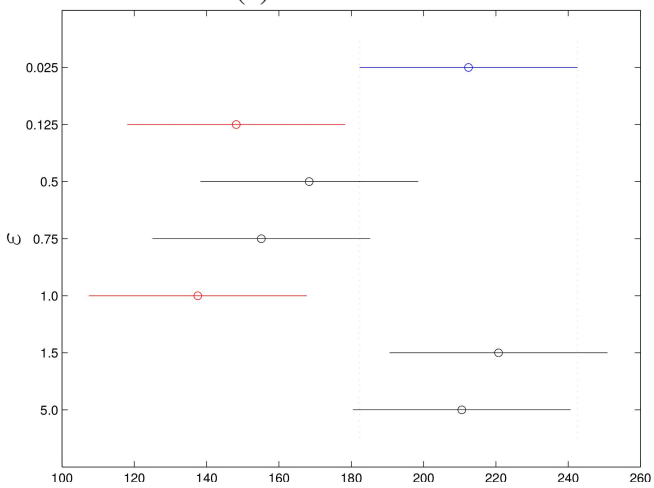
Figure D.5: CriPS performance with varying ε values. The CEC2013 competition protocol was used. Objective functions are 10 dimensional only. The figure shows the mean fitness achieved for each ε against each objective function.

Function	ε =0.025	ε =0.125	ε =0.5	ε =0.75	ε =1.0	ε =1.5	ε =5.0
1	3.15e-10 (1.38e-09)	5.77e-10 (1.14e-09)	2.33e-07 (1.57e-06)	1.47e-07 (4.13e-07)	1.62e-07 (9.39e-07)	8.20e-08 (1.52e-07)	9.01e-10 (4.53e-09)
2	2.97e+04 (3.54e+04)	5.96e+04 (5.63e+04)	8.29e+04 (9.76e+04)	6.60e+04 (4.53e+04)	9.38e+04 (7.48e+04)	8.46e+04 (8.35e+04)	5.40e+05 (3.23e+05)
3	2.35e+07 (7.78e+07)	2.05e+07 (4.50e+07)	3.35e+07 (7.37e+07)	2.24e+07 (8.83e+07)	5.18e+07 (2.00e+08)	4.88e+07 (1.58e+08)	8.39e+07 (2.14e+08)
4	2.38e+03 (2.44e+03)	1.90e+03 (1.92e+03)	1.77e+03 (1.77e+03)	1.84e+03 (2.35e+03)	2.24e+03 (3.41e+03)	2.15e+03 (2.43e+03)	2.06e+03 (1.58e+03)
5	2.76e-09 (8.40e-09)	2.75e-07 (1.15e-06)	2.06e-05 (1.08e-04)	7.47e-06 (2.80e-05)	2.68e-06 (4.75e-06)	3.16e-06 (5.76e-06)	3.55e-07 (8.88e-07)
6	1.08e+01 (1.65e+01)	9.15e+00 (1.49e+01)	1.15e+01 (1.89e+01)	7.45e+00 (1.16e+01)	1.01e+01 (1.68e+01)	1.07e+01 (1.72e+01)	1.70e+01 (2.47e+01)
7	3.97e+01 (2.83e+01)	2.29e+01 (1.66e+01)	3.01e+01 (2.82e+01)	2.63e+01 (2.31e+01)	2.23e+01 (2.10e+01)	4.16e+01 (2.81e+01)	4.16e+01 (3.31e+01)
8	2.03e+01 (8.72e-02)	2.03e+01 (8.51e-02)	2.03e+01 (6.45e-02)	2.03e+01 (7.15e-02)	2.03e+01 (8.12e-02)	2.03e+01 (6.75e-02)	2.03e+01 (8.40e-02)
9	5.22e+00 (1.61e+00)	5.31e+00 (1.52e+00)	5.03e+00 (1.51e+00)	4.96e+00 (1.58e+00)	5.33e+00 (1.75e+00)	5.67e+00 (1.60e+00)	6.20e+00 (1.44e+00)
10	6.10e-01 (4.14e-01)	6.86e-01 (4.73e-01)	7.70e-01 (4.56e-01)	8.38e-01 (5.61e-01)	9.05e-01 (5.77e-01)	8.51e-01 (6.77e-01)	2.42e+00 (1.59e+00)
11	1.18e+00 (2.05e+00)	1.49e+00 (1.94e+00)	1.41e+00 (1.54e+00)	1.71e+00 (2.05e+00)	1.74e+00 (2.16e+00)	2.15e+00 (3.07e+00)	1.27e+00 (1.67e+00)
12	2.38e+01 (1.09e+01)	2.14e+01 (9.43e+00)	2.02e+01 (8.94e+00)	2.01e+01 (7.46e+00)	1.99e+01 (1.10e+01)	2.38e+01 (1.22e+01)	3.57e+01 (1.63e+01)
13	3.58e+01 (1.34e+01)	2.86e+01 (1.18e+01)	2.95e+01 (1.01e+01)	3.09e+01 (1.04e+01)	2.97e+01 (9.84e+00)	3.27e+01 (1.17e+01)	3.56e+01 (1.07e+01)
