Fitness Landscape-Based Analysis of Nature-Inspired Algorithms

Matthew James Crossley

A thesis submitted in partial fulfilment of the requirements of the Manchester Metropolitan University for the degree of Doctor of Philosophy.

School of Computing, Mathematics and Digital Technology

Manchester Metropolitan University

2014

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I wish to offer my sincerest, and most heartfelt, thanks to my supervisor, Prof. Martyn Amos. Without his encouragement, counsel and unfailing ability to restore faith in myself on the 'rare' occasions I found myself at my wits end, there is simply no way I would have had the courage to follow this work through to the end. I hope that we continue to have opportunities to work together, as everything so far has genuinely been a delight.

Special thanks also to Prof. Joanna Verran, who helped me to explore my potential as a science communicator. Without her help, there would be no Monsters, Microbiology and Mathematics Team, and so many of the experiences I hold so close to my heart from my time as a Ph.D. candidate would not have been possible.

My family provided the necessary unconditional love and support not just throughout the Ph.D. years, but in the years beforehand, too. For their frequent, necessary reminders that achieving one's best is all one can aim for, and for always being there when the best occasionally just wasn't enough, I thank Mum, Dad, Abi and Aimée, for absolutely everything.

A thank you also to colleagues past and present, who were always willing to listen to my complaints regardless of the number of times they had heard them before, and for offering invaluable advice on the most mundane of issues. Pete, Ángel, Danny, Patrick, Kate, Dave, James, Margaret and Naomi, thank you.

Last but, as the cliché states, by no means least, a thank you to my friends, for their eternal belief in me and understanding in times when I seemingly disappeared off the face of the planet, busy with work. A huge, extra special thank you to Steven, David, Chris, Steve, Carmel, Stacey, Naomi, Jamie, Phil, Rosie, Karen and Katy.

The work of this thesis was supported by funding from the Dalton Research Institute, for which I am wholly grateful.

Declarations

This thesis is submitted to Manchester Metropolitan University in support of my application for admission to the degree of Doctor of Philosophy. No part of it has been submitted in support of an application for another degree or qualication of this or any other institution of learning. Parts of the thesis appeared in the following refereed papers in which my own work was that of a full pro-rata contributor:

Content in Chapter 4 was published as Crossley, M., Nisbet, A., and Amos, M. (2013). Fitness landscape-based characterisation of nature-inspired algorithms. In Tomassini, M., Antonioni, A., Daolio, F., and Buesser, P., editors, *Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA '13), Lausanne, Switzerland, April 4-6, 2013. Lecture Notes in Computer Science, Vol. 7824,* pages 110-119. Springer.

Content in Chapter 5 was published as Crossley, M., Nisbet, A., and Amos, M. (2013). Quantifying the impact of parameter tuning on nature-inspired algorithms. In Lio, P., Miglino, O., Nicosia, G., Nolfi, S., and Pavone, M., editors, *Advances in Artificial Life, ECAL 2013,* pages 925-932. MIT Press.

Abstract

As the number of nature-inspired algorithms increases so does the need to characterise these algorithms. A rigorous process to characterise algorithms helps practitioners decide which algorithms may offer a good fit for their given problem. One approach is to relate the characteristics of a problem's associated fitness landscape with the performance of an algorithm.

The aim of this thesis is to capitalise on the notion of fitness landscape characteristics as a technique for analysing algorithm performance, and to provide a novel algorithm- and problem-independent methodology that can be used to present the strengths and weaknesses of an algorithm. The methodology was tested by developing a portfolio of six nature-inspired algorithms commonly used to solve continuous optimisation problems. This portfolio includes the performance of these algorithms with parameters both "out of the box" and after they have been tuned using an automated tuning technique. Each of the algorithms shows a different "resilience" profile to the landscape characteristics, and responds differently to the tuning process. In order to provide a more practical way to utilise the portfolio an automated "ranking" methodology based on two machine learning techniques was developed. Using estimates of the fitness landscape characteristics on benchmark problems, the best algorithm to use is estimated, and compared with the actual performance of each algorithm. While results show that predicting algorithm performance is difficult, the results are promising, and show that this is an area worth exploring further.

This methodology has significant advantages over the current practice of demonstrating novel algorithm performance on benchmark problems, most importantly offering a practical, generalised overview of the algorithm to a potential practitioner. Choosing to use a technique such as the one demonstrated here when presenting a novel algorithm could greatly ease the problem of algorithm selection.

Chapter 1

Introduction

The number of nature-inspired algorithms for optimisation is growing continuously, as researchers develop novel algorithms, modify existing algorithms and hybridise algorithms to gain improved accuracy. This rapid expansion of the space of possible algorithms presents a new problem to practitioners: *Which algorithm should I use for my problem, and why?* This question is exacerbated by the arguments presented in the No Free Lunch Theorem Wolpert and Macready (1997), which states that where any algorithm shows improved performance on one class of problems, it must (on average) show reduced performance on another. The summary of the theorem is that, by choosing the correct algorithm for a particular problem class, better performance can be obtained than using naive selection.

It has also been argued that the very process of developing "novel" metaheuristics is harmful to the field, with newer techniques not being thoroughly explored, or with algorithms accidentally being recreated under new terminology, due to a lack of a rigorous process under which to analyse algorithm performance and behaviours (Sörensen, 2013).

Often, when a novel algorithm is proposed, results from a handful of benchmark functions, or results on a particular problem set, are used to provide comparisons against a limited set of other similar algorithms. This provides little practical information on the performance of the algorithm, leaving unanswered the question of *which types of problem is this algorithm best-suited to?*

By offering an algorithm-independent methodology for analysis, based on a landscape generation technique, it is shown that it is possible to provide an in-depth *profile* of an algorithm's performance which relates to characteristics of the fitness landscape of a problem, highlighting the relative strengths (and weaknesses) of an algorithm. Furthermore, this technique can be used to compare and contrast algorithms with others, or even different versions of the same algorithm. This offers a *practical* insight into which algorithm to choose, as the information about algorithm performance relates directly to characteristics of the problem. It can also highlight where algorithms *differ* from each other, potentially offering a solution to the problem of "overcrowding" in the nature-inspired algorithm field.

The fundamental research questions this thesis aims to answer are as follows:

1. To what extent can fitness landscape characteristics be used to establish a performance profile of an algorithm, and thus distinguish between different algorithms in terms of performance?

2. To what extent does tuning alter the performance profile of an algorithm with regard to each of the landscape characteristics defined?

3. To what extent can algorithm performance be predicted by using the defined landscape characteristics as input to a classification algorithm?

In answering these questions, it is hoped that this work provides the first steps towards a novel methodology for algorithm designers to compare algorithms which is independent of specific benchmark problems (removing the problem of comparing algorithms which were not introduced by testing on similar problems), and a technique for presenting novel algorithms which highlights the strengths and weaknesses of an algorithm and, where possible, the specific usefulness of certain 'features' of an algorithm. The final question, regarding prediction, provides a starting point for a possible prediction technique which, while still posing many technical challenges, offers fellow researchers a potential alternative strategy to current predictive methods.

The remainder of this thesis is organised as follows:

- In Chapter 2 the notion of nature-inspired algorithms is introduced, with descriptions of how and why they are used as an optimisation strategy. Six specific nature-inspired algorithms are introduced, with specific attention to their their inspirations and how these particular algorithms work. These six algorithms are then used throughout the remaining work as the algorithms under study.

- In Chapter 3 a methodology is presented which describes one approach for analysing algorithms based on fitness landscape characteristics. This methodology, or slight variants thereof, is then used in the three experimental chapters of this thesis, which affirms its usefulness and tests the extent to which landscape characteristic analysis can be used by algorithm designers to better present, or answer questions, about novel algorithms.

- In Chapter 4 the methodology introduced in Chapter 3 is tested by carrying out a performance analysis of the selected algorithms. This analysis is performed in terms of fitness landscape characteristics, using a landscape generation technique. By breaking down and relating algorithm performance to different aspects of the generated landscapes, is is shown that there is no universal way to choose a best algorithm. It is clear from the results that the landscape characteristics used in the generation technique are an appropriate basis for analysing the performance of these algorithms and that the methodology is an appropriate way to generate performance profiles of optimisation algorithms.

- In Chapter 5 the previous Chapter is expanded upon with further analysis focusing on the effect of *tuning* the parameters of each of the algorithms, offering a

more "realistic" study as algorithms are rarely used "out of the box". Whether to tune parameters or not is a significant debate in the implementation of nature-inspired algorithms, and by analysing the effect of tuning using the methodology presented in Chapter 3, it is shown that there are some algorithms which always need tuning, some which do not require tuning, and some which need only be tuned in certain (landscape characteristic dependant) circumstances. This also emphasises that the methodology is appropriate for use *within* algorithms.

- In Chapter 6 a feasibility study is presented on predicting algorithm performance using fitness landscape characteristics. It is found that although automated prediction is a challenging task promising results can be obtained with little effort when the fitness landscapes are representative of those used to generate the training data. This demonstrates again that the methodology presented in Chapter 3, and associated algorithm profiles, could form the basis a strategy for predicting algorithm performance.

- In Chapter 7 the thesis is summarised, with some concluding remarks and suggestions for future work.

Chapter 2

Nature-Inspired Algorithms

2.1 Introduction

Broadly speaking, there are three classifications of techniques used to solve optimisation problems: Methods designed to reach *exact* solutions mathematically, methods which reach a solution *iteratively* and methods designed not to provide an exact solution, but rather to *approximate* a solution - that is, they find a solution that's "good enough" but do so when the former two methodologies may not be appropriate (Gill et al., 1981).

The field of *linear programming* offered the first attempts to solve optimisation problems mathematically, and dates back to Kantorovich (1940). With the publication of the Simplex method, by Dantzig (1965), a method for calculating exact solutions to problems became available, and this marked the start of optimisation as a promising area of research, with continued development and extensions to the Simplex method. One prominent question surrounding the Simplex method regarded its complexity, particularly the complexity scaling, of the method, and how intractable problems became as they increased in size (i.e. number of variables). It has been found to generally converge in polynomial time (Wright and Nocedal, 1999; Forsgren et al., 2002), although its worst-case complexity is exponential (Klee and Minty, 1970).

An alternative are heuristics, designed to reach an *approximate* solution rather than an exact solution. These methods are often used when traditional methods as described above either prove too slow, or fail to find an exact solution (Pearl, 1984).

Many heuristic methods are inspired by nature, and it is these algorithms that form the focus of this thesis. The over-arching concept of *nature-inspired* algorithms is that natural processes can offer insight into the various "problem-solving" techniques used by living systems, for example; The way bacteria use chemical gradients to sense food concentrations (Adler, 1966), the process of evolution itself (which naturally selects the 'best solution' from a number of candidates (Fisher, 1999)) and the way insects carry out their day-to-day tasks (Bonabeau et al., 2000). Like many other disciplines which also borrow ideas from nature, the idea here is not necessarily to *mimic* nature exactly, rather, ideas are drawn from nature and these ideas are used as a basis for *inspiration* to design algorithms which solve problems in a similar manner. In doing so, at least in

theory, algorithm designers capitalise on the refinements and ingenuity of the natural process itself.

As there are a number of different nature-inspired algorithms, there is no straightforward answer to how and why they are used, but they are often used when a "good enough" solution will do, or an exact solution is not available. With an increasing number of nature-inspired algorithms available to practitioners, and increasing availability of resources such as code repositories, implementation guides, and more in-depth studies into the exact behaviour of these algorithms, using a nature-inspired algorithm in place of a traditional technique is becoming a more realistic option.

The decision to use a nature-inspired algorithm may be informed by a number of factors, including (but not limited to)

1. Which nature-inspired algorithm should be used?

    - This decision is one of the most difficult facing a potential practitioner, and is one which is not getting any easier as novel nature-inspired algorithms are proposed.
    - A practitioner can reduce their potential selection based on whether their problem is continuous or discrete.

2. Are there variants, hybridisations or components of the algorithm a practitioner needs to choose from? (Goldberg and Deb, 1991)

3. What values should the parameters for the algorithm take? (Eiben et al., 1999)

    - Some algorithms are more parameter sensitive than others.
    - There are methods to automatically determine parameter values, but these add to the overall computation required before a solution can be reached.

Rice (1976) discusses the notion of the algorithm selection problem, proposing a series of models for selecting the *best* (or at least, an effective or good) algorithm for a given situation. The relationship between problem, algorithm and performance is discussed in detail, with the importance of a *mapping* between the three the key to solving the problem. It is by this that the inspiration to develop a *portfolio* of algorithm performance arises, in a way such that algorithm performance can be related directly to characteristics of a problem. Using this, a practitioner could estimate the characteristics of their problem, examine the portfolio, and make an informed decision as to which algorithm could prove to be effective for their situation.

In the rest of this Chapter the underlying concepts of the nature-inspired algorithms that are selected for examination in this thesis are described. This includes a discussion of the origin, a description of the algorithm, and some example applications of each algorithm.

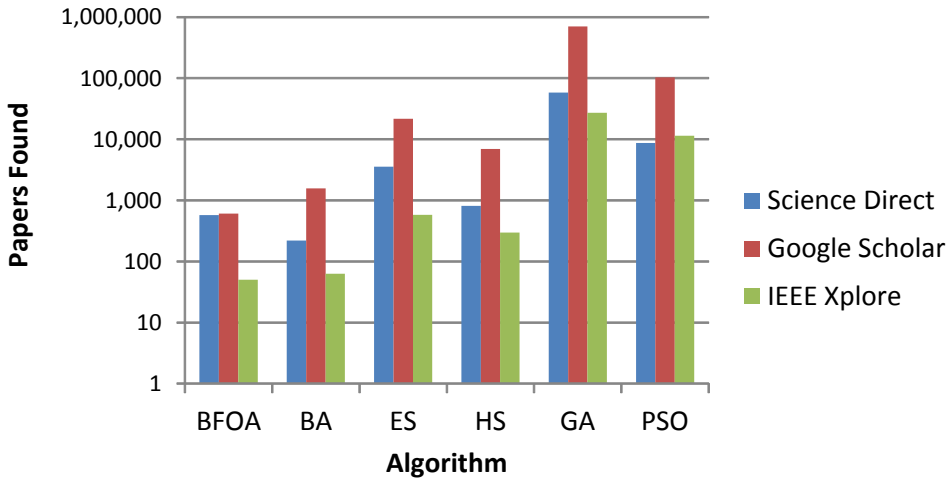


Figure 2.1: Number of papers found on six nature-inspired algorithms by three leading publication databases.

2.2 Algorithm Descriptions

Six nature-inspired algorithms are selected for detailed description and analysis, two of which are *evolutionary* algorithms (Evolution Strategies (ES) and Genetic Algorithm (GA)), three of which are generally classed as *swarm* or *social based* algorithms (Bacterial Foraging Optimisation Algorithm (BFOA), Bees Algorithm (BA) and Particle Swarm Optimisation (PSO)) and lastly an algorithm which is neither evolutionary nor swarm-based, though draws on the concepts of both, and is often classed as a *physical* algorithm, the Harmony Search (HS) algorithm. Of these algorithms, some account for a substantially larger section of the field than others, with the more recently proposed algorithms having fewer applications than those that originated the field (Yang, 2010).

To gain a brief overview of the relative size of the bodies of work on each of these algorithms, a search of each algorithm's name was performed on three leading publication databases; (1) Science Direct[1], (2) Google Scholar[2] and (3) IEEE Xplore[3]. The results are depicted in Fig. 2.1, showing that GAs have, by far, the most published literature, with PSO in second place. The relative newcomers, BA and BFOA fall significantly behind, with very few published papers returned by the searches.

One of the reasons for GAs having a high paper count compared to the other algorithms is its frequent use as the "benchmark" algorithm in many applications. As such, when a new technique is used, it is often compared to the performance of a GA as a way of validating its performance. This is particularly noticeable due to the large number of extra results from Google Scholar, which indexes entire texts, and many of these could not feature GA as the focal point of the research, but rather, as the comparison. A by-product of this, however, is that GAs have been applied to a wide range of problems and applications, and have been studied in-depth in this manner.

The six algorithms were selected based on (i) their apparant similarity with each other, (ii) their differences in computational properties (for which there is no clear

[1]http://www.sciencedirect.com
[2]http://scholar.google.co.uk/
[3]http://ieeexplore.ieee.org/

analysis of the performance benefits/hindrances), and (iii) their popularity for use on continuous function optimisation problems. The foundation for further classification of the algorithms is derived from the work of Blum and Roli (2003). Each of the algorithms selected falls into the over-arching category of either *trajectory methods* or *population-based methods* (in fact, all except HS are population-based, but this particular algorithm shares many characteristics of population-based algorithms). Further computational properties of the algorithms are described by Blum and Roli, and to give an idea of how the algorithms selected differ from each other, Table 2.1 contains a listing of the algorithm and a short description of how, where possible, each algorithm fulfils each particular property.

It may be questioned why certain other popular algorithms have not been included in this study. A popular technique not included in this study is that of artificial neural networks. Artificial neural networks were not chosen for inclusion in this study as they are already distinct in nature from population-based algorithms, and, although it is possible to use neural networks for optimisation (Joya et al., 2002), it is more common to see them used for either *fitness approximation* (Jin, 2005) or to apply a population-based algorithm to a neural network as a means of training it - many examples of this are demonstrated in the individual algorithm example sections throughout this Chapter). It should be noted that the methodology presented in Chapter 3 is algorithm independent and, as such, any additional algorithms could be incorporated into the study at a later date.

A brief description of the algorithms, including their origin, inspiration and an overview of their most popular applications, now follows.

2.2.1 Bacterial Foraging Optimisation Algorithm

The seminal BFOA algorithm was presented by Liu and Passino (2002); Passino (2002), suggesting an algorithm inspired by the behaviour of bacteria - specifically *Escherichia coli* (*E. coli*) and *Myxococcus xanthus* (*M. xanthus*). The motivation of the algorithm is that bacteria engage in social foraging, and that this has been shown to provide a method for climbing chemical gradients (Grünbaum, 1998), and so their behaviour is a reasonable inspiration for a social foraging optimisation algorithm. Through a process of modelling and examining bacteria behaviour, and relating this to the optimisation process, the BFOA was developed, for use on continuous optimisation problems.

Bacteria, through a process of sensing chemicals (chemotaxis, (Adler, 1966)), traverse an environment attempting to move away from areas containing harmful substances and towards areas of high food concentration. It is this process that forms the main inspiration for BFOA, which is in essence a hill-climbing algorithm. This process is modelled, algorithmically, in a simplistic manner: At each step of the algorithm, a bacterium selects a random direction in which to swim, and commences swimming (Berg, 2000). If this new position is "better" than the previous position, the bacterium continues to swim in this direction. If the position is worse, the bacterium halts, waiting for the next step to select a new direction. Parameters such as the step size and

Table 2.1: Computational properties of the selected algorithms.

	BFOA	BA	ES	GA	HS	PSO
Number of Parameters	10	6	2	4	4	4
Population-Based?	Yes	Yes	Yes	Yes	No	Yes
Description of the Individuals	Real-coded complete solutions	Real-coded complete solutions	Real-coded complete solutions	Binary-encoded complete solutions	Real-coded complete solutions	Real-coded complete solutions
Evolution Process	All solutions carried forward	No solutions carried forward	Mixture of new and old solutions carried forward	Mixture of new and old solutions carried forward	-	All solutions carried forward
Information Sources	At 'reproduction' steps, successful individuals are cloned.	New solutions are generated within a small range of previous generation's good solutions.	New solutions are generated as mutations of single parent selection.	New solutions are generated as crossover of two-parent selections.	New solutions generated based on memory of good solutions.	-
Intensification Strategy	Individual solutions perform gradient-based local search.	-	-	-	-	Individual solutions perform gradient-based local search.
Diversification Strategy	Solutions can be removed from the population and randomly re-sampled from throughout the entire solution space based on parameterisation.	Each generation consists of a number of solutions drawn randomly from throughout the entire solution space.	-	Child solutions are sometimes perturbed by the mutation operator.	Solutions can be randomly sampled from throughout the entire solution space based on parameterisation.	-

swim length control the exact behaviour of the chemotactic phases of the algorithm.

While chemotaxis forms the main component of the algorithm's inspiration, there are other factors which can be modelled that enhance the ability of the algorithm to perform in an optimisation context. The first of these is reproduction; Where food levels are high (or, where solutions are promising), bacteria reproduce at a high rate - by "killing off" bacteria in unpromising areas (those performing worst), and *cloning* the better performing bacteria, this allows the algorithm to explore promising areas more thoroughly while ignoring areas of little interest.

Another key feature of the BFOA are elimination-dispersal events. Due to the scale of bacteria relative to the world around them, events often occur which can redistribute/remove the population en masse, for example water or animals moving populations of bacteria from one location to another, or sudden local changes in temperature harming large numbers of bacteria in a localised region. These events can be used as a way of "shuffling" the population - in an optimisation context, this offers a local optima avoidance technique similar to that of random restarts (Hu et al., 2009), and is controlled by a parameter which dictates the chance a bacterium is eliminated/redistributed during an elimination-dispersal event.

The last component of the algorithm is an individual bacterium's interactions with the other members of its colony. Attractant and repellent chemicals are secreted by individual bacterium, and these chemicals are also sensed by other members of the colony, enabling a true "swarming" behaviour. For example, bacterium in areas of high food density may release attractants - following the optimisation metaphor, bacterium in a region of promising solutions will "beckon" other members of the population to explore the area more thoroughly. The four cell-cell interaction parameters control this behaviour, dictating the range individual bacterium's chemicals are spread and the reliance other members of the colony place on these chemicals.

Pseudocode for the BFOA is shown in Algorithm 1.

A second version, presented by Muller et al. (2002), is not labelled as the BFOA but rather the Bacteria Chemotaxis Algorithm (BCA) This version shares a lot of the ideas of the Passino (2002) implementation, in that the main inspiration for the core of the algorithm is the process of bacterial movement, driven by chemotactic sensing. Using a more rigorous mathematical model of bacterial movement, the BCA examines the probabilities of a bacterium changing direction, and selects a new direction based on a Gaussian distribution. Features such as reproduction and elimination-dispersal are not components of the BCA. The BCA is not found to compete with other algorithms at the time of its presentation, and subsequently although it has been used somewhat, it is not as popular as Passino's version of the algorithm.

Applications

As a relative newcomer, the BFOA suffers from a lack of diversity in application areas, with applications to power systems appearing most frequently. Mishra and Bhende (2007) apply the BFOA to multi-machine power system stabilizer design, using a

```
for each bacterium do
    generate random solution;
    evaluate fitness;
end
for number elimination-dispersal events do
    for number reproductive steps do
        for number chemotactic steps do
            for each bacterium do
                choose random direction;
                repeat
                    update position in random direction by step size;
                    calculate cell-cell interaction;
                    evaluate fitness;
                until fitness worsens OR maximum swim length reached;
            end
        end
        sort bacteria by fitness;
        remove R worst bacteria;
        clone R best bacteria;
    end
    for each bacterium do
        if bacterium should be redistributed then
            generate new random solution;
            evaluate fitness;
        end
    end
end
```

Algorithm 1: Bacterial foraging optimisation algorithm pseudocode.

slightly adapted version of the algorithm which adapts the run length dynamically according to heuristic rules. Results are compared with those found using a GA, and are noted to be much better. Das et al. (2008) later also apply the BFOA to the design of power system stabilizers, although this time the comparison is made to PSO and a variant of PSO referred to as small-population-based particle swarm optimisation (SPPSO). Although SPPSO is the overall "winner" of this competitive testing, it is noted that BFOA also gives "...robust damping performance for various operating conditions and disturbances."

A different problem within the power systems field, that of power system reconfiguration and loss minimisation, is tackled by Kumar and Jayabarathi (2012). Comparisons are made between ten algorithms here, including notably a set of refined versions of the GA. The BFOA provides the least power loss, and is described as having "fast and effective convergence."

2.2.2 Bees Algorithm

Introduction and Origin

The BA is one of many nature-inspired algorithms inspired by the behaviour of bees. The algorithm was first proposed by Pham et al. (2005, 2006b), where it is tested on a number of benchmark functions. It was found to outperform the rival algorithms, specifically genetic algorithms and ant colony systems. Although not the first algorithm to be inspired by the behaviour of bees, this is the first to be based on the food locating and collecting behaviour. The strengths and weaknesses of the BA are discussed - good local optima avoidance is cited as a strength (as the algorithm relies very little on gradient information), and results show good accuracy of solutions. The only stated weakness is that the algorithm has several parameters, though the authors claim these can be configured quickly using a small number of trials.

Unlike other bee-inspired algorithms, such as the Artificial Bee Colony algorithm (Karaboga and Basturk, 2007) and the Honey Bee Mating Optimisation algorithm (Haddad et al., 2006), the BA focuses specifically and solely on the nectar-collecting behaviour of the honey bee, neglecting to include many of the other interesting behaviours of the bee (Winston, 1991; Seeley, 2009). While collecting nectar, the honey bees perform optimisation on a basic level: As a colony, they need to be harvesting nectar from nectar sites with the most promising harvest (i.e. areas with a good solution).

During the process of harvesting, bee colonies send out a number of "scout" bees, whose task it is to (randomly) explore patches of flowers and evaluate these flowers for their richness of nectar (Janson et al., 2005). These scout bees then return to the colony, and report their findings, through the process of the waggle dance (Riley et al., 2005). Using the information gained from the waggle dance (which reports both the quality and location of their explored sites), bees tasked with harvesting nectar then make decisions about which patches to harvest nectar from, and commence harvesting.

```
for each bee do
    generate random solution;
    evaluate fitness;
end
while stopping criteria are not met do
    sort bees by fitness;
    for number of sites (n) do
        site = nth best bee;
        if elite site then
            siteBees = eBees;
        else
            siteBees = oBees;
        end
        for siteBees number of bees do
            generate random solution in patch size range of site;
            evaluate fitness;
        end
    end
    for remaining bees (scouts) do
        generate random solution;
        evaluate fitness;
    end
    reduce patch size;
end
```

Algorithm 2: Bees algorithm pseudocode.

The optimisation metaphor follows quite logically here, with promising flower patches representing areas of promising solutions.

While harvesting, bees continually report on the quality of their sites, and similarly, scout bees continue to assess new sites for potentially better areas of exploration, acting as both a local search (harvesting bees) and a global search (scout bees) simultaneously, with harvesting bees re-evaluating their site choice if necessary. The trade-off between local and global search is tweaked using parameters which control the number of the population assigned to the scouting task.

Pseudocode for the BA is shown in Algorithm 2.

Applications

Pham has worked with a number of researchers to show that the BA is capable of solving a wide range of problems. Some early applications for the BA were heavily focused on hybrid use with learning algorithms. One combination of hybridisation includes the use of BA as a method of training an artificial neural network, as in Pham et al. (2006c) and Pham et al. (2006d), where in the latter a neutral network is developed to recognise defects in wood (i.e. image analysis). Compared to a traditional technique for training ANNs (back propagation), the same accuracy is achieved, showing that BA is a capable technique. Another learning technique benefiting from the inclusion of the BA is described in Pham et al. (2007d), which again tackles the problem of

classifying wood defects, this time using a support vector machine (SVM) approach. Parameters of the SVM are optimised using the BA, providing a considerable increase in accuracy. A third learning technique, learning vector quantisation (LVQ) networks, benefit from the use of BA for parameter optimisation in the application of recognising control chart patterns (Pham et al., 2006a). Again, a noted improvement in learning accuracy and test accuracy is produced when parameters are optimised using the BA.

Diversifying from the field of machine learning, the BA has also been applied to problems in the domain of engineering. Initially described in Pham et al. (2007b), the BA has been used to design welded beams (Pham and Ghanbarzadeh, 2007) and in the design of cellular manufacturing systems (Pham et al., 2007c).

The BA has also been applied scheduling problems, including job scheduling (Pham et al., 2007a) - notable, as this shows the BA is also suitable for solving combinatorial optimisation problems. Compared to a variety of other algorithms (discrete particle swarm optimisation, tabu search, a genetic algorithm and hybridisations of tabu search and genetic algorithms), the BA produced better results, and the authors also found the BA to be more stable and robust than the other algorithms tested.

Other applications include workload balancing (Baykasoglu et al., 2009), assembly line balancing (Özbakır and Tapkan, 2011) and control systems for robotics (Fahmy et al., 2012).

2.2.3 Evolution Strategies

Introduction and Origin

ES was primarily formulated in the 60s and 70s, with work including Rechenberg (1971) and Schwefel (1977) (the latter republished in English (Schwefel, 1981)), providing the foundation for what has become a rich field of algorithmic exploration (Bäck, 1996; Beyer and Schwefel, 2002). ES is inspired by the process of evolution as viewed at the species-level, rather than at the individual-level, and does not concern itself with the evolutionary mechanics of individuals - that is to say, it ignores the mechanics of individuals such as genomes, alleles, genes, etc. and instead an "overview" of evolution is presented.

Fundamentally, an ES algorithm is similar to other evolutionary algorithms. Initially, a population is generated based on a population size parameter. The population is usually generated by randomly sampling from all available solutions. In this "vanilla" form of ES (which is the variant under scrutiny) there are terminological conventions, as follows: This ES in particular is referred to as the $(\mu + \lambda)$ ES. μ is the number of parents, λ is the number of children, and the "+" signifies that the next generation is the best members of *both* the parents and children, whereas a "," would indicate that only the children would be selected to form the next generation.

To generate children, which forms the main process of the algorithm, a parent is selected using some selection mechanism (this could be as straightforward as always choosing the best of the current generation as the parent, choosing a random member

```
for population size do
    generate random solution;
    evaluate fitness;
end
while stopping criteria are not met do
    for number of children do
        select parent;
        generate child solution by mutating parent solution;
        evaluate fitness;
    end
    merge children and parents into one population;
    sort merged population by fitness;
    select population size best solutions as new population;
end
```

Algorithm 3: Evolution strategies pseudocode.

of the current generation as a parent, or generating a number of children for each parent proportionally based on the ratio of children to parents in the configuration of the algorithm). Once a parent is selected, the solution is mutated, generating the child solution. Many of the selection methods for choosing a parent are shared between others used by other evolutionary algorithms (Goldberg and Deb, 1991). The mutation strategy varies based on the problem representation, and is usually problem specific. For continuous optimisation, mutation could be assumed as adjusting the parent solution by a random variation within a given limit. Children and parents are then merged, with the *population size* best solutions forming the population for the next generation, where the process repeats until some stopping criteria is reached. Pseudocode for the ES is shown in Algorithm 3.

Applications

Much of the work on ES revolves around improvement and adaptation to the algorithm (Hansen and Ostermeier, 1996; Knowles and Corne, 1999), and investigation into the way in which the algorithm works . Despite this, ES has found some applications including the training of artificial neural networks (Mandischer, 2002; Lucas, 2005), structural design (Papadrakakis et al., 1998; Hasançebi, 2008), vehicle routing problems (Mester and Bräysy, 2007; Mester et al., 2007; Repoussis et al., 2010) and engineering problems (Papadrakakis et al., 1998; Kim et al., 2007; Chen and Chen, 2009).

2.2.4 Genetic Algorithm

Introduction and Origin

The oldest and most well-known of the nature-inspired algorithms, GAs have a huge volume of published work with a large number of applications and variations, and are often used as a "standard measure" for novel algorithms to compete against. The exact origin of the GA can be debated, since they existed in theoretical work for some time

before being put into practice. Many of the theories underpinning GAs stem from Holland's work on adaptive systems (Holland, 1962), with Goldberg (1989) providing what has come to be known as the classical book on GAs.

The inspiration behind GA is population level genetics, mimicking the process of natural selection, which differs from the macro evolution level inspiration used by ES. The idea here is that each member of the population "encodes" the solution to the problem in a structure similar to that of genetic material. In this sense, in terms of continuous optimisation, a segment of a population member's DNA may encode a particular value in a given dimension, for example.

The algorithm execution follows the pattern of a number of generations, with a number of essential components creating the drive towards an optimal solution. These components are selection, reproduction (crossover) and mutation, with a number of different strategies available for each of these components, discussed below. Pseudocode for a GA is shown in Algorithm 4.

The method of encoding the solution is the first problem a practitioner is faced with, and there are different encoding strategies that are suitable for different problems (Herrera et al. (1998)). The two most common representations are binary coded GAs (Goldberg and Holland, 1988) and real coded GAs (Mühlenbein and Schlierkamp-Voosen, 1993). In binary coded representation, the solution is represented as a string of bits, which is decoded appropriately into the solution as required. This method is simple to implement in terms of the GA, as it fits in with the metaphor; Standard operators all apply to the string of bits, as this is how the GA was first conceived to encode a solution. The main drawback of binary coded GAs arises when they are applied to continuous optimisation, as the length of the string of bits limits the precision of solution that can be generated, and it is in these instances that a real coded GA may be used instead. In a real-coded GA, genes are represented directly as real numbers, thus a chromosome is a vector of floating point numbers. This avoids the need to "decode" the string of bits, and solves the problem of dealing with precision when solving continuous optimisation problems. The problem here, however, is that the implementation of selection, crossover and mutation operators is completely different to those for binary coded GAs, with new interpretations of the methods for binary coded GAs required to handle real numbers.

Selection is the process used to choose which of the individuals from the population should be used for crossover, and as with all components there are a number of different selection techniques available to practitioners (Goldberg and Deb, 1991). All of these selection techniques have one thing in common, and that is to choose the fitter solutions as parents, echoing the "survival of the fittest" notion of natural evolution. Common selection techniques include the tournament technique, in which potential parents "compete" against each other, with the fittest selected - a common implementation of this being the binary tournament, where two potential parents are selected, with the fitter of the two being selected.

Crossover defines the method of combining two members of the population to create

"offspring", which is then added to the population (and, optionally mutated, as described below), with the effect of crossover studied in Mitchell et al. (1992). A common crossover technique is n-point crossover, in which a number of points (n) are selected at which to crossover the solution representation from both parents (Eshelman et al., 1989). In one-point crossover, for example, one point is selected; Anything before that point is taken from one parent, and anything after from the other. In two point, the switch between parents occurs twice, and so on. Another popular crossover technique is uniform crossover (Syswerda (1989)), in which genes are selected from parents on an *individual* basis - there is a 50% chance that a gene will be taken from parent one, and therefore a 50% chance that the gene will instead come from parent two.

Mutation is a key component for ensuring that the population is able to explore "fresh" solutions, and is based on the natural occurrence of genetic mutation. One of the simplest techniques for implementing mutation is to "bitflip" the encoded solution - i.e., iterate through the encoding and, based on a mutation chance parameter a bit should flip (swap from a zero to a one, or vice versa). Depending on the precise encoding technique used, mutations may make a major or minor change to the actual solution.

Parameters control various aspects of the different components, plus the interaction each component has on the algorithm as a whole. The crossover rate controls the likelihood that crossover occurs (where it doesn't, one parent is selected without crossover occurring) and the mutation rate controls the likelihood that individual bits are flipped when mutation is occurring. Generally, a high probability of crossover and a low probability of mutation is used, although this is problem dependant.

Applications

As one of the oldest nature-inspired algorithms, and the standard competitive test for novel algorithms, GAs have been applied to a very diverse range of problems, including both theoretical problems used for benchmarking and real-world applications.

Among the most common applications are job shop scheduling and timetabling (Fang et al., 1993; Hou et al., 1994; Cheng et al., 1996; Ross et al., 2003), engineering problems (Gen and Cheng, 2000), power systems (Nara et al., 1992; Walters and Sheble, 1993; Hassan et al., 2013), training and generating artificial neural networks (Yao, 1999; Stanley and Miikkulainen, 2002; Nasseri et al., 2008) and control systems (Grefenstette, 1986; Pan et al., 2011).

2.2.5 Harmony Search

Introduction and Origin

The origin of the HS is described in the paper by Geem and Kim (2001). The HS algorithm is suggested to work well for both continuous and combinatorial optimisation problems, and the proposed algorithm is tested on problems of both types - the travelling salesman problem, the design of a pipeline network, and a continuous function

```
for each member of population do
    generate random solution;
    evaluate fitness;
end
while stopping criteria not met do
    for each member of next generation do
        select parent1;
        select parent2;
        if should crossover then
            child is crossover of both parents;
        else
            child is clone of parent1;
        end
        if should mutate then
            mutate child;
        end
        evaluate fitness;
        merge children and parents into one population;
        sort merged population by fitness;
        select population size best solutions as new population;
    end
end
```

Algorithm 4: Genetic algorithm pseudocode.

minimisation problem. HS outperforms the tested existing techniques on the problems tried in this paper.

HS is inspired by the process of a group of musicians reaching a harmonious performance while improvising. By listening to the notes produced by each other, and adjusting the notes they themselves are playing, the sound waves produced by each individual musician eventually reach a point where they combine in a way such that they are aurally pleasing. Unlike some of the other selected algorithms, the conceptual link to optimisation is perhaps less straightforward. To translate from inspiration to optimisation, consider each dimension of the optimisation process as an individual musician, and each note they are currently playing as the value of the variable in that dimension. When a harmonious sound is produced, the objective function produces a good fitness value for the given variables.

Adding complexity are the additional levels of inspiration included in HS - one of which i s "pitch adjustment". A musician may, when improvising, choose to alter the pitch of a note rather than opting to play a different note entirely: This is mimicking in optimisation by selecting a new value for a variable *within a parameter-controlled range* of the previous variable, i.e. select a neighbouring value. In this sense, HS implements a local search, controlled by both the pitch adjustment rate and the method used to adjust the pitch.

A second imperative feature of HS is the "harmony memory" - a stored collection of previously played harmonies. When a new harmony is determined, it only enters harmony memory if it is better than the harmonies in the musician's memories. This

```
for harmony memory size do
    generate random solution;
    evaluate fitness;
end
while stopping critera are not met do
    if should choose from memory then
        choose random harmony from memory;
        if do pitch adjustment then
            adjust solution by range;
            evaluate fitness;
        end
    else
        generate random solution;
        evaluate fitness;
    end
    add new harmony to memory;
    sort memory by fitness;
    remove worst solution from memory;
end
```

Algorithm 5: Harmony search pseudocode.

harmony memory also serves as a "pool" of promising solutions (similar to the population in swarm-based algorithms) from which candidate solutions can be selected. The size of the harmony memory controls the greediness of the algorithm, while the harmony memory consideration rate controls the convergence rate of the algorithm.

Pseudocode for the HS is shown in Algorithm 5.

HS is revisited by Lee and Geem (2005), where its applicability to continuous engineering optimisation is the main topic of discussion. Advertising the benefits over gradient-based mathematical optimisation techniques (no derivative information necessary), and also the benefits over evolutionary techniques such as GA (consideration of an entire harmony memory rather than just two parent vectors), claims for the HS are backed up by strong results on a variety of benchmark problems including both continuous unconstrained functions, constrained functions and combinatorial problems.

Applications

A comprehensive survey of the applications of HS was presented by Manjarres et al. (2013).

The largest category of practical applications of HS is in the engineering domain, with HS for structural optimisation presented in Lee and Geem (2004). The advantage of using the HS algorithm for these problems is stated as primarily being the lack of reliance on gradient information and derivative information. A general application to the design of truss structures is posed, and HS is tested on some benchmark truss-structure problems against a varying number of competing algorithms, based on the specific problem. HS outperforms the traditional mathematical optimisation techniques tested, and simple GA based methods, but failed to outperform a fuzzy controlled GA.

The authors note that the HS used is in its basic form, and also note that while trusses are one specific sample of structural optimisation problem, there are many more for which it may prove a capable optimiser.

Stemming from this, the HS has been applied to the related problems of: designing steel sway frames (Saka, 2009), cellular beam design (Erdal et al., 2011), the design of shell and tube heat exchangers (Fesanghary et al., 2009), and multi-pass face-milling (Zarei et al., 2009).

Like BFOA, another large area of application for HS is power systems. The first application to a power systems problem appears in Coelho and Mariani (2009), applying HS to power economic load dispatch. HS (and a modified version proposed) both converge to good solutions, notably out-performing PSO and performing similarly compared to variants of a GA. HS has also been applied to combined heat and power economic dispatch problems (Vasebi et al., 2007; Khorram and Jaberipour, 2011), once again outperforming a GA, particularly when some adaptations are made to the algorithm to better accommodate problem-specific knowledge.

Other areas of note include water management (Geem, 2006; Tamer Ayvaz, 2009), and robotics (Tangpattanakul and Artrit, 2009; Tangpattanakul et al., 2010).

2.2.6 Particle Swarm Optimisation

Introduction and Origin

PSO is another major nature-inspired algorithm, first proposed by Kennedy and Eberhart (1995); Eberhart and Kennedy (1995) for optimisation of continuous functions.

Swarming (or flocking) as exhibited by birds, fish, herds and even humans is a behaviour believed to improve the process of "climbing gradients" (Grünbaum, 1998). This social behaviour forms the basis of the exploratory pattern in PSO, which is based on similar rules to those proposed by the artificial swarming that kick-started the field of agent-based modelling (Reynolds, 1987). In this sense, the metaphor here does not apply directly to the process of optimisation but rather, to the exploration pattern used to carry out the process of optimisation.

To create the swarming behaviour, particle positions are updated based on three factors: Their current velocity, their personal best position and the global best position. The personal best represents an individual's personal experiences, and the global best signifies the concept of 'publicized knowledge'. A preference to rely on personal experience over group experience (or vice versa) is controlled using parameters.

Pseudocode for the PSO is shown in Algorithm 6.

Applications

In the review by Poli (2008), applications of PSO are divided roughly into categories.

The largest area of application for PSO, according to this study, is in image and video analysis. One noted application is the inversion of ocean colour observations (Slade et al., 2004), in which improved results over a GA are noted and improvements

```
for each particle do
    generate random solution;
    evaluate fitness;
    store fitness as personal best;
    if global best then
        update global best;
    end
end
while stopping criteria not met do
    for each particle do
        update velocity;
        update position;
        evaluate fitness;
        if personal best then
            update personal best;
        end
        if global best then
            update global best;
        end
    end
end
```

Algorithm 6: Particle swarm optimisation pseudocode.

in computation time and a lack of sensitivity to parameters are also commented upon. Another application in the image and video analysis field includes biomedical image registration (Wachowiak et al., 2004), where eight slightly different variants of the PSO are tested against seven slightly different variants of ES. The efficacy of using PSO for image registration is noted, although the performance of ES is also good. Also in the image analysis field, PSO has been used as a solution to the inverse scattering problem arising in microwave imaging applications (Donelli and Massa, 2005), again outperforming a GA.

'Control' applications also form a large portion of PSO's applications, which includes the design of proportional-integral-derivative (PID) controllers (Gaing, 2004), where it outperforms a GA, control of power plants (Heo et al., 2006) and reactive power and voltage control (Yoshida et al., 2000), where PSO shows promising results.

The third largest application area for PSO is distribution networks. Specifically mentioned are the problems of network reconfiguration and expansion (Kannan et al., 2004), where PSO (and variants) outperforms a more traditional technique (dynamic programming) and economic dispatch (Gaing, 2003; Park et al., 2005; Selvakumar and Thanushkodi, 2007).

Note that this study only includes papers from the IEEE Xplore database, and so is limited in scope (including roughly 700 papers), but as an overview of the field the authors suggest this is adequate. They comment on the success of PSO in such a wide range of different application fields, owing to PSO's simplicity, ease of adaptation and capability of hybridisation. Finally, the authors comment that possibly the least successful field PSO has been applied to is combinatorial optimisation, with more work

needed to adapt the PSO for combinatorial optimisation problems over continuous optimisation.

A later survey (Sedighizadeh and Masehian, 2009) finds similar results, with the largest area of application categorised as 'electrical engineering.' This covers electricity generation and power systems (Del Valle et al., 2008), design and control of neural networks (Gudise and Venayagamoorthy, 2003) and, as found by the previous study, control applications. The area of data clustering (Van der Merwe and Engelbrecht, 2003) and data mining (Sousa et al., 2004) is found to be a much larger area by this study than the previous.

2.3 Summary

In this Chapter, the existing literature on six nature-inspired algorithms has been explored, looking at the origin, inspiration and applications of the algorithms used throughout this thesis. This provides a fundamental underpinning of the basics of each algorithm, which enables the use of these as the test algorithms for the remainder of this study. In the next Chapter, a methodology is proposed which uses a fitness landscape generator to further the understanding of the capabilities of these algorithms, by analysing their performance in relation to landscape characteristics.

Chapter 3

Methodology

3.1 Introduction

Inspired by the foundational work of Wolpert and Macready (1997), practitioners have long sought to better understand the relationship between problems and solution methods (i.e., algorithms). Here, the question of interest is "Which algorithm is *best-suited* to a particular problem?", and the process of addressing this has been described by some as a "black-art" (Woodward, 2010).

Although theoretical studies in this area have yielded useful results, the *experimental analysis* of algorithms is receiving increasing attention. As Morgan and Gallagher (2010) point out, this approach is *scalable* in that it readily admits newly-described algorithms, and it is now an area of research that is supported by a number of high-profile competitions and libraries of benchmark test problems.

The fundamental properties of a problem's *search landscape* underpin much work in experimental analysis, and the use of landscape/test case generators (Gallagher and Yuan, 2006; Jani, 2008; Morgan and Gallagher, 2010; Jin, 2004; Michalewicz et al., 2000) has been proposed as one way in which algorithm designers might effectively assess algorithms against problem instances.

This methodology is based on an *experimental* approach (Barr et al., 1995) to studying the selected algorithms, using an established landscape generation technique (Gallagher and Yuan, 2006). As Morgan and Gallagher observe, "In a general sense, an algorithm can be expected to perform well if the assumptions that it makes, either explicit or implicit, are well-matched to the properties of the search landscape or solution space of a given problem or set of problems" (Morgan and Gallagher, 2010). It is the objective, therefore, to investigate the performance of several algorithms on a number of types of *fitness landscape* with specific properties or characteristics. This approach is preferred by Hooker to the use of benchmark problems, because the latter "differ in so many respects that it is rarely evident why some are harder than others, and they may yet fail to vary over parameters that are key determinants of performance. It is better generate problems in a controlled fashion... The goal is not to generate realistic problems, which random generation cannot do, but to generate several problem sets, each of which is homogeneous with respect to characteristics that are likely to affect

performance" (Hooker, 1995).

In this Chapter, a generalised methodology is introduced which gathers a comparative dataset on algorithm performance. This methodology is used throughout the thesis, in Chapters 4 through 6, with some variation described in the individual Chapters. Firstly, a rationale to the approach is presented by exploring the background (and possible alternatives) to the techniques used in this methodology, before describing the facets of the methodology in detail in the remainder of the Chapter.

3.2 Background

3.2.1 Introduction to Fitness Landscapes

Evolutionary biology provides the fundamental foundations on which many concepts of optimisation arise, with the fundamental work of Sewall Wright providing the core details borrowed in optimisation today. The highly mathematical approach to evolutionary principles (Wright, 1931) led Wright to introduce the notion of what would eventually be called *fitness landscapes* (Wright, 1932). Wright suggests that by plotting every possible gene combination, a representation of the "fitter" gene combinations becomes viewable, as seen in Fig. 3.1. With this concept, the first instance of a "fitness landscape" exists - that is, a depiction of all possible gene combinations, represented in a way such that "fitter" combinations are shown as *peaks*, and "less fit" combinations are depicted as *valleys*. Wright goes on to describe the process of evolution as one of traversing the landscape in an attempt to climb to the highest peak, and neatly posits the problem of local optima by describing a species stuck at a "fit" peak, surrounded by valleys, while a fitter peak may lie unreachable in the landscape beyond. Wright then goes on to suggest methods in which species may overcome this to find these fitter peaks, although for the purposes of this work, the notion of a fitness landscape is the interesting point, although these concepts are used in the development of various evolution-based nature-inspired algorithms.

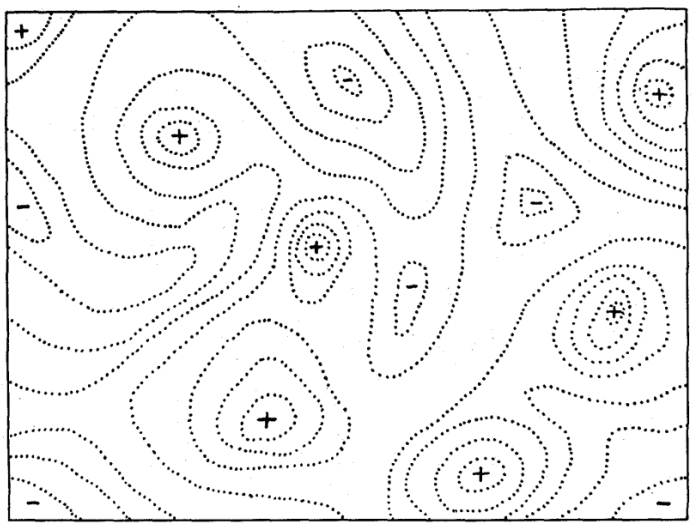


Figure 3.1: Wright's interpretation of a fitness landscape, as seen in Wright (1932).

To relate this concept to optimisation, consider that instead of representing each combination of gene combinations, and the fitness of a species, a fitness landscape is designed such that instead of gene combinations, each combination of *potential values for each variable* is used, and their fitness *as calculated by the objective function.* In this way, a fitness landscape is generated, identical in purpose to those used in evolutionary biology, and the same concept can be applied (that is to say, optimisation can be seen as the process of traversing this landscape in an attempt to find the highest peak or lowest valley) as a helpful way to both visualise an optimisation problem, and offer a useful perspective on solving optimisation problems.

3.2.2 No Free Lunch Theorem

The No Free Lunch (NFL) theorems, posed by Wolpert and Macready (1997), establish that for any given algorithm, better performance on one *class* of problem comes at the cost of decreased performance on another class of problem, and this has a variety of implications, discussed in this section.

There are two NFL theorems: The first states

$$\sum_f P(d_m^y|f, m, a_1) = \sum_f P(d_m^y|f, m, a_2). \tag{3.1}$$

Essentially, "... what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problems ..." - i.e. if an algorithm is better than random search at one class of problems, it must suffer (using the same measure of performance) in another class of problems, on other classes of problems, on average. Another way of viewing this is to consider that there can be *no one best algorithm* in any given instance, and this is a crucial concept. When algorithms are proposed, as discussed previously, a small sample of performance is demonstrated, and this does not offer robust insight into their performance across a range of classes to a practitioner; Indeed, classes are usually selected that show an algorithm in its best light. A practitioner may select an algorithm, based on some sample benchmark problems, that is wholly inappropriate for the class of problems they are trying to solve, and if an algorithm is truly novel, then there are no further performance details available for this algorithm.

The second NFL theorem states "... if one algorithm outperforms another for certain kinds of cost function dynamics, then the reverse must be true on the set of all other cost function dynamics." This theorem is interesting only for time-dependent situations, which are outside the scope of this study, but offers further insights into the differentiation between algorithms in terms of performance and the necessity of providing *complete* information about performance across a complete set of problem classes.

With the ideas presented by the NFL theorems in mind, it becomes increasingly clear that there is a need to focus on the problem of algorithm selection (Rice, 1976), as selecting the wrong algorithm for the problem to be solved could leave a practitioner

in a position where they are using an algorithm that offers poorer performance than a random search. A necessity for algorithm selection is to use any information possible from the problem to inform the algorithm selection process, and using information gained from the fitness landscape is one such way to do so.