14	2.26e+02 (1.41e+02)	1.87e+02 (1.49e+02)	1.65e+02 (1.35e+02)	1.64e+02 (1.28e+02)	1.46e+02 (1.20e+02)	2.11e+02 (1.63e+02)	1.16e+02 (1.19e+02)
15	8.62e+02 (2.78e+02)	8.92e+02 (2.62e+02)	9.20e+02 (2.93e+02)	7.76e+02 (2.51e+02)	8.23e+02 (3.22e+02)	8.84e+02 (2.75e+02)	9.35e+02 (3.21e+02)
16	5.44e-01 (2.21e-01)	6.84e-01 (2.93e-01)	7.49e-01 (2.88e-01)	8.01e-01 (2.20e-01)	7.03e-01 (2.86e-01)	6.14e-01 (2.65e-01)	1.06e+00 (2.25e-01)
17	1.57e+01 (4.07e+00)	1.49e+01 (3.46e+00)	1.69e+01 (4.08e+00)	1.61e+01 (4.28e+00)	1.49e+01 (3.87e+00)	1.71e+01 (4.42e+00)	1.52e+01 (4.79e+00)
18	2.50e+01 (8.05e+00)	2.63e+01 (8.63e+00)	2.63e+01 (8.29e+00)	2.73e+01 (7.82e+00)	2.79e+01 (8.56e+00)	2.72e+01 (8.23e+00)	3.98e+01 (9.18e+00)
19	7.60e-01 (3.11e-01)	6.61e-01 (2.83e-01)	6.13e-01 (2.27e-01)	6.26e-01 (2.51e-01)	6.45e-01 (2.90e-01)	6.34e-01 (3.56e-01)	8.89e-01 (5.56e-01)
20	3.27e+00 (4.90e-01)	3.13e+00 (5.50e-01)	3.02e+00 (6.49e-01)	3.11e+00 (6.16e-01)	3.10e+00 (3.91e-01)	3.21e+00 (5.23e-01)	3.20e+00 (4.69e-01)
21	3.75e+02 (7.45e+01)	3.71e+02 (7.83e+01)	3.75e+02 (7.17e+01)	3.83e+02 (6.55e+01)	3.79e+02 (7.02e+01)	3.81e+02 (6.94e+01)	3.65e+02 (7.96e+01)
22	2.88e+02 (2.69e+02)	2.31e+02 (1.62e+02)	2.07e+02 (1.48e+02)	2.39e+02 (1.71e+02)	2.70e+02 (1.44e+02)	2.33e+02 (1.57e+02)	2.25e+02 (1.47e+02)
23	1.21e+03 (3.42e+02)	1.16e+03 (3.48e+02)	1.14e+03 (3.55e+02)	1.08e+03 (3.12e+02)	1.08e+03 (3.38e+02)	1.29e+03 (3.12e+02)	1.18e+03 (3.66e+02)
24	2.17e+02 (5.42e+00)	2.15e+02 (5.58e+00)	2.15e+02 (5.67e+00)	2.16e+02 (5.69e+00)	2.16e+02 (5.89e+00)	2.19e+02 (4.14e+00)	2.20e+02 (4.57e+00)
25	2.15e+02 (1.54e+01)	2.14e+02 (1.48e+01)	2.12e+02 (1.55e+01)	2.13e+02 (1.29e+01)	2.13e+02 (1.55e+01)	2.14e+02 (1.91e+01)	2.17e+02 (1.50e+01)
26	2.08e+02 (7.83e+01)	2.02e+02 (7.86e+01)	1.94e+02 (7.09e+01)	2.03e+02 (8.15e+01)	1.99e+02 (7.30e+01)	2.07e+02 (7.99e+01)	2.10e+02 (7.96e+01)
27	4.34e+02 (9.03e+01)	4.36e+02 (1.01e+02)	4.19e+02 (9.93e+01)	4.03e+02 (8.97e+01)	4.32e+02 (1.04e+02)	4.62e+02 (9.85e+01)	5.12e+02 (8.79e+01)
28	3.76e+02 (1.62e+02)	3.55e+02 (1.45e+02)	3.84e+02 (1.66e+02)	3.89e+02 (1.67e+02)	3.61e+02 (1.48e+02)	4.09e+02 (1.94e+02)	4.29e+02 (2.15e+02)