3.2.3 Fitness Landscape Analysis

The fitness landscape approach has been successfully applied to the study of various nature-inspired algorithms. There exists a large body of work on fitness landscape analysis for *evolutionary* algorithms, particularly in the domain of discrete (or combinatorial) optimisation problems. Merz (2000) focusses on landscape *ruggedness* (defined, in this instance, in terms of the number of local optima, the distribution of these optima in the search space and the correlation between neighbouring points in the search space) in the context of a specific combinatorial optimization problem (Quadratic Assignment). As this approach does not use a landscape generation technique, the authors use a range of techniques for analysing fitness landscape hardness, such as the random walk correlation function (Weinberger, 1990) and the fitness distance correlation coefficient (Jones and Forrest, 1995), and relate these difficulty measures to actual algorithm performance. They find that combining several fitness landscape analysis techniques together can offer insights into the selection of evolutionary operators in memetic algorithms, and suggest that more efficient hardness measures would be beneficial in designing these operators.

Tavares et al. (2008) describe similar work, this time using the Multidimensional Knapsack problem as the case study. By using similar measures of fitness landscape hardness as in Merz (2000), the authors describe changes in performance obtained by altering components of an evolutionary algorithm (such as the problem representation). Their findings highlight the importance of selecting correct components of an evolutionary algorithm for a given problem, and the use of fitness landscape analysis is used mainly to explain performance *differences*.

It is possible to adapt these hardness measures for continuous function optimisation, as shown by Uludag and Sima Uyar (2009). Here, the fitness distance correlation coefficient and correlation length are adapted for continuous search spaces in an attempt to describe the hardness of a continuous fitness landscape and the behaviour of differential evolution algorithms. The authors determine that the new measures are capable of describing algorithm performance when the landscape does not contain sharp ridges, is not deceptive and is not unimodal with a large basin of attraction. They suggest that additional hardness measures are required to fully analyse all kinds of landscapes.

Malan and Engelbrecht (2009) propose a metric for ruggedness of continuous landscapes. Importantly for this work, the authors stress the importance of a *combined* approach, in which *multiple* characteristics of a landscape are considered. Using a random walk, an estimate of the number of local optima (i.e., the multimodality of the landscape) is made, an approach similar to that taken for discrete landscapes. While

this work focuses on the ruggedness of landscapes, it highlights that fact that, in terms of hardness measures, there still exist many unexplored characteristics of landscapes.

3.2.4 Fitness Landscape Generators

While it is interesting to look at the relationship between fitness landscape hardness and intrinsic hardness of the problem, another approach exists. Rather than analysing landscapes to determine a meaningful representation of difficulty, another approach uses randomly generated landscapes, which capture a set of pre-determined characteristics. By analysing the performance of algorithms on these randomly generated landscapes, the intention is to effectively 'reverse-engineer' the problem, associating algorithm performance with specific characteristics of landscapes rather than simply a 'hardness measure'.

The NK Model

Several techniques exist to generate fitness landscapes, with one of the most widely known being the NK model (Kauffman and Levin, 1987; Kauffman and Weinberger, 1989), a method for generating tunable NP-complete (Wright et al., 2000) combinatorial optimisation problems. A solution representation to a problem in the NK model is made up of a string (of length N), made up of values from a pre-defined set. The K value defines how many other characters in the string the fitness calculation relies on (that is to say, is the fitness calculation based on *substrings*). To give an example, an NK problem with an N of 3 and a K of 1 would have a fitness function which, in order to calculate the fitness for a given string, used the substrings (0, 1), (1, 2), (2, 3) and (3, 0). The NK model itself does not impose a specific fitness calculation, rather this is down to the specific implementation (and thus follows the distance measure chosen to create the associated fitness landscape) usually as a function, or lookup table, mapping each potential substring combination to a specified fitness value. The fitness of a complete string is then the total value of all its substrings.

A variant of the NK model, the NKp model (Barnett, 1998), offers a probability (p) that certain substring patterns make no contribution to the overall fitness of the string (i.e. a percentage of the substrings, proportional to p, have a fitness value of zero). Another variant, the NKq model (Newman and Engelhardt, 1998), defines a fixed number of "levels" (q) that the fitness values for each substring can take (i.e. if q is 2, substring fitness values will be polarised to 0 and 1, rather than the standard NK landscape where fitness values tend to be continuous in the range 0 to 1). A comparison of the effect of using NKp or NKq landscapes in place of NK landscapes on search is performed by Geard et al. (2002), who conclude that NK and NKq landscapes have similar search functionality, though NKq landscapes exhibit more "ridge-like" properties. NKp landscapes differ more significantly, however, due to large areas of neutrality (i.e. lack of gradient information).

Merz and Freisleben (1998) use the NK model to analyse the effectiveness of evo-

lutionary search strategies, in high dimensional problems. The findings stress the importance of using landscape generators like this; highlighting that previous studies have focused on small sets of test problems. Specifically, the results indicate that differences in performance of the algorithms studied are *not* highlighted in small dimensional problems (the sizes usually included in the previously used test sets) and only by using a landscape generation technique have they been able to study problems which emphasise the difference in performance of these algorithms.

Pelikan et al. (2009) also use a slightly modified version of the NK model as a method for analysing the performance of evolutionary algorithms, relating the generation parameters (n, k and, in their version of the model, *step*) directly to performance as a method of characterising the algorithms tested.

The Max-Set of Gaussians Method

A more recent technique, proposed by Gallagher and Yuan (2006), is the Max-Set of Gaussians landscape generator, as a potential technique for improving the experimental analysis of algorithms (Bartz-Beielstein, 2003). Differing from the NK Model, this technique generates *continuous* optimisation problems.

A number of Gaussians are generated, according to the n -dimensional Gaussian function

$$g(x) = [\frac{1}{(2\pi)^{\frac{\pi}{2}}|\sum|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\sum^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^T)]^{\frac{1}{n}} \tag{3.2}$$

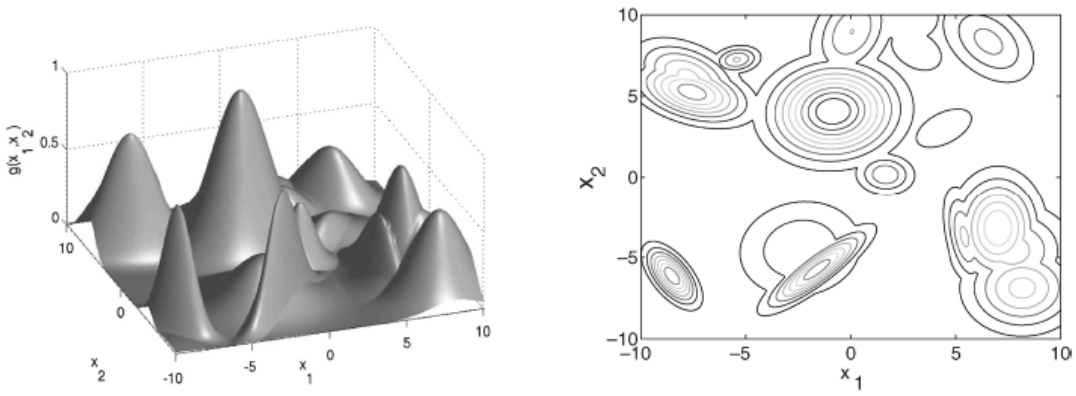
where $\boldsymbol{\mu}$ is an n-dimensional vector of means and $\sum$ is an $(n \times n)$ covariance matrix. A Gaussian function is, in this sense, described as "... an n-dimensional 'bump' or hill." Gallagher and Yuan then explain that a set of Gaussians can be combined as a weighted sum in a function, but for the purposes of generating a fitness landscape, the interest is not in the *sum*, rather the *maximum* Gaussian at any given point. Thus, the landscape is generated using the following:

$$G(x) = \max_i[w_i g_i(x)]. \tag{3.3}$$

where w is the *amplitude* of each hill - essentially, the scaling of each hill. A sample landscape generated using this technique is shown in Fig. 3.2.

Gallagher and Yuan (2006) go on to demonstrate that their technique is capable of generating landscapes of varying difficulty by tracking the behaviour of test algorithms across landscapes of varying designs, and observing differences in performance. Although they note that this is not a *thorough* comparative study, they suggest this signifies that the landscapes generated do provide different search patterns for the algorithms, and are therefore suitable for potential use as a test-bed when analysing algorithms.

This technique has been used successfully by Nannen et al. (2008) to examine the costs and benefits of tuning parameters for evolutionary algorithms. This work uses



(a) Surface plot. (b) Contour plot.

Figure 3.2: Example 2-D landscapes, shown in Gallagher and Yuan (2006), to illustrate the max-sum of Gaussians landscape generator.

the landscape generation technique to provide randomly generated problems, across four different problem sets, with varying degrees of structure, i.e. the structure of the landscape here is varied, rather than the specific characteristics used to generate the landscape, and results are taken as a whole, rather than analysed on a per-set basis. The effect of choosing different operators is analysed, alongside the interaction between different operators, and then the effect of tuning the parameters related to the various operators is also examined. Results show that the choice of operator for one component largely depends on the choice for other components, with the choice for selection having the biggest impact on performance, and tuning cost varied depending on the overall setup of the algorithm.

Morgan and Gallagher (2010) expand on the methodology, by incorporating a ridge structure generation process into the MSG technique. In doing so, a comparative experimental methodology is also proposed. Firstly, the technique for generating the ridge is illustrated, with an additional parameter added to the landscape generation technique - the angle of the ridge, and it is this parameter under investigation. The experimental methodology proposed is to generate a number of landscapes with fixed characteristics, barring that under investigation (angle of ridge), which varies between a range (in this case, 0 to 45 degrees, in increments of 5). The performance of two algorithms is trialled on a number of randomly generated landscapes at each of these landscape characteristic values, and plotted, to determine the effect varying this characteristic has on algorithm performance. It is noted that one of the benefits of using this technique is being able to recreate specific landscapes, allowing for examination of specific landscapes where results were unexpected.

This technique for generating landscapes forms the basis for the analytical method, with the parameters used to generate the landscapes as the cornerstones for the fitness landscape characteristics. A more in-depth review of these characteristics, including their effect on optimisation problems (Section 3.3.2) and on the various algorithms (Section 4.2) can be found in the following Sections.

3.3 Methodology Design

3.3.1 Algorithm selection

A number of nature-inspired algorithms that are commonly applied to continuous function optimisation are selected for comparison. These may be classified (Brabazon and O'Neill, 2006) as either social, evolutionary or physical:

- Social Systems
  - Bacterial Foraging Optimisation Algorithm (BFOA) (Passino (2002))
  - Bees Algorithm (BA) (Pham et al. (2006b))
  - Particle Swarm Optimisation (PSO) (Kennedy and Eberhart (1995))
- Evolutionary Computation
  - Genetic Algorithm (GA) (Goldberg (1989))
  - Evolution Strategies (ES) (Bäck and Schwefel (1993))
- Physical Systems
  - Harmony Search (HS) (Geem and Kim (2001))

Also included are Random Search (RS) and Stochastic Hill Climbing (SHC) as "baseline" algorithms.

Note that the references supplied above for each algorithm may serve simply as an example of their *application*, rather than their precise *implementation*. In terms of implementation, the observation that "Ideally, competing algorithms would be coded by the same expert programmer and run on the same test problems on the same computer configuration" (Barr et al., 1995) is observed. With that in mind, implementations provided by Brownlee to accompany Brownlee (2011) are used.

3.3.2 Optimisation problem characteristics

As Morgan and Gallagher (2010) explain, their Max-Set of Gaussians (MSG) method (Gallagher and Yuan, 2006) is a "randomised landscape generator that specifies test problems as a weighted sum of Gaussian functions. By specifying the number of Gaussians and the mean and covariance parameters for each component, a variety of test landscape instances can be generated. The topological properties of the landscapes are intuitively related to (and vary smoothly with) the parameters of the generator."

By manipulating these parameters, landscapes with different *characteristics* are obtained. This allows us to investigate the performance of the selected algorithms on landscapes with different features, and to identify which characteristics pose the greatest challenge. As Morgan and Gallagher observe, "Different problem types have their

own characteristics, however it is usually the case that complementary insights into algorithm behaviour result from conducting larger experimental studies using a variety of different problem types" Morgan and Gallagher (2010). The different characteristics (corresponding to problem types) under study are now described.

Ruggedness of a landscape is often linked to its difficulty (Jones and Forrest, 1995), and factors affecting this include (1) the *number* of local optima (Horn and Goldberg, 1994), and (2) *ratio* of local optima to the global optimum (Malan and Engelbrecht, 2009; Merz, 2000). Other significant factors concern (3) *dimensionality* (Hendtlass, 2009) (that is, the number of variables in the objective function), (4) *boundary constraints* (that is, the limits imposed on the value of a variable) Kukkonen and Lampinen (2005), and (5) *smoothness* of each curve making up the landscape (Beyer and Schwefel, 2002). Each of these characteristics are now described in more detail.

Number of local optima

This characteristic has long been considered one of the greatest challenges for optimization. If a landscape is made up of only one curve, the landscape is unimodal with no local optima. Each additional curve in the landscape introduces a local optimum, creating landscapes with increasing modality. The range chosen for number of local optima begins at one (starting at a unimodal landscape) and increases to ten, with three local optima as the default value when investigating other characteristics.

Ratio of local optima to global optimum

As the ratio approaches one, local optima become increasingly "attractive" to optimisation algorithms. A full range of values is investigated for this characteristic, from 0.1 (ignoring zero, as this would completely remove local optima) to 0.9 (ignoring 1, as this would make the local optima effectively equal to the global optimum). Preliminary experiments suggested an appropriate step size of 0.2, balancing information loss against computation time. A default value of 0.5 was used when other characteristics were investigated.

Dimensionality

The so-called "curse of dimensionality" (Bellman, 1972; Michalewicz and Janikow, 1992) refers to the challenge posed by a search space that expands in multiple dimensions (i.e., that grows exponentially with the addition of extra variables). Preliminary experiments suggested an upper bound of ten dimensions for effective comparison, after which performance degraded beyond the point of usefulness for most of the algorithms under test. Landscapes with dimensionality in the range 1–10 are tested, with a default value of 2.

Table 3.1: A summary of the ranges selected for the characteristics in the fitness landscapes.

Characteristic	**Min**	**Step**	**Max**	**Default**
Number of local optima	0	1	9	3
Ratio of local optima to global optimum	0.1	0.2	0.9	0.5
Dimensionality	1	1	10	2
Boundary constraint range	10	10	100	30
Smoothness Coefficient	10	10	100	15

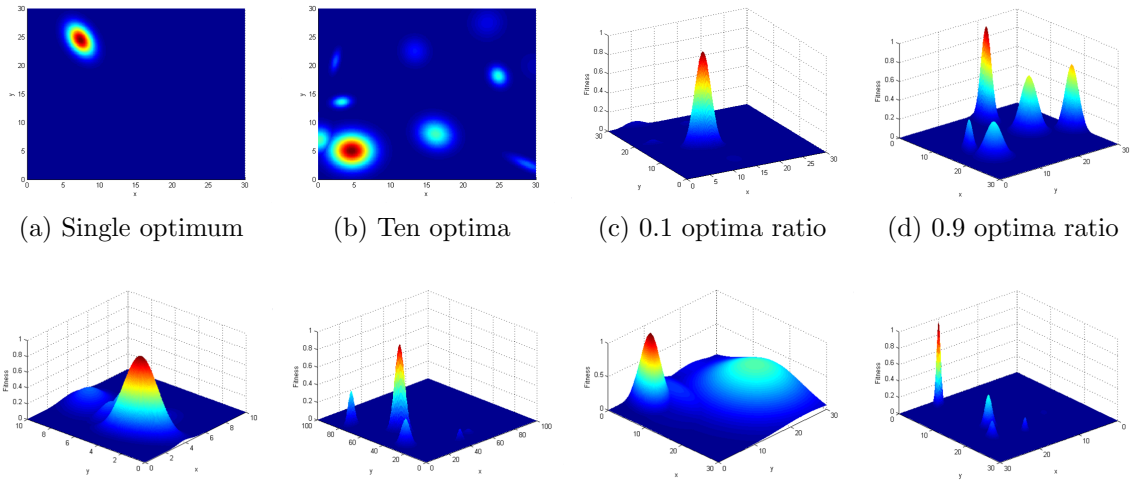
Boundary constraints

Boundary constraints define the range of each variable in an objective function. The way in which algorithms handle these boundary constraints can have a great impact on performance (Kukkonen and Lampinen, 2005). Constraints are often closely linked to a specific parameter in most algorithms, and for this reason there is only a certain amount of adjustment is it possible to make before algorithms start to struggle with no varied parameterisation. For this reason, a relatively small range of ten units to one hundred units in each dimension is chosen, increasing in steps of ten, and thirty is used as the default value.

Smoothness

"Smoothness" is defined as a coefficient controlling the gradation of each curve making up a landscape (Beyer and Schwefel, 2002). As this coefficient increases, curves become steeper, generating a landscape with larger "barren" areas (with no useful gradient information) and a much steeper slope for each optimum. Preliminary experiments showed little change in general algorithm performance with changes in the smoothness coefficient, so a broad range was chosen, with a lower limit of 10 and upper limit of 100, increasing the coefficient in steps of 10. For the default, a coefficient of 15 is used, which provides "interesting" landscapes with good coverage of gradient information, whilst not disadvantaging gradient-reliant algorithms too much when the smoothness characteristic is not under consideration. This characteristic is related to ruggedness by reference to the distribution of optima. While the MSG method offers no direct control over the precise *placement* of local optima, the smoothness of the curves is directly related to the amount of *space* these curves occupy on the landscape. With a small smoothness coefficient the curves occupy a larger proportion of the fitness landscape, in turn providing more gradient information around the optima.

A summary of the ranges selected for each characteristic is given in Table 3.1. Sample landscapes generated at the extremes of each range are also shown in Fig. 3.3.



(a) Single optimum (b) Ten optima (c) 0.1 optima ratio (d) 0.9 optima ratio

(e) Range =10 (f) Range =100 (g) Smoothness coefficient=10 (h) Smoothness coefficient=100

Figure 3.3: Illustrations of sample landscapes generated at the extremes of each characteristic range (from top: number of local optima, local optima ratio, boundary constraints, smoothness (dimensionality=2 in all cases) . Captions describe the parameter that was altered, all other parameters set to default values.

3.3.3 Performance measurement

In terms of *performance metrics*, algorithm-specific measures have been abstracted away from, due to the diverse range of methods selected. The following metrics are applied:

Accuracy

Defined as the average distance from the global optimum of the best solution found on a given set of landscape characteristics, over a number of runs (that is, the average error). This is the most commonly-used assessment metric for optimisation algorithms (Gallagher and Yuan, 2006) . The generation technique used creates landscapes with a known global optimum (zero), which permits a precise assessment of an algorithm's performance.

Variation of final solutions

A measure of variation in best solutions found across differently seeded runs. The standard deviation of the best solutions of all runs on a given set of landscape characteristics is used, defined as $(\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{\frac{1}{2}}$ (where X is the data set, n is the size of the data set and $\bar{x}$ is the mean average).

Success rate

Defined as the frequency with which differently-seeded runs of an algorithm are able to find a solution within a specified distance from the optimum (Elbeltagi et al., 2005). The success tolerance is chosen to be relatively low (error less than 1.0×10^{-4}) in order

to ensure that changes in success rate of algorithms which perform poorly are fully encapsulated by the results.

3.3.4 Experimental setup

In order to generate the landscapes, the Matlab code supplied with Gallagher and Yuan (2006) is used. All landscapes were generated using default parameters of three curves, two dimensions, 0.5 average ratio of local minima to global minimum, 30 units in each dimension with a smoothness coefficient of 15), with only the parameter under investigation changing for each experiment. Each algorithm was executed 100 times on each landscape in the set of landscapes generated for each particular characteristic value (when investigating smoothness, for example, 1000 landscapes were generated, 100 different landscapes for each parameter configuration (smoothness = $10 \ldots 100$), and each algorithm executed 100 times on each landscape). The Ruby implementations of each algorithm were taken from the repository associated with Brownlee (2011) - code was added to each one to track the number of objective function evaluations, and - where necessary - code was slightly modified for continuous optimization.

Parameterisation of algorithms provides a significant challenge when evaluating performance. The aim is not to perform "competitive testing" Hooker (1995), but to establish general performance *profiles* for different algorithms over different types of problem. As such, the so-called "vanilla" implementation of each algorithm was used, with general-purpose settings. For parameters that are unique to a specific algorithm, the default values specified in the codebase were used (see Table 3.2 for a full list of parameters used).

Termination criteria were also standardised. The most objective criterion is the number of objective function evaluations. This means each algorithm has access to the same amount of information from the landscape, and the same amount of *feedback* on potential solutions. Experimentally, it was determined that the selected algorithms generally converged within 20,000 objective function calculations, so this was used as the termination criterion.

The code used for all algorithms, as well as datasets and the landscape generator, is available on request from the authors.

3.4 Summary

The methodology proposed in this Chapter allows for the gathering of performance data on algorithms that is non-specific to a particular benchmark problem and instead highlights the strengths and weaknesses of an algorithm with relation to characteristics of a problem. Many facets of the methodology can be changed to accommodate particular questions; The algorithms are by no-means fixed and are not restricted to nature-inspired optimisation algorithms, the set of characteristics (and indeed, the problem generator) can be swapped out for another, and the measures used to assess

Table 3.2: Parameter values for the selection of algorithms.

Harmony Search		**Particle Swarm Optimisation**		**Evolution Strategies**	
Range	10	Maximum velocity	10	Population size	50
Memory size	50	Population size	50	Number of children	20
Consideration rate	0.95	Personal best weight	2	Strategy mutation	Off
Adjustment rate	0.7	Global best weight	2		
Bees Algorithm		**Bacterial Foraging Optimisation Algorithm**		**Genetic Algorithm**	
Number of bees	50	Step size	0.1	Population Size	50
Patch size	10	Population size	50	Bits per variable	16
Number of sites	3	Swim length	3	Crossover probability	0.95%
Number of elite sites	1	Elimination chance	0.25%	Mutation probability	$\frac{1}{totalbits}$
Number of elite bees	7	Attractant depth	0.1		
Number of other bees	2	Attractant width	0.2	**Stochastic Hill-Climbing**	
		Repellent height	0.1	Range	10
		Repellent width	10		
		Chemotactic steps	50		
		Reproduction steps	2		

the algorithms can also be reconsidered.

To assess the potential usefulness of the methodology as depicted here, the remainder of this thesis explores the three research questions posed at the outset, using the methodology. In Chapter 4, the viability of the method for collecting performance profiles of an algorithm is assessed. In Chapter 5, the methodology is used as a way of establishing whether there is a meaningful performance improvement in using automated parameter tuning techniques for the various algorithms and, finally, in Chapter 6, the methodology is used to generate datasets for learning algorithms as a possible technique for predicting algorithm performance.

Chapter 4

Performance Analysis Using Fitness Landscape Characteristics

4.1 Introduction

In the previous Chapter, a methodology was proposed which allows for the gathering of performance data of algorithms based on fitness landscape characteristics, as opposed to specific benchmark problems. The first question posed by this thesis is: "To what extent can fitness landscape characteristics be used to establish a performance profile of an algorithm, and thus distinguish between different algorithms in terms of performance?" and, therefore, this Chapter explores whether the methodology proposed is suitable for generating algorithm performance profiles for the set of algorithms selected for study.

The remainder of this Chapter is organised as follows: in Section 4.2 there is an examination of the properties of the selected algorithms, which suggests likely algorithms of the analytical methodology proposed in Chapter 3.3. The methodology is then used to obtain performance profiles, which are presented in Section 4.3, which are summarised with a concluding discussion in Section 4.4.

4.2 Expected Results

In work exploring the practicality of using optimisation algorithms for given problems there is speculation and assertion regarding algorithm performance on specific problems. In this section, an analysis of this literature assesses which algorithms are likely to perform well under certain combinations of characteristics.

Increasing the size of the problem space provides a more difficult fitness landscape. Specifically in the case of these randomly generated landscapes, the problems provided are two-fold: The space is larger, thus requiring more exploration to find an optimal solution, and similarly, a larger area of the problem is 'flat', providing less information for algorithms which rely on the gradation of the landscape to optimise.

The algorithms included in this study have different 'coping mechanisms' for the

increase in size of problem spaces, which will now be briefly discussed.

One strategy for ensuring that an algorithm is searching for *promising* solutions is to produce new solutions from within a smaller range of the current best solution - most commonly referred to in algorithmic terms as a *maximum step size* (expressed as a distance - in the case of continuous optimisation, this is usually Euclidean distance). It is important to note that in the algorithms selected, this is always considered as a **maximum**, and not an absolute value. This ensures that solutions can be chosen from anywhere within an infinitesimally small distance from the current best solution, up to the maximum step size. Algorithms which employ this technique are the Bees Algorithm (BA) (referred to as patch size), Particle Swarm Optimisation (PSO) (referred to as maximum velocity), Harmony Search (HS) (referred to as range) and Bacterial Foraging Optimisation Algorithm (BFOA) (referred to as step size). The advantages (and disadvantages) of including a step size parameter in each algorithm are discussed in the proposal of each algorithm, with references found in Chapter 2.

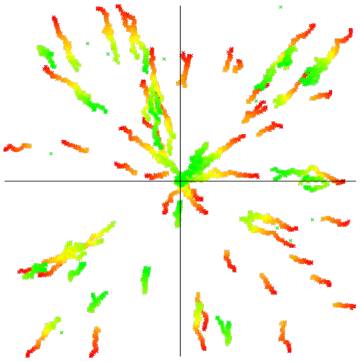
The obvious downside of limiting new solutions to a certain area within the fitness landscape is that the current best solution may not be in an area where the *actual* best solution is (i.e. the algorithm may be trapped in a local optima). Some of the algorithms under investigation in this study have additional coping mechanisms to counter this - with the added benefit that, if the step size parameter is *inappropriate* for a given problem space size, the algorithm is not hindered too much. The BA performs a global search at each step of the algorithm (described as 'scout bees' - essentially, random solutions from *anywhere* within the fitness landscape are included in the new population, in addition to those from promising regions) and this should allow the BA to cope well with varying boundary constraint ranges while the step size parameter remains constant (Pham and Castellani, 2009). The BFOA has elimination-dispersal events (in other words, solutions can be randomly redistributed throughout the fitness landscape at periodic intervals), which attempts to ensure that the fitness landscape is fully explored regardless of both local optima and also inappropriate step sizes being chosen (Chen et al., 2008).

Other algorithms under study here draw from the fitness landscape without restriction, or by adjusting the current solution (without regarding topological information such as distance). Of particular note, the Genetic Algorithm (GA) in this study functions differently from the other algorithms as it is binary-coded, rather than real-coded. The parameter in the GA that controls the *granularity* of exploration is the number of bits used in the encoding. A larger number of bits allows for a fine-grained exploration of the fitness landscape, while a smaller number of bits means less possible solutions can be found (and, therefore, less precise solutions - and less gradient information can be obtained). If the parameters are kept constant as problem space size increases (as they are in this Chapter), the increase in problem space suggests that the GA (or any other binary-coded algorithm) can obtain less information, and provide less precise solutions, and may be hampered severely. This is one of the main problems with using binary-encoded solutions for continuous optimisation (Goldberg and Holland, 1988).

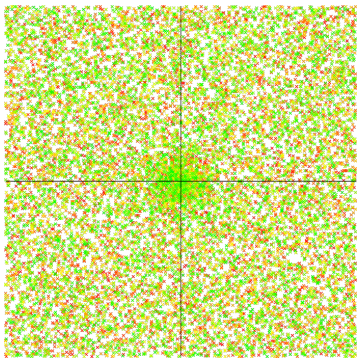
Increased dimensionality offers the greatest challenge for optimisation algorithms as each additional dimension increases the search space exponentially. For this reason, it is expected that all algorithms show greatly decreasing performance as the number of dimensions increases, with algorithms which can explore a large space quickly (i.e. algorithms that generate large pools of solutions) offering the best resilience towards increased dimensionality. BFOA is dubbed as an algorithm in which the performance heavily decreases as the search space grows (Chen et al., 2008), and as such it is expected that algorithms exhibit poor performance as dimensionality increases. Algorithms which can explore a search space efficiently and quickly are expected to cope best with the increase in dimensionality. Such algorithms include BA which has a constant global search alongside a local search (Pham and Castellani, 2009). PSO also claims to rapidly explore a search space and should therefore be able to cope with the increase in dimensionality (Eberhart and Shi, 2001).

It is expected that algorithms which are more robust, that is, less sensitive to parameter tuning, should be able to cope with the increased size in each dimension. HS has few parameters to adjust, with some versions of the algorithm not offering any parameterisation at all (Wang and Huang, 2010). Offering a high level of diversification, a wide and efficient search and refinement of local solutions (Yang, 2009), it is expected that the HS may perform well as search space size increases. BA has many parameters, but according to Pham's work after the initial publication, the algorithm is very robust - the algorithm is insensitive to parameter changes (Pham and Castellani, 2009). Given this, it is expected that the BA may also perform well. Each individual in the PSO algorithm is described with a "... tendency to hurtle past their target." Overshooting a destination will promote excellent exploratory search within the algorithm, and as such PSO is expected to perform well at larger problem sizes (Kennedy and Eberhart, 1995). It is possible that smaller problem sizes may prove difficult for PSO - if such overshooting extends the potential solution beyond the bounds of the problem, the exploration may become 'stuck' around the edges of the problem space.

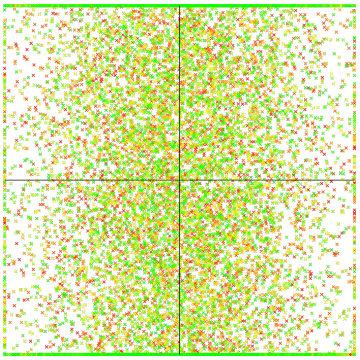
Fig. 4.1 shows the exploration pattern of the selected algorithms attempting to optimise a simple benchmark function (a two dimensional paraboloid given by $f(x, y) = x^2 + y^2$ with boundaries of -15 to 15). Algorithms were executed for 20,000 objective function calculations, as in the full methodology, and explore a 2-dimensional space of size 30, based on the default parameters used when generating landscapes. This gives some additional insight into the exploratory nature of the varying algorithms. In order to quantify the coverage of the fitness landscape, a grid (of 0.3 units square) was imposed, and the percentage of grid spaces visited measured. The results of this can be seen in Table 4.1. PSO and HS both have parameters which controls the 'range' of new solutions and this is visible from the plots - this parameter defaults (in this case) to 10, and you can see a square of heavy exploration around the centre of the landscape. BA, Evolution Strategies (ES) and Random Search (RS) are the top three algorithms in terms of coverage all showing above 60% coverage of the landscape. While this is suitable for a landscape of this size, they may show some problems as the landscapes



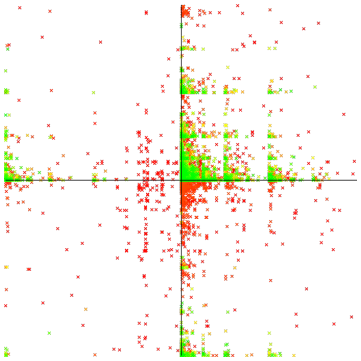
(a) Bacterial foraging optimisation algorithm.



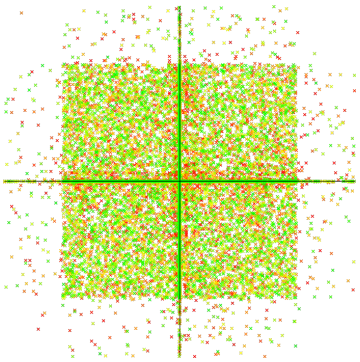
(b) Bee algorithm.



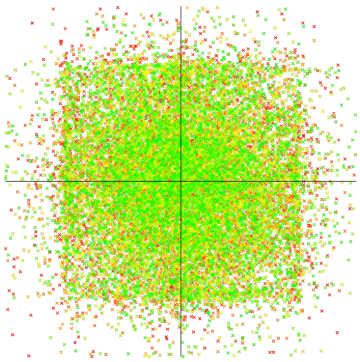
(c) Evolution strategy.



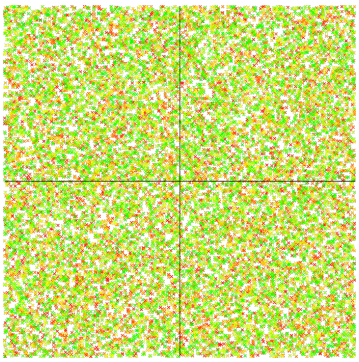
(d) Genetic algorithm.



(e) Harmony search.



(f) Particle swarm optimisation.



(g) Random search.

Figure 4.1: Exploration pattern of algorithms, executed for 20,000 objective calculations, optimising a two dimensional paraboloid given by $f(x, y) = x^2 + y^2$ in the range -15 to 15. Each cross indicates an area of the landscape that was searched, coloured from red (first objective calculation) to green (20,000th objective calculation). The global optimum is at (0, 0).

Table 4.1: A 0.3 unit square grid was imposed on the fitness landscape, and the percentage of grid spaces explored calculated as an estimate of the fitness landscape covered in the algorithm's exploration pattern. The results for both the parabolic function (Fig. 4.1) and Rastrigin's function (Fig. 4.2).

Algorithm	Percentage of Parabolic Function Landscape Explored	Percentage of Rastrigin's Function Landscape Explored
BA	78.74	79.52
BFOA	13.78	2.91
ES	61.36	61.61
GA	9.7	9.43
HS	46.7	47.61
PSO	55	57.04
RS	86.31	86.31

increase in dimensionality and search space size. GA focuses on areas of promise rather than providing a 'general overview' of the landscape, which offers good performance on a landscape with little to no multimodality. BFOA shows a very clear search pattern 'honing in' on the global minimum (at the origin), but movement in the algorithm is quite small and exploration levels are quite low just under 14%).

Another of the main concerns within optimisation algorithm design is that of getting 'stuck' in a local optimum. By altering the fitness landscape parameter which controls the number of curves in the landscape, increasing numbers of local optima are introduced to the fitness landscape and algorithm performance can be observed. BA boasts strong avoidance of local optima, owing to the abandonment of nest sites and rapid re-deployment of bees when the area no longer seems promising (Pham et al., 2006b; Pham and Castellani, 2009). PSO, another algorithm from the swarm-based category, is also recommended for problems with a high level of multimodality and is therefore expected to cope well with the increase in local minima (Poli et al., 2007). BFOA, however, relies very heavily on gradient information which is dubbed as both a benefit and a hindrance. For the purpose of avoiding local optima, the reliance on fitness landscape gradient means poor performance as the number of local minima increases is to be expected (Passino, 2002). This is in direct contrast with the expectations of the other swarm-based algorithms. HS is described as using little information about the gradient of the landscape contributing towards the generation of new solutions, and is therefore expected to perform well (Lee and Geem, 2005).

Adjusting the average ratio of local optima closer to the global optimum has two effects on algorithms which use the gradient of the landscape to optimise a solution. Firstly, it makes the local optima seem more attractive. This may make them more difficult to escape from as they may not discover a better solution while searching the immediate area unless they stumble across an area very close to the global optimum. Secondly, it means optimisation may succeed despite becoming trapped in a local

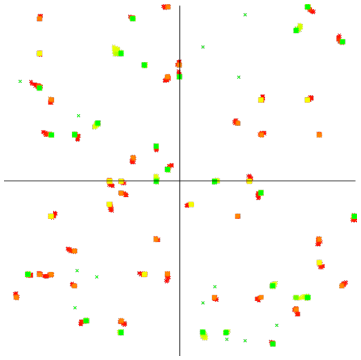
optima, as the local optimum may be very close to the global optimum. This behaviour would be quite unexpected, however, as even at a ratio of 0.9 (the upper limit on the range of values chosen for this parameter), the optimal value of a landscape would be 0.1, compared to the global minimum of 0 - this means, even successfully optimising accurately on a local minima, the error should be outside of the success criteria. Results when adjusting the ratio may be similar to those when increasing the number of local optima - that is to say, algorithms which do not rely on landscape gradation to generate new solutions (such as HS (Lee and Geem, 2005), BA (Pham and Castellani, 2009) and PSO (Eberhart and Shi, 2001)) should cope well, contrasting algorithms which rely on gradient information (such as BFOA (Passino, 2002)) to perform poorly.

Fig. 4.2 shows the exploration pattern of the selected algorithms on a benchmark function with some multimodality (Rastrigin's function). As with the exploration of the sphere equation discussed earlier, algorithms were executed for 20,000 objective calculations on a 2-dimensional space with a search size of 30 (selected to better parallel the full experiments, as opposed to the usual 10.24 search space size commonly used in optimising Rastrigin's function). Again, a grid was imposed, and visits to the cells of the grid were measured as a means of quantifying the coverage of the landscape. Results are presented in Table 4.1. There is little change between the exploration pattern of the sphere equation for BA, ES and PSO suggesting these algorithms are capable of escaping local optima. HS shows some failed attempts at exploring areas of interest (represented by the cross-like exploration pattern not around the origin), and GA also shows extra exploration around local optima. BFOA suffers greatly here, with a drastically different search pattern to that in the function with no multimodality, suggesting that the heavy reliance of gradient information hampers the algorithm's ability to escape local minima severely.

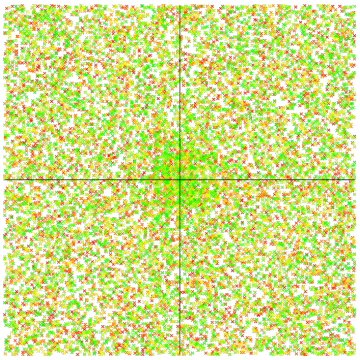
As a combination of both problem space size and ratio, curve gradation provides larger areas of 'barren' landscape (that is, landscape that provides no gradation), potentially offering difficulty to algorithms which rely on the gradation of a landscape to find an optimal solution, as discussed above.

4.3 Results

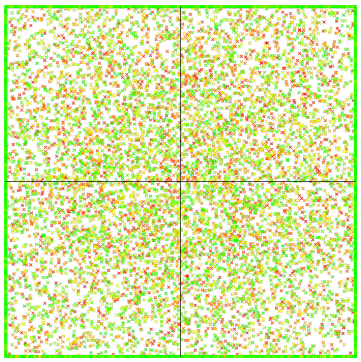
In this Section, results are explored on a per-characteristic basis. The presentation of the results takes the form of graphs depicting the mean error (accuracy), standard deviation (stability) and success rate of each algorithm as the characteristic in question's value changes. Also provided is a visual presentation of the p-values obtained using unpaired two-tailed t-tests, to assess the significance of the characteristic value's change on the algorithm's performance. Numerical values for the p-values are also presented in Appendix A.



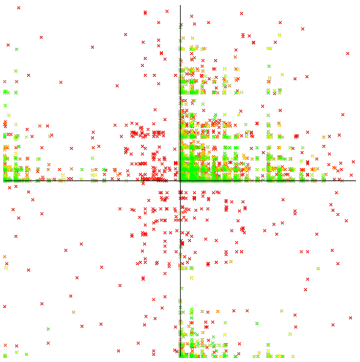
(a) Bacterial foraging optimisation algorithm.



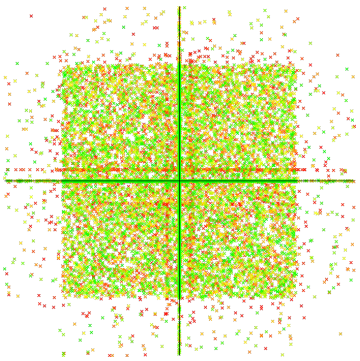
(b) Bee algorithm.



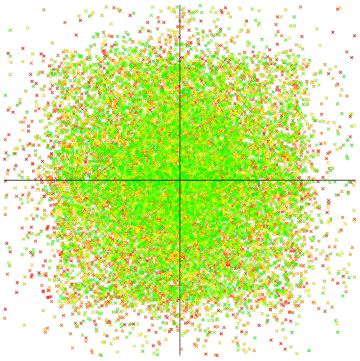
(c) Evolution strategy.



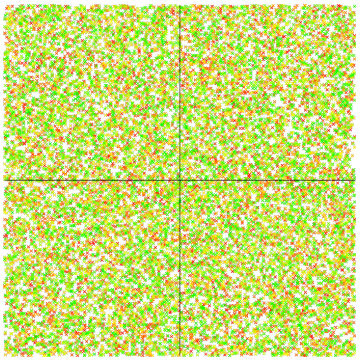
(d) Genetic algorithm.



(e) Harmony Search.



(f) Particle swarm optimisation.



(g) Random search.

Figure 4.2: Exploration pattern of algorithms, executed for 20,000 objective calculations, optimising Rastrigin's function in two dimensions ($f(x, y) = 10 \times 2 + [x^2 - 10cos(2\pi x)] + [y^2 - 10cos(2\pi y)]$) in the range -15 to 15. Each cross indicates an area of the landscape that was searched, coloured from red (first objective calculation) to green (20 000th objective calculation).

4.3.1 Number of local optima

Graphical results are depicted in Fig. 4.3. All algorithms produce the smallest average error when no local optima (minima) are present in the fitness landscape. This is expected, as, with only one optimum, there are no alternative solutions to which the algorithms may converge. The greatest average error occurs with only one optimum from Stochastic Hill Climbing (SHC), with BFOA (approx. 0.14) falling short also. There are very small average errors (almost zero) from GA, ES, PSO, HS, RS and BA. BFOA also produces the largest variation in final solutions (0.32).

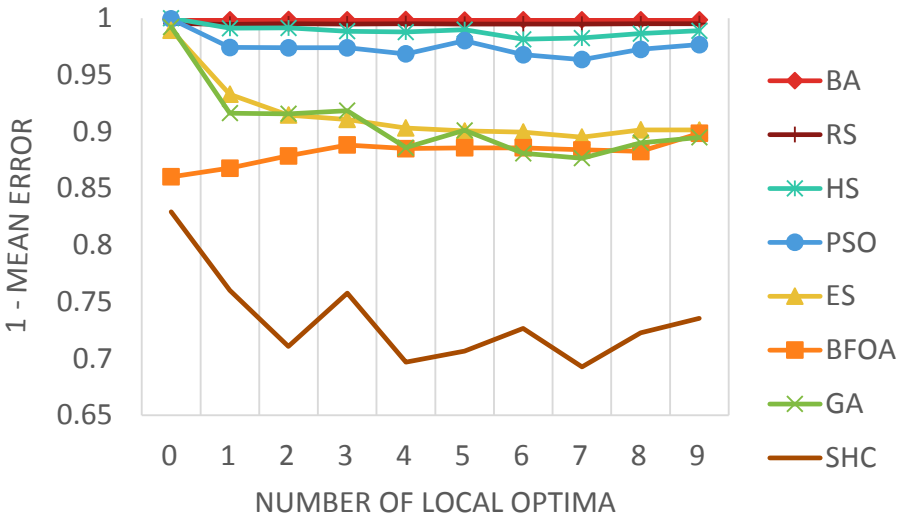
With the introduction of only a single local optimum, performance of *most* algorithms degrades significantly. ES and GA suffer significantly, with average error increasing from approximately zero to 0.07 and 0.09 respectively, and the standard deviation of solutions increasing by around 0.15 for each algorithm. SHC also performs poorly, with a similar increase in average error. The least affected are RS (which blindly chooses random solutions, and is therefore unaffected by local minima) and BA which contains a global search mechanism.

As the number of local optima increases beyond one, ES and GA continue to suffer from larger average errors. PSO and HS only suffer from slightly decreased performance as the number of local optima increases beyond one, while RS continues to suffer no performance decrease. Similarly, BA continues to show no performance degradation as the number of local minima increases, most likely owing to the nature of the global search they both perform. The swarm-based algorithms, therefore, seem to handle the problem of local optima most effectively.

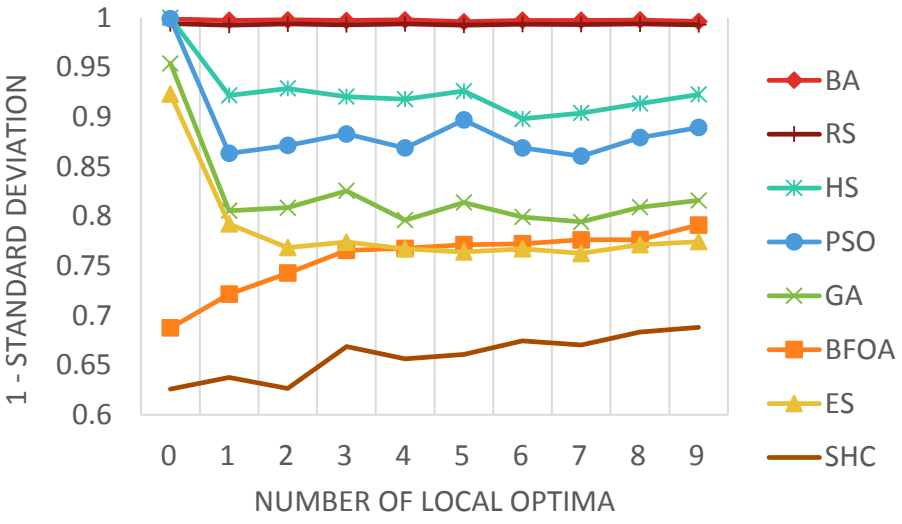
BFOA demonstrates a decreased average error as the number of local optima increases, most likely due to an increased likelihood of *latching on* to a gradient which leads towards any optimum at all, rather than becoming "stuck" in areas of the landscape with no information.

Statistical analysis of the results as the number of local optima changes (illustrated in Fig. 4.4) shows a varied profile across the selected algorithms. There is no statistical significance when comparing results for any number of local optima between three and eight inclusive for BFOA, suggesting that within this range adjusting the number of local optima has no significant effect. Similarly, increasing the number of local optima beyond five has no significant effect on ES. This profile is not reflected in BA, GA, HS, PSO, SHC or RS - all of which exhibit a mix of unique patterns.

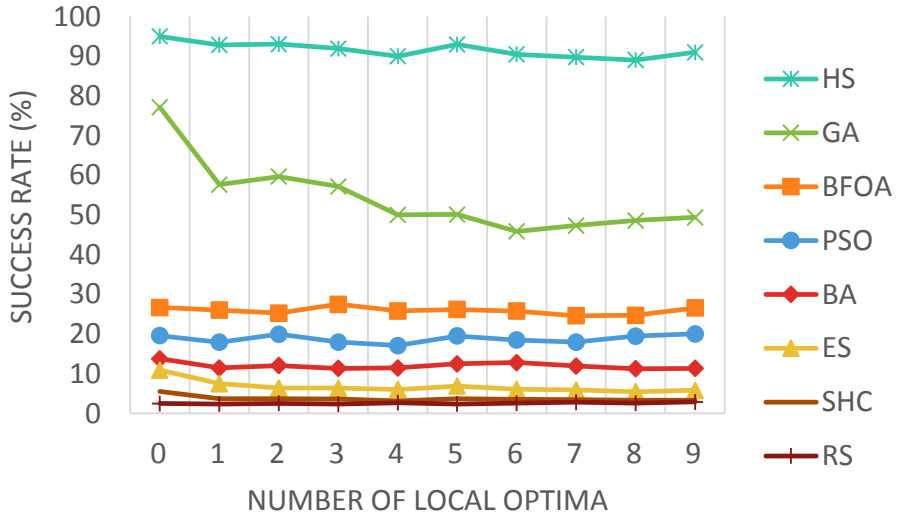
From the statistical analysis, it can be seen that in many cases adjusting the number of local optima does have an effect on the performance of the algorithm (even if that effect could be considered negligible in terms of actual performance), and that for all algorithms, the profile is varied - particularly BFOA and ES for which there exist ranges of local optima values where there is no statistically significant performance variation.



(a) Accuracy.

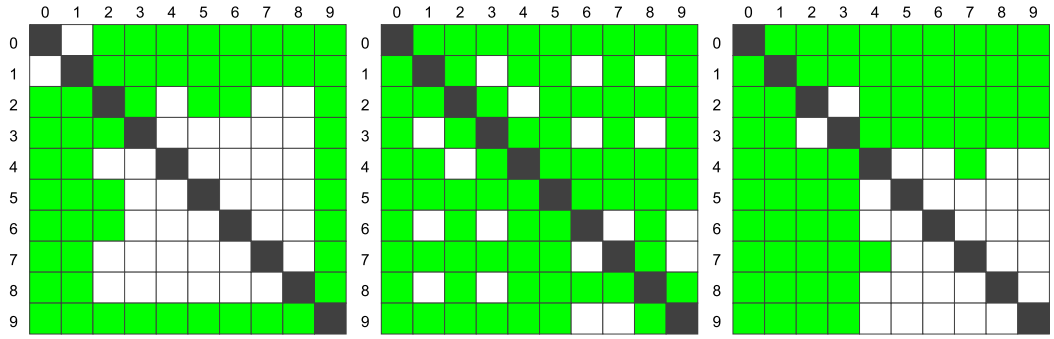


(b) Diversity.



(c) Success rate.

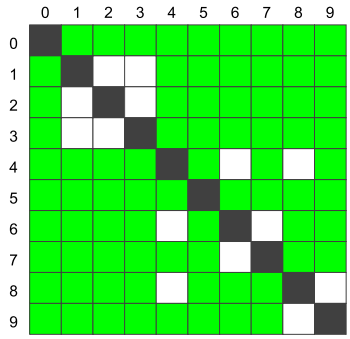
Figure 4.3: Result of varying the number of local optima.



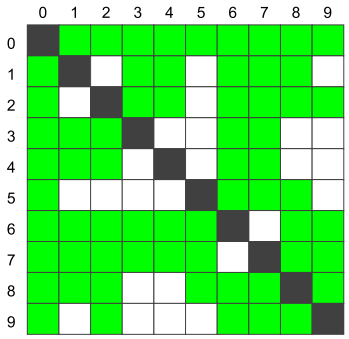
(a) Bacterial foraging optimisation algorithm.

(b) Bees algorithm.

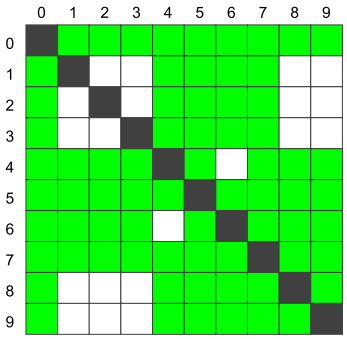
(c) Evolution strategy.



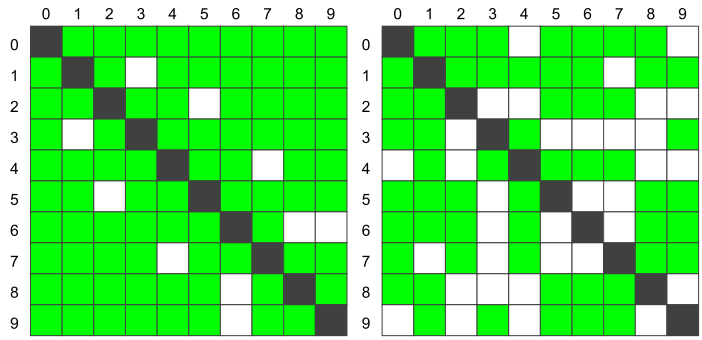
(d) Genetic algorithm.