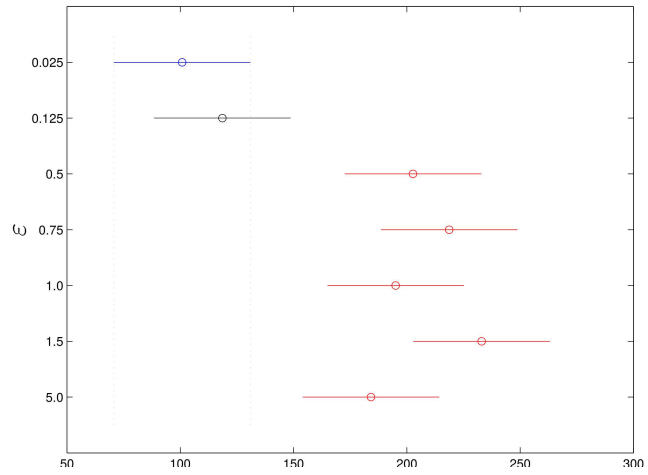
Table D.5: Mean and standard deviation (in parentheses) of CriPS running with a range of values of ε against the CEC2013 competition functions. Objective functions evaluated in 10 dimensions.



(a) Function 1



(b) Function 7



(c) Function 28

Figure D.6: CriPS performance with varying ε values. The CEC2013 competition protocol was used. Objective functions are 10 dimensional only. The best ε values are those whose data lines lie leftmost of each graph. All comparisons are made with the smallest ε case (shown in blue). Black lines indicate results that overlap with this and are not significant. Red lines show results that appear to be significant. Mean fitness value achieved is shown on the horizontal axis.

Appendix E

Lyapunov exponent estimation method

E.1 Method

Vanneste (2010) presents a method for estimating generalised Lyapunov exponents for products of random matrices. It is a Monte Carlo algorithm that performs importance sampling using a simple random resampling step. The method outlined was proposed as brute force Monte Carlo methods are inefficient at calculating large Lyapunov exponents. As we are looking for Lyapunov exponent $\lambda = 0$ this is not the case here. We opt to use this approach though as we may have wished to explore regions of fast convergence or divergence and we could do easily without changing algorithm.

Algorithm 2 reproduces the algorithm from Vanneste's paper. The variable λ_t is the Lyapunov exponent we would like to find. It is used to tune the algorithm to that value. PSO parameters to be used are also supplied as input paramters. Additionally we provide the number of iterations (N) and matrix realisations (K) to use. We used 1000 realisations and 2500 iterations to generate the data. For each iteration K unit vectors are updated by drawing and applying different dynamic PSO matrices. The application of these matrices rotates the unit vectors and moves them outward or inward. The size of this expansion or contraction is recorded in our distances array, D(K). These distances are further scaled by raising them to a power equal to our target Lyapunov exponents. For the next iteration we select from the new set of unit vectors at random but weighed to favour larger size changes.

We ran the algorithm over the range of PSO paramter values shown in the earlier stability plots with steps in ω of of 0.05 and steps in α of 0.1. For each parameter pair

the algorithm was run and generated an estimated Lyapunov exponent. This generated a grid of estimated λ values against α and ω. In general these will not be exactly zero so we need to estimate a contour through the grid data that marks the likely location of zero values. Matlab's contour method was used to draw the zero contour .

Algorithm 2: Vanneste's Lyapunov exponent estimation algorithm based on (Vanneste, 2010).