(e) Harmony search.



(f) Particle swarm optimisation.



(g) Stochastic hill-climbing.

(h) Random search.

Figure 4.4: The results of unpaired two-tailed t-tests for each algorithm comparing algorithm performance results at one value of number of local optima with every other value of number of local optima (H_0 = The number of local optima in the fitness landscape has no effect on an algorithm's performance, H_A = The number of local optima in the fitness landscape has an effect on an algorithm's performance). Green shaded cells depict p-values below 0.05. Where a cell is shaded, a significant difference has been found between the results with one number of local optima (row) versus the second number of local optima (column). An algorithm with more shaded cells indicates it is more sensitive to changes in the number of local optima (i.e. there are more statistically significant performance differences as the number of local optima changes) than one with less shaded cells.

4.3.2 Ratio of local optima to global optimum

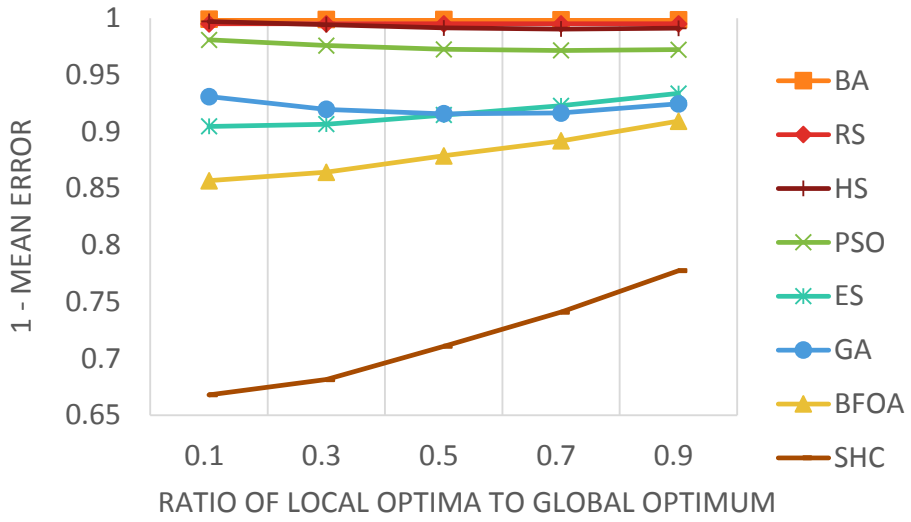
Results are depicted in Fig. 4.5. For algorithms which do not directly use the gradient of the landscape, no change in performance is expected as the ratio parameter is adjusted. Note that, for example RS, which selects new solutions randomly from the entire search space, offers very similar performance in terms of mean error and success rate for all ratio values. Similarly, algorithms which perform a global search should be better at avoiding local minima even when they are attractive - and this is true for BA and HS. PSO shows little change in success rate as the ratio becomes more attractive, owing to the fact that solutions are directed towards the best particle, and their own best solution, regardless of their individual experience with the gradation of the landscape. Interestingly, SHC average error decreases as ratio increases - most likely due to an increased availability of 'better' solutions throughout the landscape.

ES demonstrates very poor, yet consistent, performance as the ratio changes. Success rates are very low, and, interestingly, there is a decrease in the standard deviation of solutions as the ratio increases. This suggests that ES is perhaps more "content" to optimise at a local minima, with the algorithm getting trapped in these more frequently as ratio increases. This could also be true of other algorithms whose deviation decrease, such as BFOA and SHC. GA performs in a similar manner to ES with regard to average error and diversity, although with a considerably better success rate, suggesting that this may be a general problem for algorithms which use an evolutionary approach.

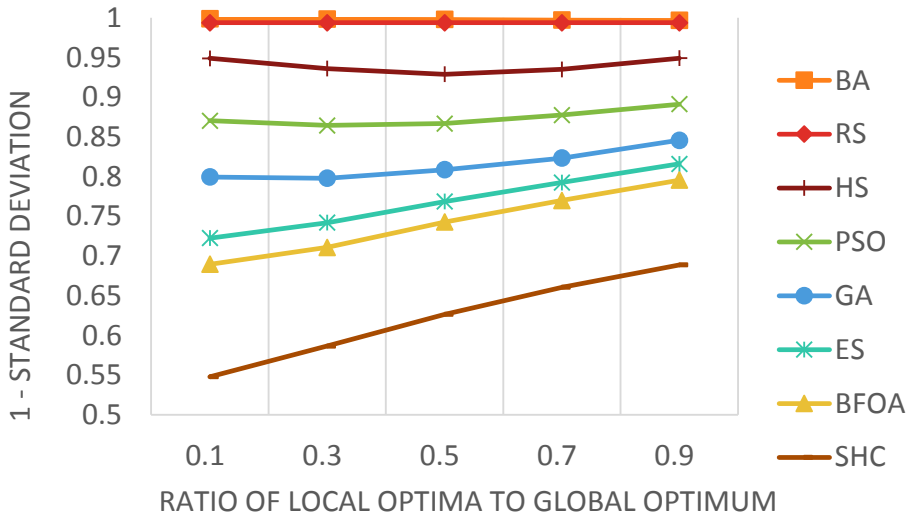
HS seems quite unhindered by the attractiveness of local minima, showing little to no change in performance as the ratio changes, with an interesting dip at 0.5 ratio. This suggests that this algorithm is quite capable of escaping local minima, with the dip being too small a change in average error to draw further conclusions.

Statistical analysis of the results as the ratio of local optima to global optimum changes (illustrated in Fig. 4.6) shows a varied profile of robustness across the algorithms. Most notably, RS's lack of performance change is confirmed as it shows no significant differences for any comparisons - the ratio of local optima to global optimum has no significant effect on RS at all, understandable as the selection of random solutions is unlikely to be affected by the availability of gradient information. SHC, which relies wholly on gradients, shows significant differences at each comparison - it, along with BA - are algorithm that are seemingly the most affected by the attractiveness of local optima. The BFOA and ES demonstrate the same profile in terms of ratio - they are unaffected by unattractive ratios of local optima (no significant performance change between 0.1/0.3), but there are significant performance changes when the ratio is higher. The GA is affected by changes to ratio only when there is an extreme difference. HS and PSO have a similar profile to each other in that they are affected by changes when the ratio is small, but not when the ratio is large.

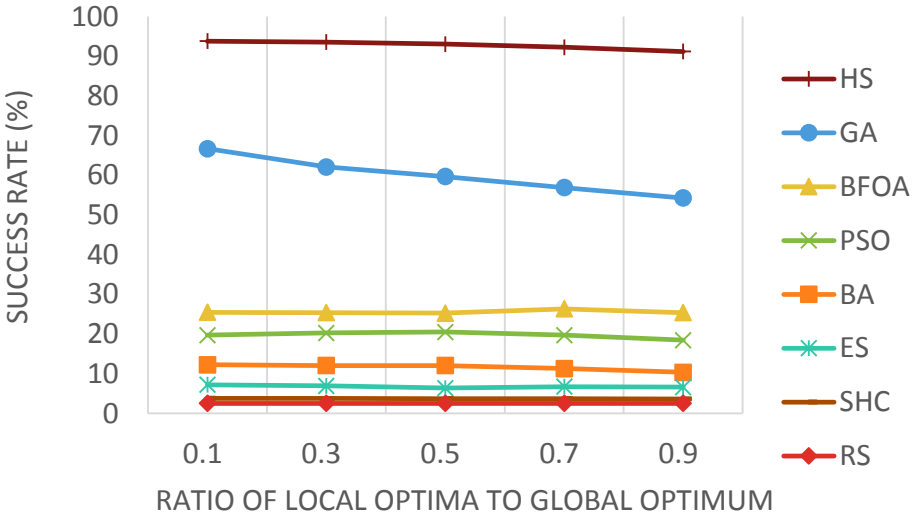
It is clear that amongst the algorithms selected for study, examining the ratio of local optima to global optimum highlights a range of different performance profiles. For the majority of the algorithms, there are also significant performance differences



(a) Accuracy.

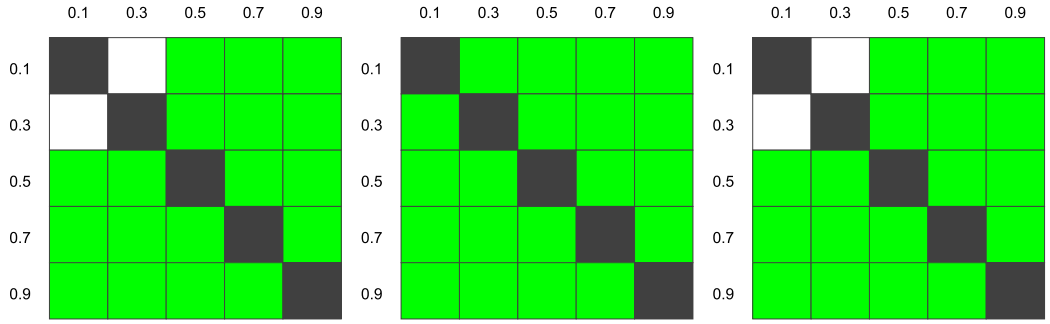


(b) Diversity.



(c) Success rate.

Figure 4.5: Result of varying the average ratio of local minima to the global minimum.



(a) Bacterial foraging optimisation algorithm.

(b) Bees algorithm.

(c) Evolution strategy.

(d) Genetic algorithm.

(e) Harmony search.

(f) Particle swarm optimisation.

(g) Stochastic hill-climbing.

(h) Random search.

Figure 4.6: The results of unpaired two-tailed t-tests for each algorithm comparing algorithm performance results at one ratio of local optima to global optimum with every other ratio of local optima to global optimum (H_0 = The ratio of local optima to global optimum in the fitness landscape has no effect on an algorithm's performance, H_A = The ratio of local optima to global optimum in the fitness landscape has an effect on an algorithm's performance). Green shaded cells depict a p-value below 0.05. Where a cell is shaded, a significant difference has been found between the results with one ratio of local optima to global optimum (row) versus the second ratio of local optima to global optimum (column). An algorithm with more shaded cells indicates it is more sensitive to changes in the ratio of local optima to global optimum (i.e. there are more statistically significant performance differences as the ratio of local optima to global optimum changes) than one with less shaded cells.

between different ratio values, suggesting it is a good characteristic for distinguishing between algorithms in terms of performance.

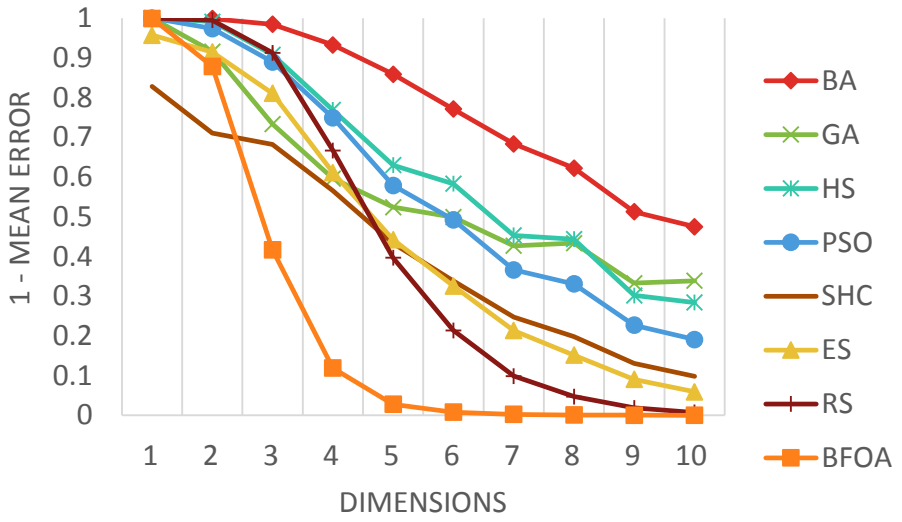
4.3.3 Dimensionality

Results are depicted in Fig. 4.7. At only one dimension, fitness landscapes are trivially easy. The performance of all algorithms reflects this, with all algorithms performing well on landscapes of a single dimension. All algorithms show a success rate (that is, optimisation with an error of under 1.0×10^{-4}) above 90%.

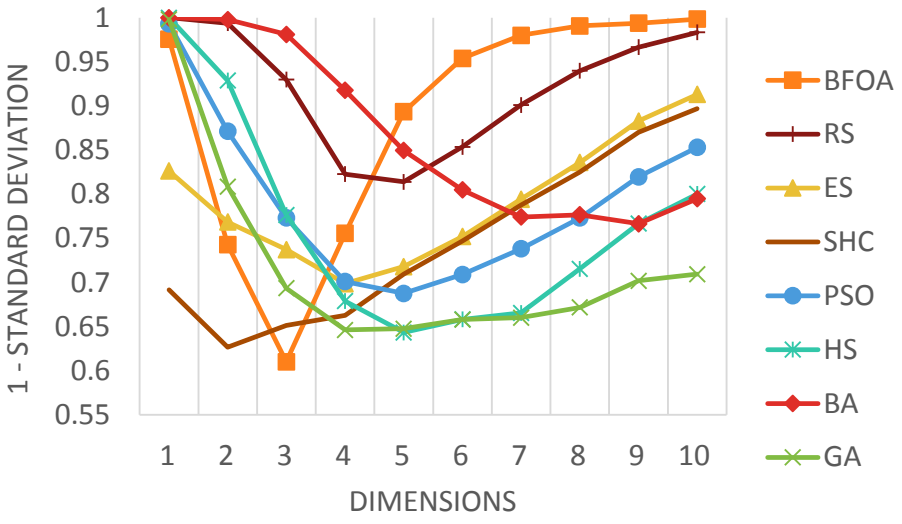
Once dimensionality is increased to two, the performance of most algorithms begins to degrade. Suffering mostly severely is RS, which is to be expected, as random search is the most basic algorithm. Its success rate drops from a 99.98% to 2.94%. Algorithms which also perform poorly at only two dimensions are ES (from 93.79% success to 7.46%), BA (from 100% success to 13.57%) and PSO (from 99.79% success to 19.66%). It is perhaps surprising, at first, to see BA performing poorly given that the algorithm contains a randomly sourced global search. However, this global search is effectively RS, which performs poorly, so it may be reasonable to assume the global search is not covering enough of the landscape. Coupled with the non-adaptive nature of the algorithm (meaning that solution selection around the current best area is within a relatively large range), poor algorithm performance is easily explained. It is reasonable to suggest that PSO and ES suffer from a similar problem, in that exploration is limited, and neither optimise their current best as accurately as their adaptive variants.

As dimensionality increases beyond two, the performance of all algorithms, as expected, continues to degrade significantly. At only three dimensions, BA, ES, PSO, SHC, BFOA and RS all demonstrate particularly poor performance in terms of success. Performing with the greatest success rate is HS, still dropping to a 40% success rate at three dimensions, with GA in second place offering a 24% success rate.

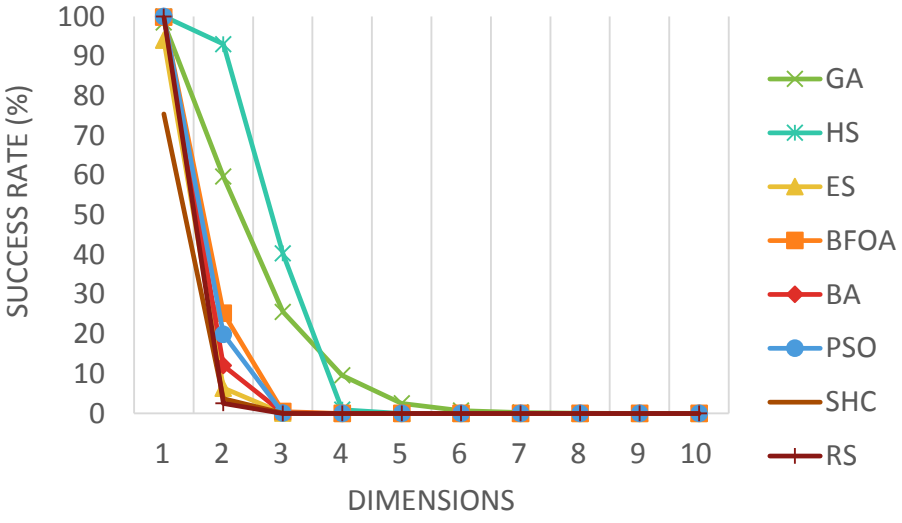
These results are reaffirmed by the statistical analysis (results in Fig. 4.8), which shows significant performance differences for *all* algorithms when comparing *all* numbers of dimensions, with two exceptions. There is no significant difference between the performance of the GA at seven and eight dimensions, or at nine and ten dimensions - though these seem to be unusual cases rather than the standard. On the whole, it is apparent that dimensionality has a large, significant effect on performance of all the selected algorithms, noticeable by both the observed performance differences and the statistical testing.



(a) Accuracy.

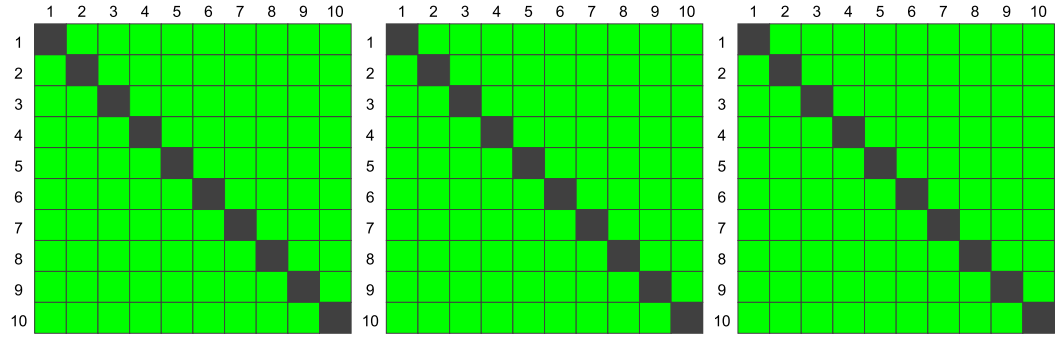


(b) Diversity.



(c) Success rate.

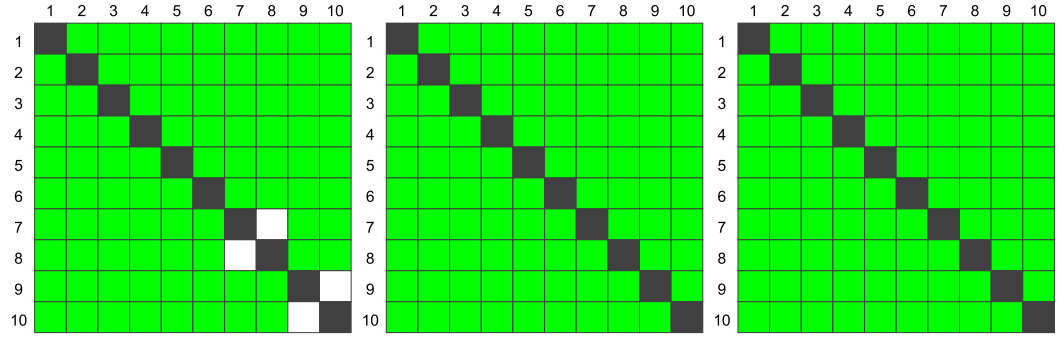
Figure 4.7: Result of varying dimensionality.



(a) Bacterial foraging optimisation algorithm.

(b) Bees algorithm.

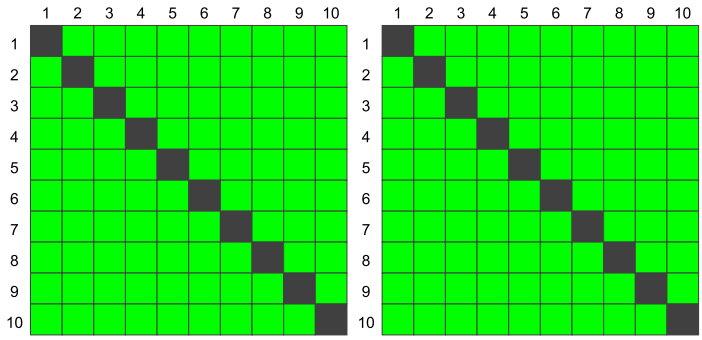
(c) Evolution strategy.



(d) Genetic algorithm.

(e) Harmony search.

(f) Particle swarm optimisation.



(g) Stochastic hill-climbing.

(h) Random search.

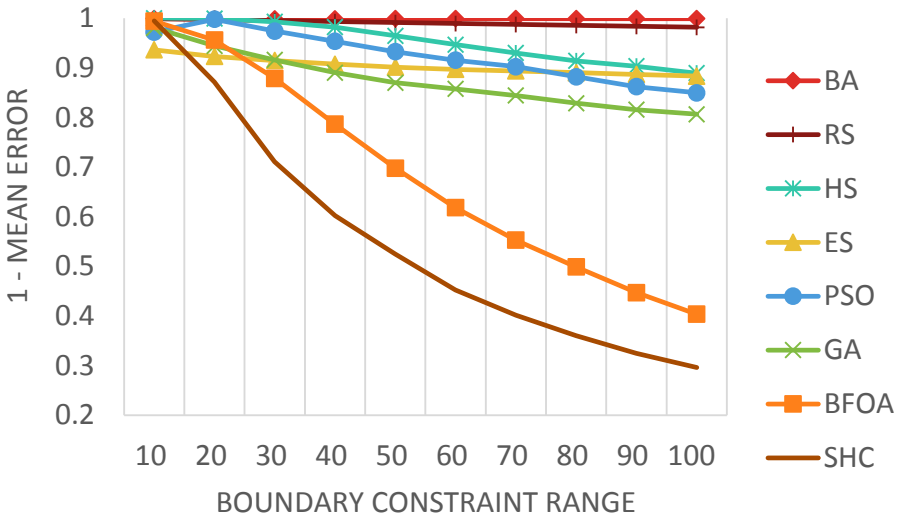
Figure 4.8: The results of unpaired two-tailed t-tests for each algorithm comparing algorithm performance results at one number of dimensions with every other number of dimensions (H_0 = The number of dimensions in the fitness landscape has no effect on an algorithm's performance, H_A = The number of dimensions in the fitness landscape has an effect on an algorithm's performance). Green shaded cells depict a p-value below 0.05. Where a cell is shaded, a significant difference has been found between the results with one number of dimensions (row) versus the second number of dimensions (column). An algorithm with more shaded cells indicates it is more sensitive to changes in the number of dimensions (i.e. there are more statistically significant performance differences as the number of dimensions changes) than one with less shaded cells.

4.3.4 Boundary constraints

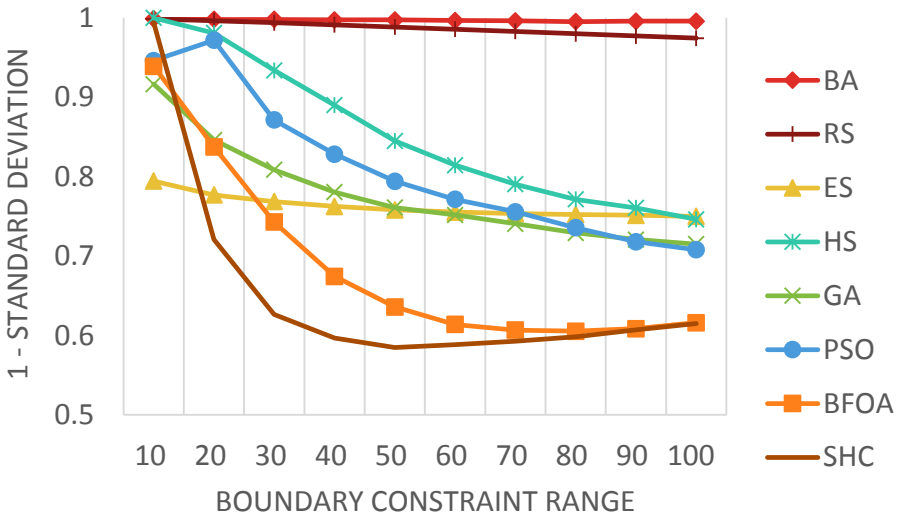
Results are depicted in Fig. 4.9. RS exhibits a similar, yet less extreme, reaction to the increase of problem space as with the increase in dimensionality. This is to be expected, as the limit on objective function calculations results in random search having less chance to explore the search space. SHC also has an almost linear increase in average error, matching the linear increase in search space size, but produces consistently poor results in terms of success. The social system algorithms (BA and PSO) both exhibit slightly unusual behaviour - as the problem space increases, their success rate also increases. This suggests that their reliance on a parameter to search within a range is hindering the algorithms when the problem space is too small to properly explore. HS provides the best success rate for the entire range of sizes selected in this problem, indicating good exploration of the search space irrespective of the range parameter. BFOA also suffers significantly as search space size increases, again implying a heavy reliance on the parameter which controls the range of search for new solutions. The evolutionary algorithms do not cope particularly well with the increase of problem size, with performance in terms of both average error and success rate decreasing consistently as size increases.

Statistical analysis shows that most algorithms show significant differences across all characteristic values (Fig. 4.10). BFOA, GA, HS, SHC and RS all have significant differences for every comparison. The BA, as discussed above, shows significant differences up to a point, with a majority of significant differences occurring between boundary constraint range values up to fifty. Above fifty, there are no significant differences between performance - except between very extreme values (fifty and one hundred), indicating that there is a large boundary range for which the BA will perform consistently. ES shows some resilience, with no significant performance difference found between most border cases (e.g. forty and fifty, fifty and sixty), suggesting that despite its noted poor performance above, the ES is not as strongly affected by change in boundary constraint range as most of the other algorithms.

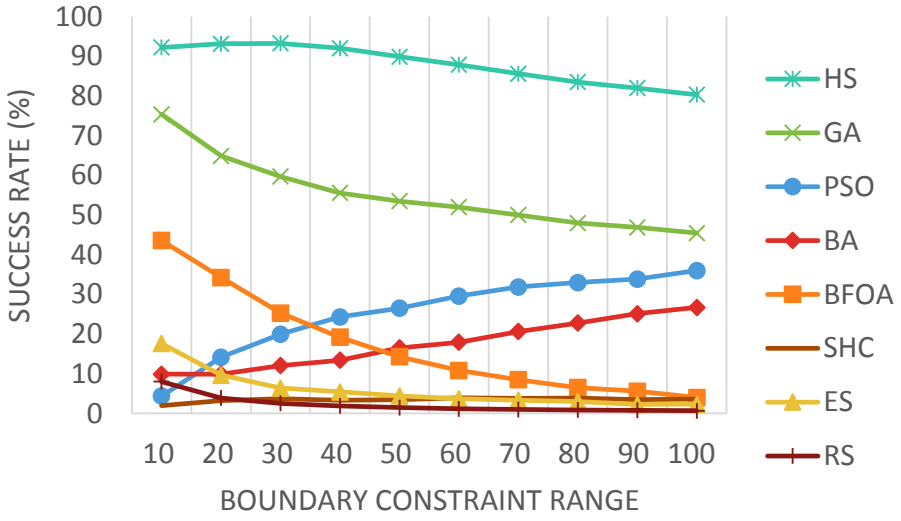
Although nearly all the algorithms show significant performance differences for boundary constraint range, the magnitude of the effect boundary constraint range has on each of the algorithms is quite varied - for example, as mentioned, BFOA has a much greater increase in mean error as boundary constraint range increases compared to an algorithm with only a minor mean error increase such as PSO. The performance profiles here are perhaps not as obviously varied, but there is certainly still performance variety, and distinguishing features, between some, if not all, of the selected algorithms.



(a) Accuracy.

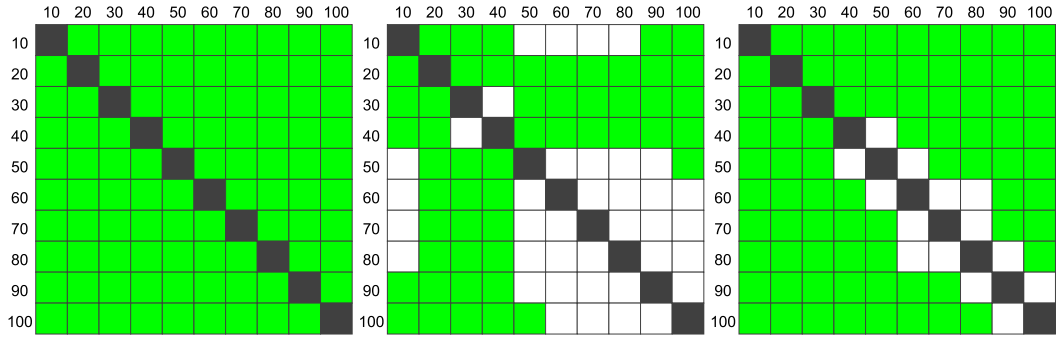


(b) Diversity.



(c) Success rate.

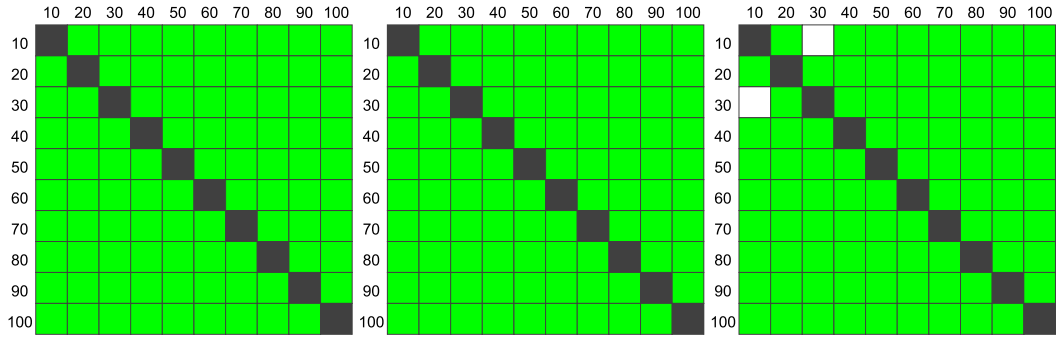
Figure 4.9: Result of varying boundary constraint range.



(a) Bacterial foraging optimisation algorithm.

(b) Bees algorithm.

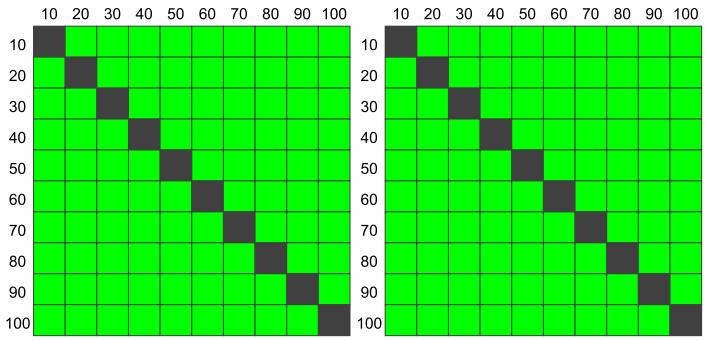
(c) Evolution strategy.



(d) Genetic algorithm.

(e) Harmony search.

(f) Particle swarm optimisation.



(g) Stochastic hill-climbing.

(h) Random search.

Figure 4.10: The results of unpaired two-tailed t-tests for each algorithm comparing algorithm performance results at one boundary constraint range with every other boundary constraint range (H_0 = The boundary constraint range of the fitness landscape has no effect on an algorithm's performance, H_A = The boundary constraint range of the fitness landscape has an effect on an algorithm's performance). Green shaded cells depict a p-value below 0.05. Where a cell is shaded, a significant difference has been found between the results with one boundary constraint range (row) versus the second boundary constraint range (column). An algorithm with more shaded cells indicates it is more sensitive to changes in the boundary constraint range (i.e. there are more statistically significant performance differences as the boundary constraint range changes) than one with less shaded cells.

4.3.5 Smoothness

Results are depicted in Fig. 4.11. The evolutionary algorithms (ES and particularly GA) perform poorly, with GA being greatly affected by changing the smoothness coefficient. BA and PSO all also show decreasing success rate as the curves become steeper, as does BFOA which relies heavily on gradient information.

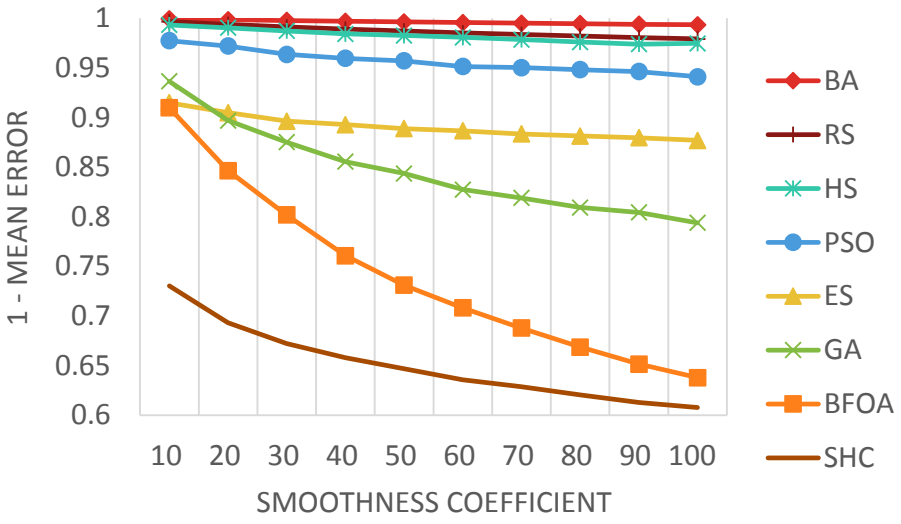
Harmony search suffers similarly to the evolutionary algorithms, and swarm algorithms, as curves become more steep. The similarity in terms of success rate for all algorithms suggests that the availability of gradient information is something which affects all algorithms, including random search to some respect. This is largely due to the decreased area of successful solutions, which hinders the likelihood of "stumbling" across a good solution irrespective of the way in which gradient information is used, or not used, by the specific algorithm.

Statistical analysis confirms the observations (Fig. 4.12), showing that BFOA, BA and the GA are all significantly affected by the change in smoothness coefficient at all (or nearly all) characteristic values. RS also shows significant performance differences at every value - perhaps due to the availability of good solutions, rather than a reliance on gradient information to succeed. HS, PSO and SHC both show some resilience to changes in smoothness coefficient around edge cases (e.g. forty to fifty, seventy to eighty), but this only holds true within a limited range of smoothness coefficient. That is to say, they show *some* resilience compared to the other algorithms, but not a particularly noteworthy amount. ES demonstrates a greater resilience towards changing smoothness coefficients than the other algorithms - although, as noted previously, its performance is generally poor compared to some of the other algorithms. Above a smoothness coefficient of forty, ES demonstrates less significant change between smoothness coefficients as the value increases. As performance does degrade slightly before this point, it could be that ES reaches a point where further changes to smoothness coefficient do not further hinder the performance - but a greater range of smoothness coefficients would need to be tested to confirm this.

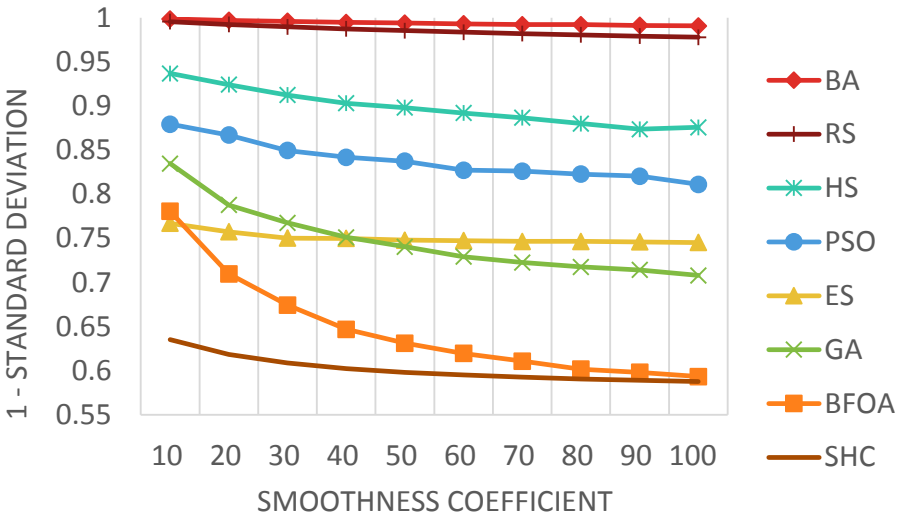
As with the other characteristics, a fitness landscape's smoothness (and by extension, availability of gradient information) is a characteristic capable of determining a range of performance profiles for different algorithms. Each of the algorithms studied here shows a different level of robustness to changes in the smoothness coefficient used to generate the fitness landscapes, in the form of greater or lesser declines in mean error.

4.3.6 Overview

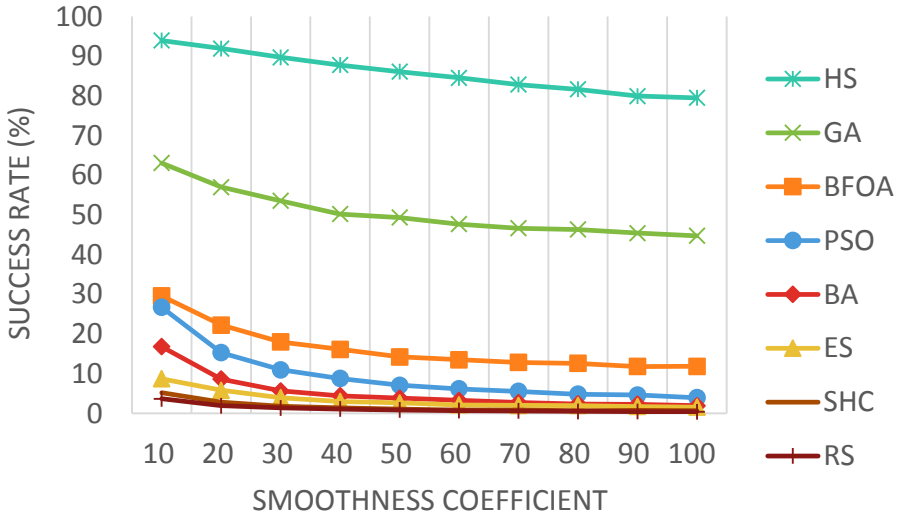
To summarise the results, the resilience of each algorithm to changing landscape characteristics is presented in the form of a radar plot in Fig. 4.13. To assess the resilience of an algorithm, the standard deviation of the average error across all values of a landscape characteristic is used, normalised on a per-characteristic basis. This "ranking" shows which algorithms do not show performance variability versus those which are



(a) Accuracy.

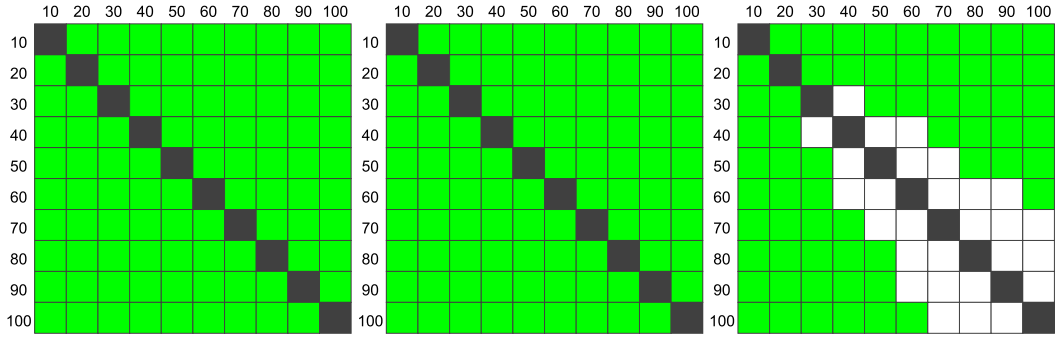


(b) Diversity.



(c) Success rate.

Figure 4.11: Result of varying the smoothness coefficient.



(a) Bacterial foraging optimisation algorithm.

(b) Bees algorithm.

(c) Evolution strategy.

(d) Genetic algorithm.

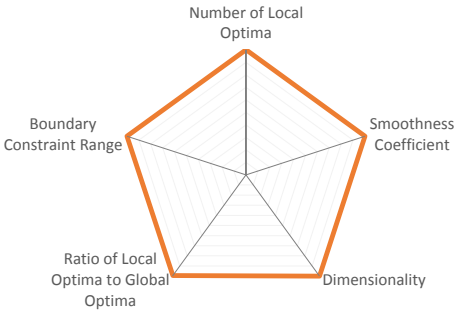
(e) Harmony search.

(f) Particle swarm optimisation.

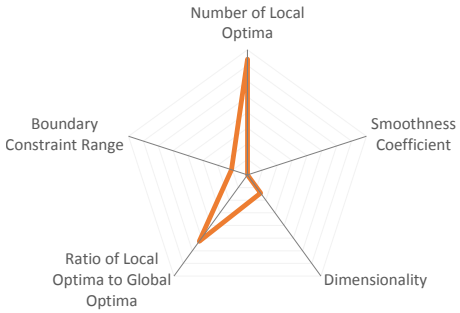
(g) Stochastic hill-climbing.

(h) Random search.

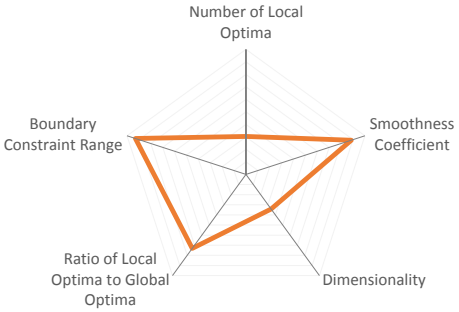
Figure 4.12: The results of unpaired two-tailed t-tests for each algorithm comparing algorithm performance results at one smoothness coefficient with every other smoothness coefficient (H_0 = The smoothness coefficient of the fitness landscape has no effect on an algorithm's performance, H_A = The smoothness coefficient of the fitness landscape has an effect on an algorithm's performance). Green shaded cells depict a p-value below 0.05. Where a cell is shaded, a significant difference has been found between the results with one smoothness coefficient (row) versus the second smoothness coefficient (column). An algorithm with more shaded cells indicates it is more sensitive to changes in the smoothness coefficient (i.e. there are more statistically significant performance differences as the smoothness coefficient changes) than one with less shaded cells.



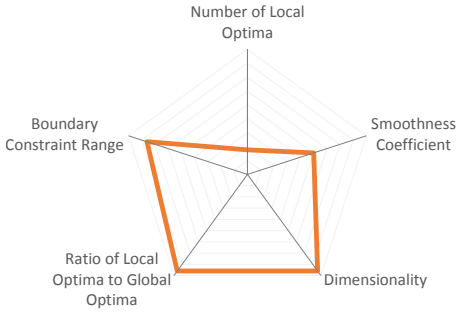
(a) Bees algorithm.



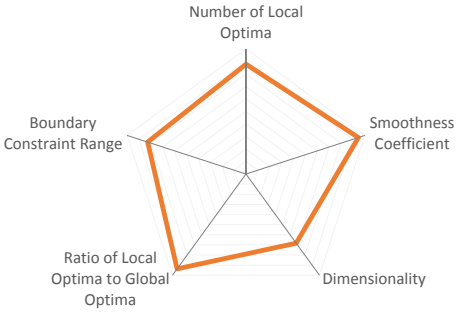
(b) Bacterial foraging optimisation algorithm.



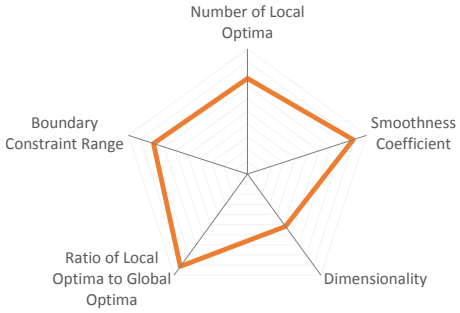
(c) Evolution strategies.



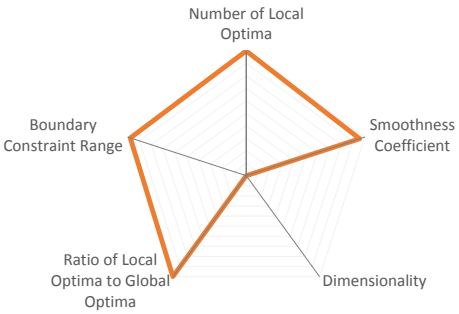
(d) Genetic algorithm.



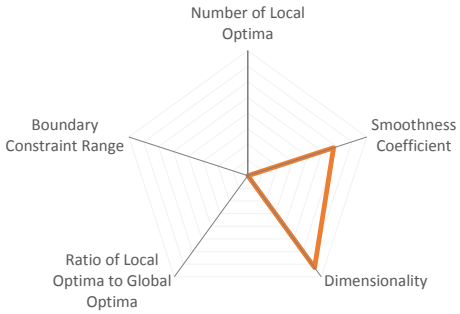
(e) Harmony search.



(f) Particle swarm optimisation.



(g) Random search.



(h) Stochastic hill climbing.

Figure 4.13: Radar plots depicting the standard deviation of the average error of each algorithm with respect to differing landscape characteristics. Standard deviations are normalised on a per-axis basis. Values close to the centre of the plot indicate a larger variation in average error, indicating these algorithms are more affected by the characteristic.

heavily influenced by a characteristic. BFOA shows large deviations in average error to boundary constraint range, smoothness coefficient changes and dimensionality, indicating that BFOA is an algorithm heavily dependent on the landscape of a problem - perhaps because of a heavy reliance on careful parameterisation. SHC also shows large

variation - perhaps, in large part again, to the lack of parameters and complicated local optima avoidance techniques, etc. GA and ES depict a large variation with respect to number of local optima, confirming earlier speculation that evolutionary algorithms suffer the problem of becoming "stuck" in a local optima most significantly.

4.4 Summary

In this Chapter, the results of an extensive study of nature-inspired algorithms have been described, in terms of their performance on fitness landscapes with different characteristics. Six nature-based methods (plus two stochastic baseline algorithms) were studied, by varying a number of landscape features. The most significant characteristic appears to be the number of local minima, where a combination of global and local search appears to be beneficial. On the other hand, the ratio of local optima to the global minimum appears to have little effect on the success of the algorithms under study. As expected, dimensionality proved problematic for all algorithms, whereas landscape smoothness appeared to have little effect.

In obtaining performance profiles for each of these algorithms, the methodology proposed in Chapter 3 (and used here) has been demonstrated to be suitable for establishing the performance profile of an algorithm. Each of the algorithms in the study showed differing resilience to the varying characteristics (as well as different performance profiles across the characteristics), and this allows for the algorithms to be distinguished from each other in terms of performance.

This also answers the first of the research questions posited in this thesis. In using fitness landscape characteristics to analyse the performance of the selected algorithms, unique and varied performance profiles have been obtained. While there are some characteristics which are unanimously difficult in this study (e.g. dimensionality), they provided varying degrees of difficulty across a range of algorithms. Similarly, the algorithms showed varying degrees of resilience to all of the characteristics (i.e. the algorithms exhibit various levels of robustness to these characteristics), owing to the algorithms exhibiring a range of different features, parameters, etc. Ultimately, although this work offers only the first steps in illustrating how fitness landscape characteristics could be used to develop performance profiles of algorithms, with the purpose of differentiating algorithm performance, the results are promising.

The results obtained here offer an insight into only a small section of the algorithms available to practitioners, and so only go a little way to offering a picture of the nature-inspired field. One of the most obvious ways in which this study could be built upon is to increase the number of algorithms for which this technique is applied, developing a much richer overview of the general strengths and weaknesses of nature-inspired algorithms for the continuous optimisation domain.

In particular, certain algorithms, such as the GA, offer many customisable components, and in this Chapter only a single possible configuration has been examined. The technique presented here would be entirely plausible for analysing the benefit, or

drawbacks, of choosing other problem representation, selection, crossover and mutation strategies, and how they affect the performance of the algorithm with respect to each of the identified characteristics of the fitness landscapes.

Another drawback of this technique is potential difficulty in identifying some of the characteristics of the landscapes. While the characteristics of dimensionality and boundary constraint range are specified by the problem - at least, in the case of constraint continuous optimisation problems such as these - features such as number of local optima, ratio of local optima to the global optimum and smoothness coefficient, require *landscape sampling* to estimate, and this is a process that can become more time consuming and difficult than finding a solution. Further work may look at the different techniques used to estimate these characteristics and relate this to how *accurate* a prediction needs to be to obtain a good estimate of performance based on the strengths and weaknesses identified by an analysis technique such as this.

A final criticism that could be made of the work in this Chapter is that these algorithms were all used "out of the box" with no effort expended into the process of parameter tuning. This is very unlike the way algorithms are used in practice, as very often considerable effort is made to use algorithms with carefully selected parameters, to ensure optimal performance. In the following Chapter, this work is expanded upon using an automated parameter tuning methodology to examine the performance of algorithms, with respect to landscape characteristics, both pre- and post-tuning, offering an insight into how tuning affects algorithm performance.

Chapter 5

Investigation of the Relationship Between Parameter Tuning and Landscape Characteristics

5.1 Introduction

Parameter tuning is a considerable obstacle in the implementation of a nature-inspired algorithm on a given problem, and Eiben and Smit (2011) discuss the importance of ensuring parameter tuning takes place. It is commonly accepted that there is little to no *pattern* to parameter values, and often it is not obvious how parameters relate to the properties of problems. While articles describing novel algorithms may make some suggestions as to sensible parameter ranges, and how changing these parameters affects the exploration pattern of the algorithm, it is often unintuitive as to which values prove promising for a given problem without a large amount of trial and error. Tuning, therefore, can often become a task more intractable than the optimisation problem itself. There is then no discussion as to how *valuable* parameter tuning is in obtaining improved performance, which means many hours could be wasted tuning an algorithm which does not benefit significantly from the tuning process.

There is great benefit in exploring an algorithm in terms of parameter tuning using *scientific testing* (Hooker, 1995), for example to explore the *robustness* of an algorithm to changes in problem specification (Eiben and Smit, 2011). In this Chapter the six different nature-inspired algorithms are examined by testing them against a number of different randomized landscapes with several different pre-defined properties (e.g., ruggedness). An automated parameter tuning method is used to obtain performance data both pre- and post-tuning, which enables a breakdown of the effect tuning has on algorithm performance, individualised by a landscape characteristic. This offers a more "complete" view of the relationship between parameter tuning, performance and problem specification, by highlighting which specific characteristics of the problems are relevant to the tuning process.

The rest of the Chapter is organised as follows: in Section 5.2 a a brief overview of

previous work is presented before the testing methodology is described in Section 5.3. Experimental results are then presented in Section 5.4, before conclusions are drawn in Section 5.5 with a discussion of findings and further work.