Input: Target Lyapunov exponent λ_t, Maximum iterations N, Maximum realisations K, PSO parameters ω, α_1 and α_2
Output: Returns calculated Lyapunov exponent λ.

```
for k = 1 to K do
    E(k) = (1,0) ;                                   // Unit vectors in R^2
end
for n = 1 to N do
    for k = 1 to K do
        Draw random matrix M(k) ;                    // PSO update matrix
        E'(k) = M(k)E(k) / ||M(k)E(k)|| ;            // Move and re-project
        D(k) = ||M(k)E(k)||^λt ;                     // store weighted moves
    end
    for k = 1 to K do
        γ(k) = Σ_{l=1}^{k} D(l) ;                    // partial sums of weighted moves
    end
    β(n) = γ(K) ;                                    // sum of all weighted moves
    for k = 1 to K do
        Draw ε uniformly in [0, β(n)] ;              // Resampling loop
        j = min_{l∈(1,...,K)} (γ(l) − ε ≥ 0) ;       // first bigger partial sum
        E(k) = E'(j) ;                               // Select this new unit vector
    end
end
λ = (Σ_{n=1}^{N} log β_n) / (N − log K) ;            // Calculate estimated value
```

Bibliography

A. Ajith, G. Crina, R. Vitorino, G. Crina, and A. Ajith. Stigmergic optimization: Inspiration, technologies and perspectives. In *Stigmergic Optimization*, volume 31 of *Studies in Computational Intelligence*, pages 1–24. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/978-3-540-34690-6_1. 10.1007/978-3-540-34690-6_1.

A. Attanasi, A. Cavagna, L. Del Castello, I. Giardina, S. Melillo, L. Parisi, O. Pohl, B. Rossaro, E. Shen, E. Silvestri, et al. Finite-size scaling as a way to probe near-criticality in natural swarms. *Physical review letters*, 113(23):238102, 2014.

P. Bak. *How nature works*. Oxford University Press, Oxford, 1997.

P. Bak and K. Sneppen. Punctuated equilibrium and criticality in a simple model of evolution. *Physical review letters*, 71(24):4083–4086, 1993.

P. Bak, C. Tang, K. Wiesenfeld, et al. Self-organized criticality. *Physical review A*, 38 (1):364–374, 1988.

P. Ball. *Nature's patterns, A Tapestry in Three Parts: Volume 2 Flow*. Oxford University Press, 2011a.

P. Ball. *Nature's patterns, A Tapestry in Three Parts: Volume 1 Shapes*. Oxford University Press, 2011b.

J. Bansal, P. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham. Inertia weight strategies in particle swarm optimization. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 633–640. IEEE, 2011.

M. Batty. Spatial entropy. *Geographical Analysis*, 6(1):1–31, 1974.

L. Bayindir and E. Sahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering and Computer Sciences*, 15(2):115–147, 2007. ISSN 13000632.

M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green, T. Ikegami, K. Kaneko, and T. S. Ray. Open problems in artificial life. *Artificial life*, 6(4):363–376, 2000.

J. M. Beggs and D. Plenz. Neuronal avalanches in neocortical circuits. *The Journal of neuroscience*, 23(35):11167–11177, 2003.

J. M. Belmonte, G. L. Thomas, L. G. Brunnet, R. M. de Almeida, and H. Chaté. Self-propelled particle model for cell-sorting phenomena. *Physical review letters*, 100(24):248702, 2008.

M. Bodi, R. Thenius, T. Schmickl, and K. Crailsheim. Robustness of two interacting robot swarms using the beeclust algorithm. In *MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, 2009.

E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: From natural to artificial systems*. Oxford University Press, 1999.

V. Braitenberg. *Vehicles: experiments in synthetic psychology*. Bradford Books. MIT Press, 1986. ISBN 9780262521123. URL `http://books.google.com/books?id=7KkUAT_q_sQC`.

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

S. Camazine. The regulation of pollen foraging by honey bees: How foragers assess the colony's need for pollen. *Behavioral Ecology and Sociobiology*, 32(4):pp. 265–272, 1993. ISSN 03405443. URL `http://www.jstor.org/stable/4600814`.

S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraula. *Self-Organization in Biological Systems*. Princeton University Press, 2001. ISBN 0691012113.

A. Cavagna, A. Cimarelli, I. Giardina, G. Parisi, R. Santagati, F. Stefanini, and M. Viale. Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107(26):11865–11870, 2010.

CEC2013. URL http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm.

H. Chaté and M. Muñoz. Insect swarms go critical. *Physics*, 7:120, 2014.

L. Chittka and J. Niven. Are bigger brains better? *Current Biology*, 19(21): R995–R1008, 2009.

S. Chowdhury, W. Tong, A. Messac, and J. Zhang. A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation. *Structural and Multidisciplinary Optimization*, pages 1–22, 2013.

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

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

M. Clerc. Confinements and biases in particle swarm optimisation. Technical Report hal- 00122799, Open archive HAL, 2006. URL http://hal.archives-ouvertes.fr/.

M. Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

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

C. G. Cordero. Parameter adaptation and criticality in particle swarm optimization. Master's thesis, The School of Informatics, The University of Edinburgh, 2012.

K. Crailsheim. Trophallactic interactions in the adult honeybee. *Apis mellifera*, pages 97–112, 1998.

N. Darke. The wrecking season, Boatshed films., 2004.

A. de Andrade Costa, M. Copelli, and O. Kinouchi. Can dynamical synapses produce true self-organized criticality? *Journal of Statistical Mechanics: Theory and Experiment*, 2015(6):P06004, 2015.

A. Deluca, N. R. Moloney, and Á. Corral. Data-driven prediction of thresholded time series of rainfall and self-organized criticality models. *Physical Review E*, 91(5): 052808, 2015.

J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, 1991.

M. Eigen and P. Schuster. The hypercycle. *Naturwissenschaften*, 65(1):7–41, 1978.

S. M. Elsayed, R. A. Sarker, and D. L. Essam. A genetic algorithm for solving the cec'2013 competition problems on real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 356–360. IEEE, 2013a.

S. M. Elsayed, R. A. Sarker, and T. Ray. Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1932–1937. IEEE, 2013b.

A. P. Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, pages 191–202. Springer, 2010.

A. Erskine. A honeybee inspired multimodal heterogeneous artificial colony. Master's thesis, The School of Informatics, The University of Edinburgh, 2011.

A. Erskine and J. M. Herrmann. Cell division behaviour in a heterogeneous swarm environment. In *ECAL 2013 conference*. ECAL, 2013.

A. Erskine and J. M. Herrmann. Crips: Critical particle swarm optimisation. In *Proceedings of the European Conference on Artificial Life 2015*, pages 207–214. ECAL, 2015a.

A. Erskine and J. M. Herrmann. Cell-division behavior in a heterogeneous swarm environment. *Artificial Life*, pages 481–500, 2015b. doi: http://dx.doi.org/10.1162/ARTL_a_00188.

C. W. Eurich, J. M. Herrmann, and U. A. Ernst. Finite-size effects of avalanche dynamics. *Physical review E*, 66(6):066137, 2002.

T. Feder. Statistical physics is for the birds. *Physics Today*, 60(10):28–30, 2007. doi: 10.1063/1.2800090. URL `http://link.aip.org/link/?PTO/60/28/1`.

H. Fellermann, S. Rasmussen, H.-J. Ziock, and R. V. Solé. Life cycle of a minimal protocell-a dissipative particle dynamics study. *Artificial Life*, 13(4):319–345, 2007.

C. M. Fernandes, J. J. Merelo, and A. C. Rosa. Controlling the parameters of the particle swarm optimization with a self-organized criticality model. In *Parallel Problem Solving from Nature-PPSN XII*, pages 153–163. Springer, 2012.

K. Frisch. *The dancing bees: an account of the life and senses of the honey bee*. Methuen & Co. Ltd, 1954.

H. Furstenberg and H. Kesten. Products of random matrices. *Annals of Mathematical Statistics*, 31(2):457–469, 1960.

A. García-Villoria and R. Pastor. Introducing dynamic diversity into a discrete particle swarm optimization. *Computers & Operations Research*, 36(3):951–966, 2009.

F. W. Grasso, T. R. Consi, D. C. Mountain, and J. Atema. Biomimetic robot lobster performs chemo-orientation in turbulence using a pair of spatially separated sensors: Progress and challenges. *Robotics and Autonomous Systems*, 30(1):115–131, 2000.

P. Grodzicki and M. Caputa. Social versus individual behaviour: a comparative approach to thermal behaviour of the honeybee (apis mellifera l.) and the american cockroach (periplaneta americana l.). *Journal of insect physiology*, 51(3):315–322, 2005.