5.2 Background

In this Chapter, work from the previous Chapter is expanded upon with the inclusion of the performance of algorithms *after* parameter tuning has taken place, thus resolving the key criticism that algorithms are not commonly used 'out of the box.' By examining the same six different nature-inspired methods both pre- and post- tuning, conclusion can be made regarding the effect parameter tuning has on each of the different algorithms with respect to landscape characteristics. Analysis of the benefit of parameter tuning in other fields (for example, text categorisation (Koster and Beney, 2007)), has found great importance in tuning parameters to achieve maximal performance, and Eiben and Smit (2011) echo this notion of importance for implementing evolutionary algorithms. While there are some individual studies in the importance of parameter tuning and parameter control for specific algorithms (such as that for the Bee Colony Algorithm provided by Akay and Karaboga (2009), or Evolutionary Algorithms provided by Nannen et al. (2008)) there are no comprehensive studies covering a range of nature-inspired algorithms, or any studies which relate the benefit of tuning to fitness landscape characteristics, identifying *when* it is beneficial to tune.

Racing, first introduced in the field of machine learning Maron and Moore (1993, 1997), is one approach proposed to deal with the intractable task of parameter tuning. The racing methodology suggests that a subset of algorithm configurations should be generated, and their performance analysed on a small subset of problems. Those configurations which are found to be performing *statistically significantly* better are carried through to a further step of the racing methodology, which increases the problem space to gain more information about the promising algorithms, leading to narrowing of the algorithm configuration pool. These steps are repeated until only one configuration remains, or a maximum number of steps has been reached.

Many variants of the racing methodology exist, mostly focusing on the *distribution* used for determining which configurations are performing well. Hoeffding (1963) proposes the use of Hoeffding's formula for the confidence measure, and Bayesian statistics were later recommended by Maron and Moore (1997). Eventually a *middle ground* was reached in the form of a race using the Friedman test. Originally proposed by Birattari et al. (2002), the F-Racing methodology is both non-parametric and takes advantage of a blocking design. Refined later, Balaprakash et al. (2007) introduce the notion of *sampling design* and *iterative refinement* - sampling design consisting of randomly generated algorithm configurations and iterative refinement offering a method for generating new configurations that seem promising. The performance of F-Racing was compared to other methodologies by Yuan and Gallagher (2004), and F-Races were found to be most effective at eliminating candidates, and overall, racing is a promising

methodology drastically reducing the compute time compared to exhaustive experiments (Smit and Eiben, 2009).

Adopting a *scientific* testing strategy, instead of a *competitive* testing as suggested by Hooker (1995) and Eiben and Smit (2011), the methodology of Chapter 4 is enhanced, and this methodology is discussed in the following section.

5.3 Methodology

Having demonstrated in Chapter 4 that the methodology proposed in Chapter 3 can be used to establish the variation in performance of nature-inspired algorithms *cross-algorithm*, the second question addressed by this thesis asks whether fitness characters can be used as a way of comparing *intra-algorithm* changes.

The methodology used is exactly as proposed in Chapter 3, with one crucial change: Two data-sets are generated rather than one, allowing for comparison between them. The first uses the exact same algorithms, characteristics and algorithm parameters as in the standard methodology. The second uses algorithm parameters that have been automatically tuned for the landscape characteristics in question using an F-Race. Specific details of the F-Race configuration follow.

5.3.1 F-Race Configuration

An F-Race (Birattari et al., 2010) framework was implemented in the Ruby scripting language, selected for convenience and easy integration with the existing algorithm codebase. The initial pool was populated with 500 configurations for each algorithm using the random sampling design methodology (Balaprakash et al., 2007; Birattari et al., 2010) taking random values for each parameter between sensible limits, drawn from the original literature for each algorithm where possible. Ranges for each parameter of each algorithm are shown in Table 5.1.

To produce a test set of problems, the MSG landscape generator was used. Five problems were generated to form the initial problem set for each of the landscape characteristic values. Algorithms were executed until reaching 20,000 objective function calculations, and repeated with a differing random seed twenty times. The average exact error ($\epsilon = |v - v_{est}|$, where v is the known best solution and v_{est} is the best solution found) across the twenty repeat runs was used as the performance criteria for the F-Race, which itself used a 0.05 significance level for the rejection of the null hypothesis. This is a commonly used significance level that also performed well in initial testing (Yuan and Gallagher, 2004; Birattari et al., 2010). After each step of the F-Race process, the problem space was expanded with an additional five randomly generated landscapes, providing a more complete picture of the performance of the algorithms which proved promising. The F-Races were terminated when there is only one configuration remaining, or ten race steps have been reached - whichever occurs first.

Table 5.1: Parameter ranges for the generation of algorithm configurations. Where s is used, it represents the size of the problem space in each dimension.

Bees Algorithm	
Number of bees (n)	(1, 250]
Patch size	(0, s]
Number of sites (i)	[1, n]
Number of elite sites	[1, i]
Number of elite bees (e)	[1, n]
Number of other bees	[1, e]

Particle Swarm Optimisation	
Population size	[20, 40]
Maximum velocity	(0, s]
Personal best weight	(0, 4]
Global best weight	(0, 4]

Evolution Strategies	
Population size (n)	(1, 250]
Number of children	(1, n]
Strategy mutation	Off

Genetic Algorithm	
Population Size	(1, 250]
Bits per variable	[8, 64]
Crossover probability	(0, 1)
Mutation probability	(0, 1)

Harmony Search	
Range	(0, s]
Memory size	(1, 250]
Consideration rate	(0, 1)
Adjustment rate	(0, 1)

Bacterial Foraging Optimisation Algorithm	
Step size	(0, s]
Population size	(5, 50]
Swim length	[1, 10]
Elimination chance	(0, 1)
Attractant depth	(0, 2)
Attractant width	(0, 2)
Repellent height	(0, 2)
Repellent width	(0, 2)
Chemotactic steps	[1, 100]
Reproduction steps	[1, 10]

Stochastic Hill Climbing	
Neighbourhood size	(0, s]

F-Races were performed for a range of characteristics described previously using the same value ranges for each characteristic as in Chapter 3, giving optimised parameters for *each* value of *every* landscape characteristic used. The optimised parameters are then used to perform an identical performance evaluation of each algorithm. This provides adequate performance data of the algorithms against landscape characteristics, using both default parameters and optimised parameters. This data is analysed in the next section, with reference to the effect tuning has on each of the algorithm under different landscape characteristic combinations and conditions.

5.4 Results

The effect of tuning was varied across all of the algorithms chosen for the study, though broadly algorithms fit into three categories:

- Algorithms which did not benefit from tuning
  - Evolution Strategies (ES)
- Algorithms which only benefit notably from tuning when a landscape is 'difficult' for the algorithm with default parameters
  - Bees Algorithm (BA)
  - Harmony Search (HS)
  - Particle Swarm Optimisation (PSO)
- Algorithms which always benefit from tuning
  - Bacterial Foraging Optimisation Algorithm (BFOA)
  - Genetic Algorithm (GA)
  - Stochastic Hill Climbing (SHC)

It is worth noting, however, that GA and SHC were unable to tune when the problem was *too* difficult (i.e. when dimensionality was high).

Summarised results, including the average error of the algorithm performance across characteristic values and the standard deviation of average error across all characteristic values (effectively, how much performance varied across landscape characteristic values), are presented in Table 5.2, with the complete configuration of parameters for each algorithm available in Appendix B and complete performance data available in Appendix C. Additionally, Table 5.3 presents a count of the number of unique parameter configurations selected by the F-Racing process for each algorithm, both in total and for each characteristic. The performance of each algorithm individually is now examined, and the parameters available for tuning investigated to provide suggestions as to why the results are reasonable.

Table 5.2: The mean average (μ) and standard deviation (σ) of the exact average error of algorithm performance (both untuned (UT) and tuned (T)) as different landscape characteristic values are used in the landscape generation process. A smaller standard deviation indicates that average error changes less as the characteristic value changes, indicating good robustness to this characteristic. All untuned and tuned mean differences are significantly different, as determined by a two-tailed t-Test with a confidence value of 0.05, except those with p-Values (listed in the table as *p) highlighted in* ***bold***.

		BFOA		**BA**		**ES**		**GA**		**HS**		**PSO**		**SHC**	
		UT	**T**	**UT**	**T**	**UT**	**T**	**UT**	**T**	**UT**	**T**	**UT**	**T**	**UT**	**T**
# of Local Optima	μ	0.118	0.003	0.001	8.8×10^{-6}	0.085	0.078	0.093	0.015	0.011	2.2×10^{-5}	0.025	0.014	0.266	0.072
	σ	0.011	0.001	2.1×10^{-4}	9.2×10^{-7}	0.028	0.026	0.033	0.008	0.005	2.9×10^{-5}	0.010	0.010	0.041	0.020
	p	7.63×10^{-11}		5.79×10^{-9}		1.98×10^{-4}		2.32×10^{-5}		7.16×10^{-5}		0.021		4.80×10^{-9}	
Dimensions	μ	0.754	0.417	0.216	0.073	0.542	0.544	0.420	0.529	0.364	0.263	0.420	0.157	0.577	0.589
	σ	0.388	0.360	0.202	0.069	0.345	0.346	0.233	0.402	0.271	0.204	0.307	0.145	0.261	0.371
	p	0.002		0.008		**0.157**		**0.113**		0.005		9.96×10^{-4}		**0.755**	
Local Optima Ratio	μ	0.120	0.003	0.001	8.7×10^{-5}	0.084	0.082	0.079	0.007	0.007	0.002	0.025	0.016	0.284	0.088
	σ	0.021	0.002	2.3×10^{-4}	1.9×10^{-4}	0.012	0.012	0.006	0.006	0.003	0.004	0.004	0.011	0.045	0.011
	p	2.01×10^{-4}		4.14×10^{-4}		**0.116**		1.29×10^{-5}		0.033		**0.080**		2.19×10^{-4}	
Boundary Range	μ	0.317	0.022	0.001	0.002	0.097	0.093	0.125	0.021	0.048	0.001	0.076	0.022	0.446	0.305
	σ	0.213	0.033	1.3×10^{-4}	0.002	0.017	0.018	0.057	0.016	0.041	0.001	0.050	0.013	0.239	0.217
	p	1.05×10^{-3}		**0.794**		1.20×10^{-3}		1.23×10^{-4}		5.34×10^{-3}		1.43×10^{-2}		4.83×10^{-5}	
Smoothness	μ	0.260	0.010	0.004	0.001	0.110	0.102	0.154	0.021	0.018	0.001	0.043	0.014	0.349	0.112
	σ	0.089	0.005	0.002	4.3×10^{-4}	0.012	0.012	0.045	0.011	0.007	0.001	0.012	0.006	0.039	0.014
	p	6.50×10^{-6}		2.23×10^{-4}		$9.54\times^{-7}$		6.99×10^{-6}		2.38×10^{-5}		7.98×10^{-7}		2.60×10^{-10}	

Table 5.3: The count of different parameter configurations selected by the F-Racing process for each algorithm for each characteristic, and also the number of unique configurations selected across all characteristics. The maximum number of possible configurations for each characteristic is ten, except for 'Ratio of Local Optima to Global Optimum' which only has nine possible configurations. This makes for a total of 49 possible unique configurations for each algorithm.

	BA	BFOA	ES	GA	HS	PSO	SHC
Number of Local Optima	1	4	10	3	3	5	8
Number of Dimensions	4	4	10	6	7	3	10
Ratio of Local Optima to Global Optimum	3	3	9	3	9	3	9
Boundary Constraint Range	3	4	10	4	10	6	10
Smoothness Coefficient	2	4	10	2	10	3	10
Different Configurations Across All Characteristics	**4**	**9**	**46**	**8**	**34**	**12**	**44**

5.4.1 Bacterial Foraging Optimisation Algorithm

Proving the most difficult to tune, there is little discussion on the role of the different parameters in the BFOA. While some elements of the search pattern are inherently altered by various parameters, it is very difficult to estimate sensible values for these. In the original proposal of the BFOA (Passino, 2002), the parameters were assigned based on observation of actual bacterial colonies. While this is true to the nature-inspired concept, it is not necessarily the best way to get the optimal performance from the algorithm.

Broad ranges for all parameters of the BFOA were selected to avoid being too restrictive on any particular parameter. The F-Races returned configurations which proved promising, and obtaining performance data using these configurations highlights the importance of tuning the parameters for bacterial foraging, which is the most tuning-sensitive of the selected algorithms. BFOA also provides the *most* parameters for tuning, suggesting a possible link between number of parameters affecting the search behaviour and tuning sensitivity.

The combination of parameters offered by BFOA gives a highly configurable search environment. Parameters such as step size and population size directly affect the potential area the algorithm can explore in a given number of objective function calculations. Additional parameters include attraction and repulsion weights and the "space" over which these attraction and repulsion effects spread. Working in a similar manner to the personal best and global best weightings offered by particle swarm optimisation, these control the reliance of an individual on the solutions found by the rest of the population. The final set of parameters control the number of 'chemotactic steps'[1] that occur before a reproduction step, and the number of reproduction steps that occur before an elimination-dispersal event. This has a direct effect on the search behaviour:

[1]That is, steps in which the population perform a local search.

A large number of chemotactic steps encourages broad exploration, while fewer encourages more reproductive steps - thus a larger focus on exploring promising areas. It is clear from the descriptions of these parameters that the search space is highly configurable for different problems, and the reliance on tuning follows logically.

Across all characteristics, tuning has a vast improvement on the average error and standard deviation of the bacterial foraging optimisation algorithm - in many cases, improving from the largest average error to one of the smallest, and coping well with the changing characteristics. Tuning provides the largest performance improvement where boundary constraint ranges change, a characteristic that is heavily reliant on parameters which control the range from which new solutions are selected (in the case of BFOA, this is the step size). Improvements are also shown for dimensionality and smoothness coefficient, increasing the performance of BFOA where there is little gradient information in a large fitness landscape. Smaller improvements were demonstrated by the increasing number of local optima and the increasing attractiveness of these local optima, but tuning still benefits the algorithm considerably.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.1, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that, on the whole, there is a clear improvement in average error for all characteristics and, for at least three of the five characteristics, there is a clear improvement in the spread of results. It would appear that BFOA always benefits from the tuning process, and in most cases, the performance improvement is quite noticeable. Each characteristic is now examined in detail.

As described above, there is an overall improvement to average errors for all values of number of local optima (from 0.118 untuned to 0.003 tuned). The range also narrows slightly (from 3.82×10^{-2} untuned to 3.77×10^{-3}) tuned), suggesting that the ability of the algorithm to cope with changes to the number of optima has increased with tuning. This improvement is almost identical to that shown with changes to the ratio of optima, which shows an improvement of average error again (from 0.120 untuned to 0.003 tuned) and another narrowing of the spread of results (from 5.25×10^{-2} untuned to 4.07×10^{-3}) tuned).

Boundary constraint range shows improvements again, but on a much greater scale. Originally offering a much greater challenge for the BFOA, with an mean average error of 0.317, post-tuning the mean average error improves to 0.022). The improvement here is much greater than that of number and ratio of local optima, though does not reach the same level of improvement, suggesting that there is still some difficulty presented by increasing boundary constraint ranges - or, that there was still some further parameter optimisation that could have taken place. The range narrows greatly (from 0.59 to 0.1), but again the range is greater than for the previously discussed characteristics suggesting there is still some difficulty presented in dealing with increasing boundary constraint ranges - though tuning characteristics does provide much better solutions, and does allow for the algorithm to perform more consistently against

changing boundary constraint ranges. The pre- and post-tuning results for smoothness coefficient are similar to those of boundary constraint range, though the pre-tuning results were less widely spread. The mean average error improves from 0.260 to 0.010, and range decreases from 0.27 to 0.016. An improvement in both overall average and range smaller than that of boundary constraint range, but greater than number/ratio of local optima.

For changes to the number of dimensions, the average error increases somewhat (from 0.754 to 0.417), and the range shows a very limited improvement (from 0.998 to 0.914). This suggests that although improvements to performance can be made through tuning as dimensionality increases, dimensionality still poses a problem to the BFOA. All values increase in accuracy - tuning is always potentially worthwhile - but there is a much less notable increase in benefit, particularly when compared to other characteristics.

When tested with a paired, two-tailed t-test, the untuned and tuned BFOA datasets were shown to be significantly different using a significance level of 5% for all characteristic values (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). Tuning the parameters using an F-Race in the manner demonstrated here offers statistically different performance for all characteristics, although as described above, despite tuning there is still a wide spread of average errors post-tuning when the number of dimensions is varied, suggesting that tuning does not help the BFOA completely overcome the problem of increasing dimensionality in the same way it appears to help changes in other characteristics.

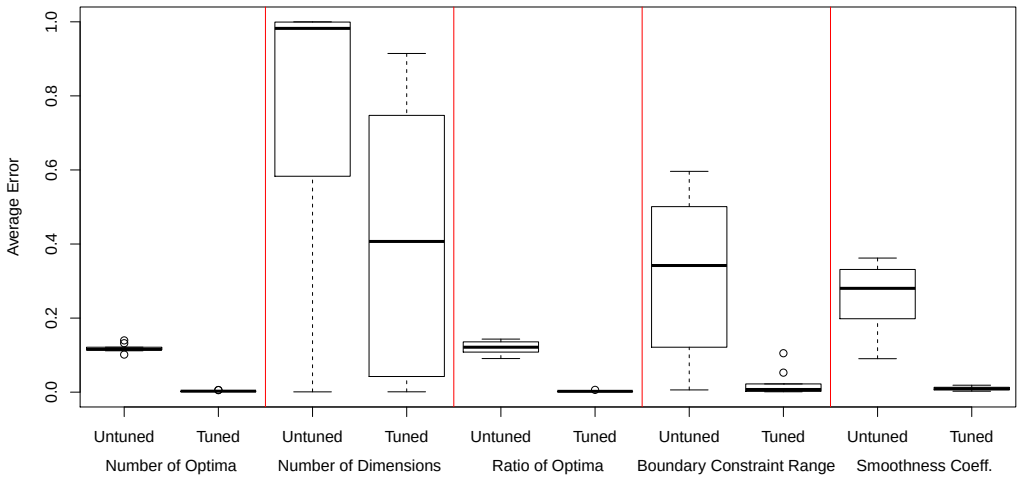


Figure 5.1: Box plots depicting the average errors of the **bacterial foraging optimisation algorithm** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

In terms of the configurations selected by F-Racing, there is little variety in configuration as characteristics change. Across all characteristics, and all values for those characteristics, there are only nine different configurations selected by racing. This suggests that while it is difficult to find a *good* configuration, once it has been found, it is likely good for all *similar* problems. Tuning is vital to the performance of the bacterial foraging optimisation algorithm, but it is possible that by exploring problems using a similar methodology to that demonstrated here, it may be possible to create a 'bank' of promising configurations.

5.4.2 Bees Algorithm

The BA is considered an algorithm in which parameterisation has little effect on the performance of the algorithm (Pham et al., 2006b). As such, only small performance increases after the algorithm parameters have been tuned are expected. It is worth noting that the bees algorithm is one of the best untuned performers in this study, offering weight to the parameter insensitivity argument.

In terms of adjusting the BA to cope with an increasing number of local optima, there are several parameters which have an effect. Parameters such as the number of sites under investigation, the number of bees attributed to those sites, and the differentiation between sites and 'elite' sites are all factors which affect the searching behaviour of the algorithm to allow for greater flexibility as the modality of the problem landscape increases.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.2, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is a clear improvement in the spread of results for the number of dimensions, but all other characteristics do not seem to have improved by a noticeable amount. It should be reiterated that the untuned performance was already very strong, and as such, the BA does not have much room to improve. It would appear that the BA only benefits from tuning when the problem is 'difficult', and in many cases, tuning may not improve performance by a noticeable (or worthwhile) amount. Each characteristic is now examined in detail.

Post-tuning, the BA used the *same* parameter configuration, regardless of the number of local optima present in the landscape. Examining the results, it would appear that tuning has no very little effect on the ability of the algorithm to cope with increasing numbers of local optima, with the range narrowing from 7.67×10^{-4} to 3.25×10^{-6} - an improvement, but on a range that was already very narrow. The average error across all characteristics improves from 1.38×10^{-3} to 8.78×10^{-6}. Again, this is an improvement, but on an error that was already much smaller than many of the algorithms in the study.

A possible explanation as to the selection of only a single set of parameter configurations for all numbers of local optima, is that as long as the number of sites under investigation is greater than the number of optima, the algorithm is capable of dealing with multi-modality. Coupled with the abandonment of 'unpromising' sites, this means

that 'too many' sites is not detrimental to the exploration pattern of the algorithm, and as such, one 'good' parameter configurations suits for a wide range of landscapes regardless of the number of local optima. That is to say, the algorithm naturally has very good local optima avoidance, regardless of the parameter configuration.

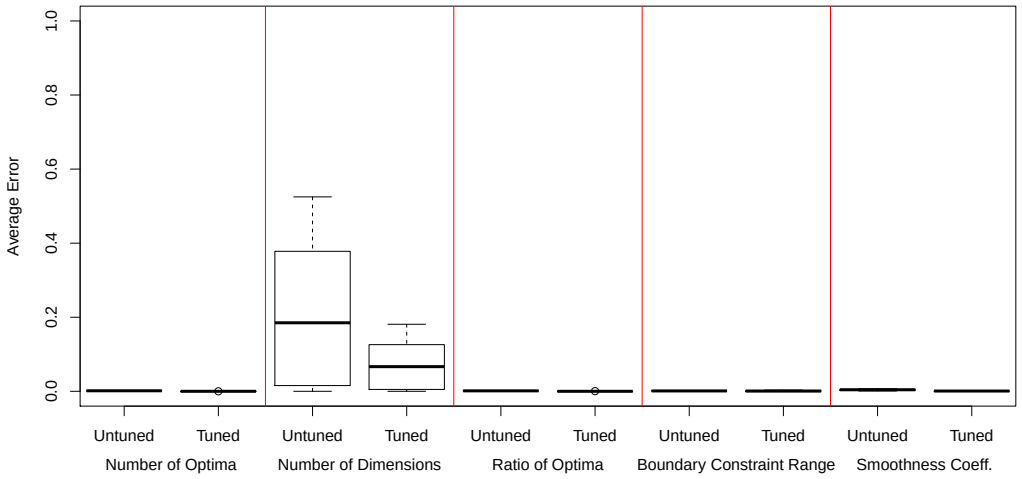


Figure 5.2: Box plots depicting the average errors of the **bees algorithm** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

Similar results occur when increasing the ratio of local optima to the global optimum. Again, as long as the parameter controlling the 'number of sites' under investigation is greater than the modality of the landscape, the bees algorithm is hardly affected by increasing levels of attractiveness regardless of parameter settings. A similar improvement in terms of mean average is noted (from 0.001 to 8.7×10^{-5}), but the range is very similar both pre- and post-tuning (2.3×10^{-4} and 1.9×10^{-4}) suggesting that although performance is improved by tuning, the ability to cope with changes in ratio of local optima (and therefore availability of gradient information) is not affected by the parameter tuning process.

The *patch size* parameter of the BA controls the distance from a site bees are allowed to explore. This is the parameter which would affect the search behaviour of the algorithm as boundary constraint size increases. Unlike the other swarm algorithms, however, the bees algorithm allows for full coverage of any sized search space: *Scout bees* are employed to investigate new random sites, allowing "teleportation" across the landscape, and ensuring coverage across a full landscape regardless of this parameter. As with the number of local optima, the F-Races for the bees algorithm choose the same parameter set for most boundary constraint sizes. Post-tuning, the performance of the bees algorithm actually *decreases* slightly, with a larger average error (from 0.001 to 0.002), increased standard deviation and widened spread of averages (from 4.09×10^{-4} to 3.25×10^{-3}) - suggesting the algorithm can cope slightly less well with changes in boundary constraint size, although the difference is very small. It is possible that

the configurations in this instance have become overfitted to the landscapes used for tuning, and while performance on the landscapes used for racing may have increased, their ability to search generalised landscapes has decreased.

For the ratio of local optima to global optimum, the mean average error shows improvement (from 0.001 to 8.7×10^{-5}), but the spread of results is only improved very slightly (from 5.67×10^{-4} to 4.24×10^{-4}). There is clearly a performance benefit from tuning, however, tuning does not help the BA to cope with the availability of gradient information where the ratio of local optima to global optimum is concerned. Due to the BA's ability to search a landscape thoroughly regardless of the parameter used for patch size, it is likely the case that reasonably 'good' performance can be achieved without tuning - and the performance benefit may come from the parameters optimising for other characteristics of the landscape.

Similarly, for smoothness coefficient (which again controls the availability of gradient information), there is a small improvement in terms of average error (from 0.004 to 0.001), but the range of results only narrows by a very small amount (from 5.81×10^{-3} to 1.13×10^{-3}). The BA does not appear to cope better with changes to available gradient information due to tuning, although small performance improvements can be obtained through tuning.

Dimensionality provides the most significant result in terms of pre-tuning and post-tuning performance for the BA. Fig. 5.3 shows the average error as dimensionality increases for both the untuned and tuned bees algorithm. There is little change to the performance at one to three dimensions - the point where the untuned algorithm is already performing well. As dimensionality increases beyond this the effect of tuning becomes *increasingly* beneficial. It may be the case that there is no increase in performance in other characteristics because these landscapes are simply *not challenging enough* to the bees algorithm to require adjusting the parameters. Fig. 5.2 shows a large decrease in the 'worst' average, from 0.525 to 0.181 - a vast improvement in performance. The spread in averages has also narrowed greatly with tuning, suggesting that tuning has had a large impact on the ability of the bees algorithm to cope with the problem of increasing dimensionality.

When tested with a paired, two-tailed t-test, the untuned and tuned BA datasets were shown to be significantly different using a significance level of 5% for all characteristic values except boundary constraint range (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). As has been argued previously, because the BA exhibits a global search alongside a local search, there does not appear to be any performance change when tuning for differently constrained search spaces. It is entirely possible that a larger range of boundary sizes may require some tuning, or that the 'defaults' for the BA are a wide-ranging enough to suit a wide variety of boundary constraint ranges, but whichever of these is true, it is still apparent at this stage that the BA's performance does not change due to tuning with regard to changing boundary constraint ranges.