C. A. Hale, H. Meinhardt, and P. A. de Boer. Dynamic localization cycle of the cell division regulator mine in escherichia coli. *The EMBO journal*, 20(7):1563–1572, 2001.

J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tache, I. Said, V. Durier, S. Canonge, J. Ame, et al. Social integration of robots into groups of cockroaches to control self-organized choices. *Science Signalling*, 318(5853):1155, 2007.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

N. Hill and S. Bullock. Modelling the role of trail pheromone in the collective construction of termite royal chambers. In P. Andrews, L. Caves, R. Doursat,

S. Hickinbotham, F. Polack, and S. Stepney, editors, *Advances in Artificial Life, ECAL 2015*, pages 43–50. MIT Press, April 2015.

O. Holland and C. Melhuish. Stigmergy, self-organization, and sorting in collective robotics. *Artif. Life*, 5:173–202, Apr. 1999. ISSN 1064-5462. doi: 10.1162/106454699568737. URL `http://portal.acm.org/citation.cfm?id=338945.338956`.

B. Holmes. Alive! The race to create life from scratch. *New Scientist*, 12, 2005.

G. A. Jagers op Akkerhuis. Analysing hierarchy in the organization of biological and physical systems. *Biological reviews*, 83(1):1–12, 2008.

S. Janson and M. Middendorf. On trajectories of particles in PSO. In *Swarm Intelligence Symposium (SIS 2007)*, pages 150–155. IEEE, 2007.

H. J. Jensen. *Self-organized criticality: emergent complex behavior in physical and biological systems*, volume 10. Cambridge university press, 1998.

M. Jiang, Y. Luo, and S. Yang. Stagnation analysis in particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 92–99. IEEE, 2007.

J. C. Jones, M. R. Myerscough, S. Graham, and B. P. Oldroyd. Honey bee nest thermoregulation: diversity promotes stability. *Science*, 305(5682):402–404, 2004.

D. Karaboga and B. Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1-4):61–85, 2009.

S. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, USA, 1996.

D. Kengyel, T. Schmickl, H. Hamann, R. Thenius, and K. Crailsheim. Embodiment of honeybee's thermotaxis in a mobile robot swarm. In *Proceedings of the 10th European conference on Advances in artificial life: Darwin meets von Neumann-Volume Part II*, pages 69–76. Springer-Verlag, 2009.

J. Kennedy. The behavior of particles. In V. Porto, N. Saravanan, D.Waagen, and A. E. Eiben, editors, *Evolutionary programming VII*, pages 579–589. Springer, 1998.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl. Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17 (3):237–259, 2009.

S. Kernbach, D. Häbe, O. Kernbach, R. Thenius, G. Radspieler, T. Kimura, and T. Schmickl. Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research*, 32(1):35–55, 2013.

R. Z. Khas'minskii. Necessary and sufficient conditions for the asymptotic stability of linear stochastic systems. *Theory of Probability & Its Applications*, 12(1):144–147, 1967.

M. Klotsman and A. Tal. Animation of flocks flying in line formations. *Artificial Life*, 18(1):91–105, 2011.

D. Lancet. Life without the double helix: Oparin revisited, 2003. URL `http://ool.weizmann.ac.il/publications.html`.

W. B. Langdon and R. Poli. Evolving problems to learn about particle swarm optimizers and other search algorithms. *Evolutionary Computation, IEEE Transactions on*, 11(5):561–578, 2007.

C. G. Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1):12–37, 1990.

A. Levina, J. Herrmann, and T. Geisel. Dynamical synapses causing self-organized criticality in neural networks. *Nature Physics*, 3(12):857–860, 2007.

H. Levine, W.-J. Rappel, and I. Cohen. Self-organization in systems of self-propelled particles. *Physical Review E*, 63(1):017101, 2000.

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

T. Liao and T. Stutzle. Benchmark results for a simple hybrid algorithm on the cec 2013 benchmark set for real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1938–1944. IEEE, 2013.

B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang. Improved particle swarm optimization combined with chaos. *Chaos, Solitons & Fractals*, 25(5):1261–1271, 2005.

Q. Liu. Order-2 stability analysis of particle swarm optimization. *Evolutionary computation*, 2014.

I. Loshchilov. CMA-ES with restarts for solving CEC 2013 benchmark problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 369–376. Ieee, 2013.

M. Lovbjerg and T. Krink. Extending particle swarm optimisers with self-organized criticality. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1588–1593. IEEE, 2002.

J. Lutkenhaus. Tinkering with acellular division. *Science*, 320(5877):755–756, 2008.

F. E. Lytle. Sum of two uniform densities, 2001. URL `https://www.chem.purdue.edu/courses/chm621/text/stat/funcs/twovar/sum2uniform.htm`. [Online; accessed 26-April-2016].

D. Ma. Examples of convolution (continuous case), 2011. URL `https://probabilityexam.wordpress.com/2011/05/26/examples-of-convolution-continuous-case/`. [Online; accessed 26-April-2016].

S. Maeda, Y. Hara, R. Yoshida, and S. Hashimoto. Chemical robot-design of peristaltic polymer gel actuator. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4325–4330. IEEE, 2009.

M. Maeterlinck. *The life of the bee*. Dodd, Mead, 1901.

K. Malan and A. Engelbrecht. Characterising the searchability of continuous optimisation problems for pso. *Swarm Intelligence*, 8(4):275–302, 2014.

B. McMullin. SCL: An artificial chemistry in swarm, 1997.

O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836. ACM, 2011.

M. Miller and B. Bassler. Quorum sensing in bacteria. *Annual Reviews in Microbiology*, 55(1):165–199, 2001.

Y. Mohan and S. Ponnambalam. An extensive review of research in swarm robotics. In *Nature & Biologically Inspired Computing*, pages 140–145. IEEE, 2009.

N. P. Money. More g's than the space shuttle: ballistospore discharge. *Mycologia*, pages 547–558, 1998.

C. K. Monson and K. D. Seppi. Exposing origin-seeking bias in pso. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 241–248. ACM, 2005.

P. A. Moore, N. Scholz, and J. Atema. Chemical orientation of lobsters, homarus americanus, in turbulent odor plumes. *Journal of Chemical Ecology*, 17(7): 1293–1307, 1991.

J. P. Newman and H. Sayama. Effect of sensory blind zones on milling behavior in a dynamic self-propelled particle model. *Physical Review E*, 78(1):011913, 2008.

M. E. Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

Z. Olami, H. J. S. Feder, and K. Christensen. Self-organized criticality in a continuous, nonconservative cellular automaton modeling earthquakes. *Phys. Rev. Lett.*, 68: 1244–1247, Feb 1992. doi: 10.1103/PhysRevLett.68.1244. URL `http://link.aps.org/doi/10.1103/PhysRevLett.68.1244`.

G. Papa and J. Silc. The parameter-less evolutionary search for real-parameter single objective optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1131–1137. IEEE, 2013.

R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008:3, 2008.

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

A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791. IEEE, 2005.

S. Rasmussen, M. A. Bedau, L. Chen, D. Deamer, D. C. Krakauer, N. H. Packard, P. F. Stadler, et al. *Protocells: bridging nonliving and living matter*. Mit Press Cambridge, 2009.

C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

T. J. Richer and T. M. Blackwell. The lévy particle swarm. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 808–815. IEEE, 2006.

J. Riget and J. S. Vesterstrøm. A diversity-guided particle swarm optimizer — the ARPSO. *EVALife Technical Report no. 2002-02*, 2002.

M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

T. J. Rudge, F. Federici, P. J. Steiner, A. Kan, and J. Haseloff. Cell polarity-driven instability generates self-organized, fractal patterning of cell layers. *ACS Synthetic Biology*, 2(12):705–714, 2013. doi: 10.1021/sb400030p.

E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer, 2005.

H. Sayama. Swarm chemistry. *Artificial Life*, 15(1):105–114, 2009. ISSN 1064-5462. URL `http://search.ebscohost.com/login.aspx?direct=true&db=mnh&AN=18855565&site=ehost-live`.

H. Sayama. Swarm chemistry evolving. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, pages 32–33, 2010.

H. Sayama. Swarm-based morphogenetic artificial life. In R. Doursat, H. Sayama, and O. Michel, editors, *Morphogenetic Engineering*, Understanding Complex Systems,

pages 191–208. Springer Berlin Heidelberg, 2012a. ISBN 978-3-642-33901-1. doi: 10.1007/978-3-642-33902-8_8. URL http://dx.doi.org/10.1007/978-3-642-33902-8_8.

H. Sayama. Morphologies of self-organizing swarms in 3d swarm chemistry. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*, pages 577–584. ACM, 2012b.

H. Sayama. Swarm chemistry homepage, 2015. URL http://bingweb.binghamton.edu/~sayama/SwarmChemistry/.

T. Schmickl and K. Crailsheim. Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. *Autonomous Robots*, 25(1-2):171–188, 2008.

T. Schmickl and H. Hamann. Beeclust: A swarm algorithm derived from honeybees. *Bio-inspired Computing and Communication Networks. CRC Press*, 2011.

T. Seeley and P. Visscher. Group decision making in nest-site selection by honey bees. *Apidologie*, 35(2):101–116, 2004.

T. Seeley, P. Visscher, and K. Passino. Group decision making in honey bee swarms: When 10,000 bees go house hunting, how do they cooperatively choose their new nesting site? *American Scientist*, pages 220–229, 2006.

T. Seeley et al. The wisdom of the hive: the social physiology of honey bee colonies. *The wisdom of the hive: the social physiology of honey bee colonies.*, 1995.

T. D. Seeley, S. Camazine, and J. Sneyd. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290, 1991.

D. Segré, D. Lancet*, O. Kedem, and Y. Pilpel. Graded autocatalysis replication domain (gard): kinetic analysis of self-replication in mutually catalytic sets. *Origins of Life and Evolution of Biospheres*, 28(4):501–514, 1998.

D. Segré, D. Ben-Eli, D. W. Deamer, and D. Lancet. The lipid world. *Origins of Life and Evolution of Biospheres*, 31(1):119–145, 2001.

W. L. Shew and D. Plenz. The functional benefits of criticality in the cortex. *The Neuroscientist*, 19(1):88–100, 2013.

W. L. Shew, H. Yang, S. Yu, R. Roy, and D. Plenz. Information capacity and transmission are maximized in balanced cortical networks with neuronal avalanches. *The Journal of Neuroscience*, 31(1):55–63, 2011.

Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.

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

A. Stabentheiner, H. Kovac, and R. Brodschneider. Honeybee colony thermoregulation–regulatory mechanisms and contribution of individuals in dependence on age, location and thermal stress. *PLoS ONE*, 5(1), 01 2010.

P. T. Starks, R. N. Johnson, A. J. Siegel, and M. M. Decelle. Heat shielding: a task for youngsters. *Behavioral Ecology*, 16(1):128–132, 2005.

F. Stepanek. Chemical swarm robots: Design, synthesis and functionality. *Sci Pharm*, 78:544, 2010.

D. Thompson. *On growth and form.* Cambridge University Press, 1917.

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

V. N. Tutubalin. On limit theorems for the product of random matrices. *Theory of Probability & Its Applications*, 10(1):15–27, 1965.

J. Tvrdík and R. Polakova. Competitive differential evolution applied to CEC 2013 problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1651–1657. IEEE, 2013.

J. Vanneste. Estimating generalized lyapunov exponents for products of random matrices. *Physical Review E*, 81(3):036701, 2010.

T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6): 1226–1229, 1995.

C.-R. Wang, C.-L. Zhou, and J.-W. Ma. An improved artificial fish-swarm algorithm and its application in feed-forward neural networks. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 5, pages 2890–2894. IEEE, 2005.

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

Wikipedia. Flocking (behavior) — wikipedia, the free encyclopedia, 2014. URL `http://en.wikipedia.org/w/index.php?title=Flocking_(behavior)&oldid=622633795`. [Online; accessed 1-December-2014].

Wikipedia. Ant colony optimization algorithms, 2015a. URL `https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#Applications`. [Online; accessed 09-November-2015].

Wikipedia. Bees algorithm, 2015b. URL `https://en.wikipedia.org/wiki/Bees_algorithm#Applications`. [Online; accessed 09-November-2015].

Wikipedia. Lyapunov exponent, 2016. URL `https://en.wikipedia.org/wiki/Lyapunov_exponent`. [Online; accessed 26-April-2016].

E. O. Wilson. The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1): 89–98, 1984.

S. Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):1–35, 1984.

C. Worasucheep. A particle swarm optimization with stagnation detection and dispersion. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 424–429. IEEE, 2008.

X.-S. Yang. Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications*, pages 169–178. Springer, 2009.

X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

X.-S. Yang and S. Deb. Cuckoo search via Lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics and Automation Magazine*, 14(1):43–52, 2007.

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