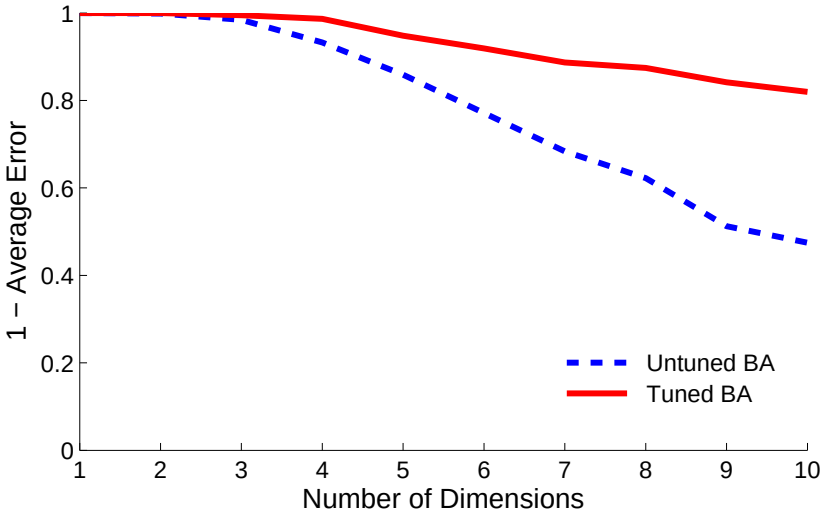


Figure 5.3: The average error of the Bees Algorithm pre- and post-tuning as dimensionality increases.

For the ranges of landscape characteristics the BA has been tested on, it is clear that tuning generally makes little difference to the algorithm's ability to cope with changing landscape characteristics, and generally to the algorithm's performance, as suggested by the original algorithm designer. This could be because the bees algorithm is generally very highly performing and, when tried on more difficult problems, it may be that tuning could still provide significant benefits, similar to those seen when the dimensionality of the problem is high.

5.4.3 Evolution Strategies

ES has the smallest number of parameters of all the algorithms studied here (excepting the baseline algorithm, stochastic hill climbing). The two parameters this form of ES offers are (1) the population size and (2) the number of children. It is suggested that by tweaking these parameters selection pressure is adjusted: That is to say, the *greediness* of the algorithm changes. The parameter configurations obtained through F-Racing are varied, implying that there are some configurations more successful than others. A range of configurations are selected across each characteristic - both in terms of different values for the two parameters, and different selection pressures when the two parameters are combined.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.4, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is either little or no improvement to the average error, or spread of results, for all of the characteristics. This implementation of ES does not appear to benefit from tuning by a noticeable amount in any circumstances. Each characteristic is now examined in detail.

It is unexpected to see that the results of using the tuned parameters show little to no change in performance across all characteristics. There is a small decrease in average error as the number of local optima changes, but the standard deviation is similar for

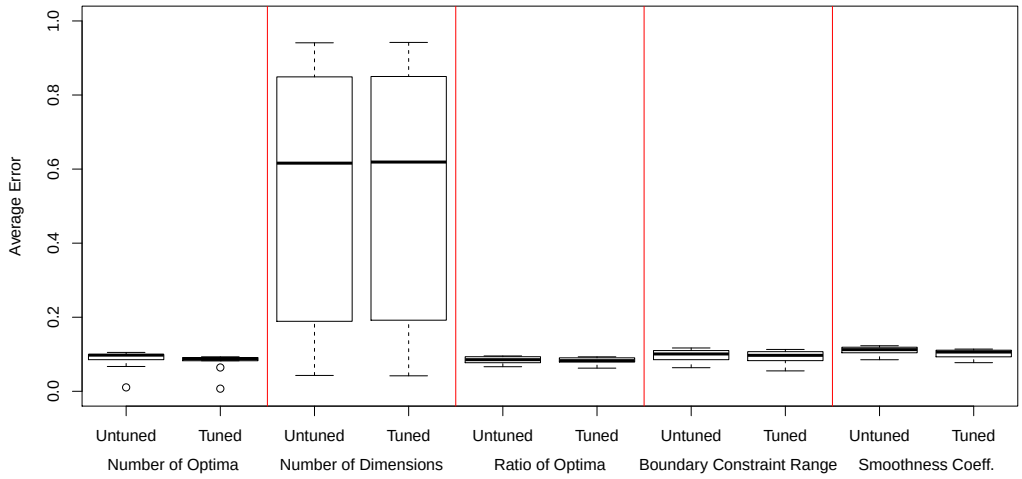


Figure 5.4: Box plots depicting the average errors of **evolution strategies** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

both untuned and tuned, suggesting that while the average error has decreased very slightly, the ability of the algorithm to cope with increasing numbers of local optima is unchanged.

For all other characteristics, there is little change in average error, standard deviation and range across characteristics values (that is to say, the algorithm is no more capable of dealing with changes in these characteristics). This is in-line with the definition of the two parameters the algorithm offers - the selection pressure can only affect the way evolution strategies explores local optima, there is no control over the area that is explored around each point of interest, or any way to encourage the algorithm to rapidly explore an increasingly large search space, for example.

For the number of local optima, there is a small improvement in the mean average error (from 0.085 to 0.078), suggesting a small performance improvement from tuning. The range of averages is very close both pre- and post-tuning (0.0094 and 0.0086 respectively), which suggests that tuning has either had a very small, or negligable, effect on the algorithm's ability to cope with increasing number of local optima. The results *are* statistically significant, so there is some effect from tuning, but whether this effect is worth the extra effort of tuning is questionable.

The results are similar for boundary constraint range and smoothess coefficient also, with an improvement of average error for boundary constraint range from 0.097 to 0.093 and for smoothness coefficient from 0.110 to 0.102. The range broadens slightly for boundary constraint range (from 0.0053 to 0.0058) and narrows slightly for smoothness coefficient (from 0.0038 to 0.0037). As with number of local optima, there is a small performance improvement from tuning, but the changes to range are very small - and it is questionable whether this means that tuning is worthwhile in these instances.

The differences between pre- and post-tuning for dimensions and ratio of local op-

tima, when tested with a paired, two-tailed t-test, were not found to be statistically significant. That is to say, there was no statistically signifnicant difference in this instance between the tuned and untuned results. Indeed this is apparent from the results, with both means changing only slightly (within 0.002), and the ranges exhibiting a similarly small change. This is perhaps unsurprising, as the very basic ES used in this study does not have advanced features or parameters which can capitalise on strategies for better coping with changes to dimensionality and avoidance of local optima.

When tested for significance using a paired, two-tailed t-test with a significance level of 5%, the post-tuning results were found to be significantly different for number of local optima, boundary constraint range and smoothness coefficient, but not significantly different for number of dimensions or ratio of local optima to global optimum (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). Although the results of some of these characteristics are significantly different, the change in mean, and the change in the spread of means, is still very small, and it is important to consider whether a non-substantial improvement, while statistically significant, can be considered a worthwhile improvement in the algorithm's performance.

This form of ES is now considered outdated, and there are many more adaptations of the ES algorithm that offer a greater range of parameters (such as CMA-ES (Hansen and Kern, 2004)). Evolution strategies will give you its best performance with an out-of-the-box parameter configuration, so it is quick to implement, but it is also vital to be aware that there is little you can do to improve this implementation without switching to a more complex variant of ES.

5.4.4 Genetic Algorithm

The performance of the GA generally increases greatly post-tuning, coping significantly better with increasing numbers of local optima, increasing boundary constraint range and an increasing smoothness coefficient. A graphical summary of the average error as characteristics change for the GA can be seen in Fig. 5.5. This particular implementation of a genetic algorithm offers four configurable parameters: (1) Population size, (2) 'Bits' per parameter in the bitstring representation, (3) Crossover rate and (4) Mutation rate. In experiments with a fixed number of objective function calculations, population size affects the number of generations the algorithm reaches before completing. A larger number of bits in a bitstring representation allows more 'precise' solutions to be generated at the expense of a representation which is less affected by mutation. Similarly to BFOA, there are a few configurations which re-occur across different characteristics and different characteristic values. It is probable that once a 'good' configuration has been found for a genetic algorithm, it is extendable to 'similar' landscapes, agreeing with Goldberg (1989)'s suggestion that genetic algorithms are robust problem solvers exhibiting approximately the same performance across a wide range of problems.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.5, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is a reasonable improvement in average error for all characteristics excepting dimensionality, and in some of these cases (boundary constraint range and smoothness coefficient), the spread of results narrows considerably also - suggesting improved robustness to this characteristic. The average error worsens for dimensionality, and the spread of errors broadens, suggesting that the GA has become less capable of dealing with increased dimensionality post-tuning. The GA does benefit from tuning in all cases, but there is a difficulty when the problem is too tough, likely related to the automated tuning methodology selected for this study rather than the GA itself. Each characteristic is now examined in detail.

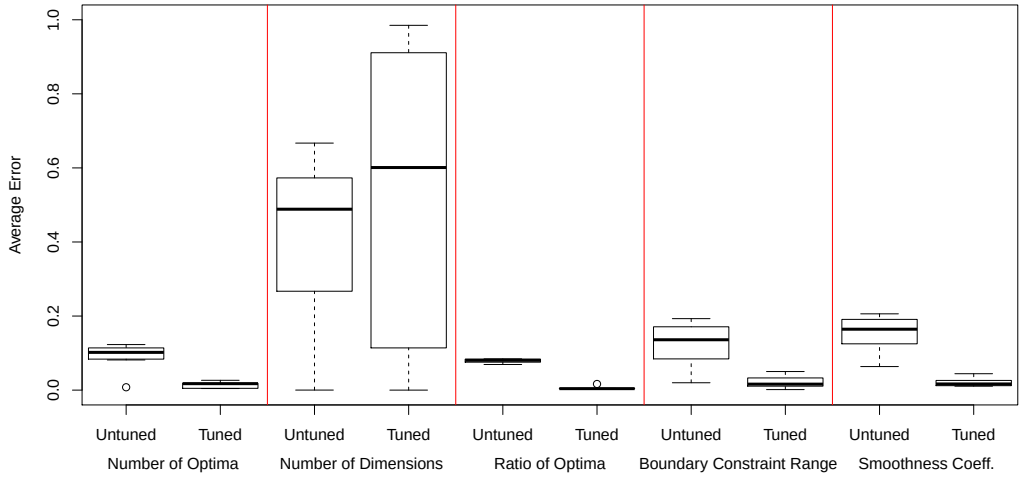


Figure 5.5: Box plots depicting the average errors of the **genetic algorithm** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

When considering the results of changing the number of local optima, post-tuning the mean average error improves from 0.093 to 0.015 - a noticeable improvement. The range of errors also decreases from 0.116 to 0.002, a much narrower range, suggesting that the tuning of parameters has enabled the GA to provide a more robust set of results when the number of local optima changes. Another large improvement is observed in the average mean error for the changing ratio of local optima to global optimum, which improves from 0.079 to 0.007, but the range here decreases less notably, from 0.0015 to 0.0013. This suggests that while there is a performance improvement here from tuning, the ability of the algorithm to cope with changing attractiveness of local optima is not helped by adjusting the parameters and the improvements may be related to better fitting the parameters to other landscape characteristics. It should be noted that the range here was already very narrow - there was little room for improvement, as the GA was already one of the better algorithms in this study for coping with changes to the

ratio of local optima to the global optimum.

Improvements for boundary constraint range and smothness coefficient are also notable. The average mean errors improve from 0.125 to 0.021 and from 0.154 to 0.021 respectively, a similar improvement for both characteristics. Likewise, the range of average errors narrow from 0.17 to 0.048 and from 0.14 to 0.034 respectively. The GA is gaining a performance benefit from tuning both in terms of improvement to the error, and also in terms of improvement to its ability to cope with the changes in these characteristics.

For increasing dimensionality the GA initially shows promising results in terms of tuned performance, with a marked performance increase up to four dimensions. The benefit from tuning rapidly declines, however, until the tuned performance is *worse* than that of the tuned version (see Fig. 5.6). There are two possible explanations for this: The first is that the number of objective calculations used as the termination criteria did not allow the F-Race to gather any meaningful performance data from the configurations. The second explanation is that the test was not conducted on a wide enough range of parameter configurations - although, two of the four parameters have definite ranges (mutation and crossover rates are percentages, thus generation was bounded between zero and one) so this is unlikely. Across the full range of values for dimensionality, tuning decreases the performance of the GA greatly. The average mean error increases from 0.420 to 0.529, and the range of averages increases greatly from 0.67 to 0.97. The negative effect of tuning at greater dimensions severely hinders the overall performance of the GA, and the lessened ability of the GA to cope with varied dimensionality becomes hugely problematic for the algorithm's general performance.

When tested for significance using a paired, two-tailed t-test with a significance level of 5%, the post-tuning results were found to be significantly different for all characteristics excepting dimensionality, where no significance was found (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). The improvements in all other characteristics are both statistically significant and noteworthy, and with the narrowing of the ranges show that the GA is reliant on careful parameter tuning to adapt to varying fitness landscape characteristics. However, this can backfire, as has been demonstrated here with the cases of high dimensionality, where the automated tuning methodology was unable to find a suitable parameter configuration, resulting in increased average errors.

Genetic algorithms offer additional levels of customisation in the form of component selection. Design decisions such as selecting the most appropriate problem representation, crossover and mutation strategies, and selection strategies also have to be made. Only a single combination of these features has been examined in this study (a genetic algorithm with a binary tournament selection process, bitstring problem representation, one-point crossover and "bit flip" mutation). This methodology could be used to examine the effect of different components on a genetic algorithm, which may further highlight the importance of design decisions in relation to characteristics of fitness

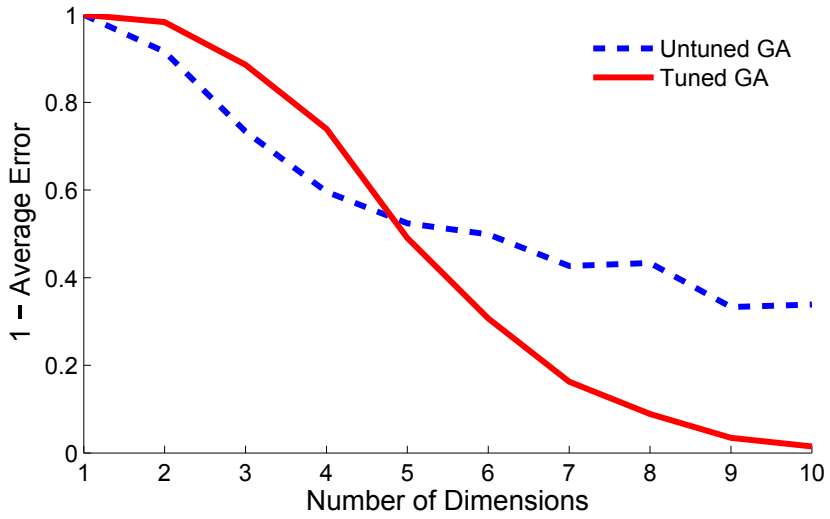


Figure 5.6: The average error of the Genetic Algorithm pre-tuning and post-tuning as dimensionality increases.

landscapes.

5.4.5 Harmony Search

The four parameters of HS all control different aspects of the search strategy. Memory size dictates how many promising solutions can be stored - effectively, how many potential sites of interest are retained by the algorithm. Consideration rate and adjustment rate control how new solutions are generated. The consideration rate is the percentage chance that a solution based on one in memory will be generated (conversely, *1-consideration rate* is the chance a random solution is generated instead). The adjustment rate is then the percentage chance that the randomly chosen solution from memory will be adjusted. In cases where a solution is adjusted the fourth parameter, which controls the maximum range from which solutions can be selected, is used. If the adjustment does not occur, the considered solution potentially occupies an additional slot in the memory - thus increasing the chance that this solution may be chosen for consideration again. The interplay between these parameters is crucial, and it is somewhat hard to see how consideration rate and adjustment rate can directly affect the search strategy - unlike memory size and range, which are more obvious.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.7, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is an improvement in average error for the number of dimensions, and the boundary constraint range, coupled with a shortening of the range of errors also. In these cases, tuning has improved the results of the HS. For the other characteristics, results are less impressive, with either little or no obvious improvement. The HS does benefit, but as with the BA, with untuned results being very good, the problem needs to be sufficiently 'difficult' to justify the tuning process. When the problem proves challenging, e.g. at high dimensionality or large boundary constraint ranges, the HS is improved considerably by tuning. Each

characteristic is now examined in detail.

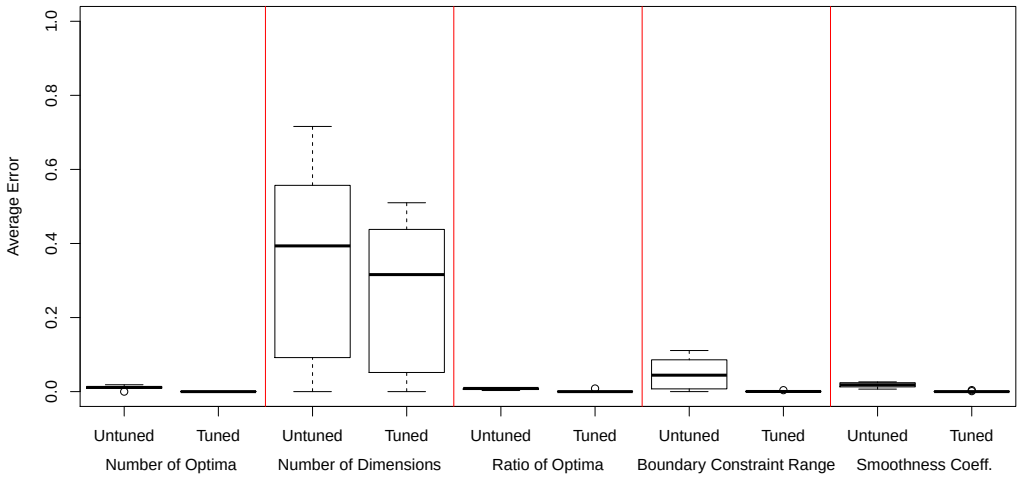


Figure 5.7: Box plots depicting the average errors of the **harmony search** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

HS, like the BA, offers some of the lowest 'out of the box' average errors in this study. For most characteristics, there is little room for a performance increase post-tuning. Boundary constraint range proved the second most challenging characteristic to harmony search pre-tuning, but post-tuning shows improved performance with a substantial improvement in average mean error from 0.048 to 0.001. The range of errors also narrows, from 0.111 to 3.97×10^{-3} - which suggests that HS improves not only in terms of performance, but also in terms of its ability to cope with the change in this characteristic. The range values in all the configurations selected by F-Racing are much smaller than those in the 'out of the box' values, and this contributes significantly to the performance improvement where boundary constraint ranges are increasing. The consideration rate also decreases almost linearly as size increases - effectively, more random solutions are used instead of a reliance on the 'memory'. These random solutions allow the solution pool to *jump* from one position in the search space to another, encouraging a wider search space, explaining the significant improvement as boundary constraint range increases.

Dimensionality also shows some minor improvement in the tuned parameter performance of harmony search in terms of both average error and ability to cope with the increasing characteristic, with an improvement in average mean error from 0.364 to 0.263, and a narrowing of the range of errors from 0.716 to 0.510. High dimensionality problems (seven and above) have a much higher consideration rate than the successful configurations for lower dimensionality, suggesting that a focus on exploitation rather than exploration is beneficial to the harmony search when dimensionality is high. This is the opposite case of what happens with boundary constraint range as discussed above.

Number of local optima, ratio of local optima to global optimum and smoothness coefficient were all characteristics with which the HS already coped very well - providing both good quality solutions, within a very small range, across all characteristic values. Post-tuning, there is a significant improvement in the HS's performance when the number of local optima changes, although as mentioned previously, the pre-tuning performance was already better than many other algorithms in this study. The average mean error decreases from 1.12×10^{-2} to 2.17×10^{-5}, with the range narrowing from 1.86×10^{-2} to 7.34×10^{-5}. Again, while the HS already offered fairly consistent performance as the number of local optima changed, this is improved upon post-tuning.

Much smaller improvements are observed for ratio of local optima to global optimum and smoothness coefficient, with the average mean error improving from 7.14×10^{-3} to 1.72×10^{-3} and 1.79×10^{-2} to 6.19×10^{-4} respectively. The range for ratio of local optima shows little change post-tuning, suggesting that the HS is no more capable of escaping increasingly attractive local optima due to parameter changes. However, the range of average errors for smoothness coefficient shortens from 1.95×10^{-2} to 4.13×10^{-3}, indicating some improvement in ability to cope with availability of gradient information.

When tested for significance using a paired, two-tailed t-test with a significance level of 5%, the post-tuning results for all characteristics were found to be significantly different to the pre-tuning results (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). However, as with the BA, the performance of the HS is very strong pre-tuning, and the improvements made by tuning are most noticeable when the problem is 'difficult'.

5.4.6 Particle Swarm Optimisation

PSO in this form has four parameters. These parameters control the population size, the maximum velocity of a particle and two parameters which control the bias towards the particle best solution and the bias towards the global best solution. With these parameters, it is possible to control the coverage of a search space (the number of particles), enforce a large search area or a small search area for each particle (the maximum velocity) and through tweaking of the local and global best solution bias, control the capability of the algorithm to converge on a single solution or explore several areas of interest (optima avoidance).

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.8, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is an improvement in average error as the number of dimensions increases, as the boundary constraint range increases, and a smaller yet noticeable improvement when the smoothness coefficient changes. There is not an obvious improvement for the number, or ratio, of local optima. As with the BA and HS, the PSO does appear to benefit from tuning, but only when the problem is 'difficult' with the default parameters. Each characteristic is now examined in detail.

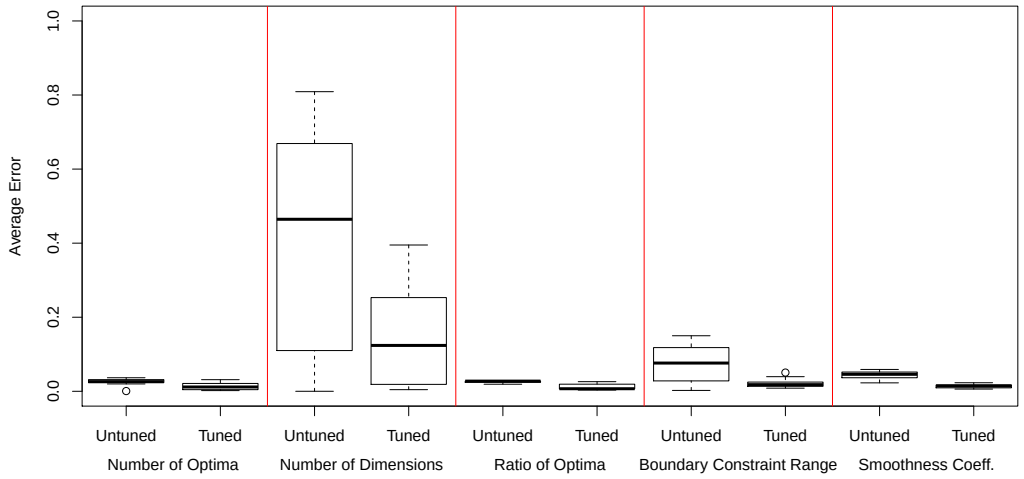


Figure 5.8: Box plots depicting the average errors of **particle swarm optimisation** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

A small decrease in average error is shown as local optima increases (from 0.025 to 0.014) - and the range narrows only very slightly, from 3.59×10^{-2} to 2.95×10^{-2}, indicating that the algorithm is no more capable (or only slightly more capable) of dealing with increasing numbers of local optima post-tuning. The results for ratio of local optima to global optimum mirror those of number of local optima in terms of average error, with an improvement from 0.025 to 0.016. The range, however, for this characteristic actually widens very slightly, from 9.31×10^{-3} to 2.27×10^{-2}. Better performance has been obtained from the PSO when the ratio of local optima is high, which either suggests that the algorithm has optimised its parameters to better take advantage of the local optima, or, has better optimised its ability to avoid local optima when they are more attractive. Smoothness coefficient results also show a small improvement post-tuning. There is a small increase in the average error (from 4.32×10^{-2} to 1.41×10^{-2}), and the range shortens slightly also, from 3.62×10^{-2} to 1.71×10^{-2}.

As there are techniques in PSO to avoid local optima, and parameters to control these, a stronger improvement in error and robustness for these two characteristics may have been expected. As with other local optima avoidance strategies in previously discussed algorithms, however, the presence of these strategies, and not the specific parameterisation, is likely enough to encourage the algorithm to escape local optima, meaning that the actual parameters chosen are not as important as some of the other parameters available in the configuration of the algorithm.

PSO starts to show a much larger improvement with tuning where changes to boundary constraint range is concerned. The increase to average error here is much greater than that of the three previously mentioned characteristics (from 7.6×10^{-2} to 2.19×10^{-2}), and the spread of results also makes a dramatic decrease from 0.148 to

4.21×10^{-2}. It is perhaps not surprising, considering PSO's parameter which so directly controls exploration (maximum velocity), that there is a great improvement in the algorithm's ability to adapt to changing boundary constraint ranges post-tuning, and indeed, six different configurations are selected for the varying boundary constraint ranges - suggesting that having a customised parameter configuration for specific boundary constraint ranges does make a considerable difference to the performance of PSO.

Performance of PSO most greatly improves in terms of dimensionality post-tuning, in terms of both average error (improving from 0.420 to 0.157) and ability to cope with the changing characteristic values (with the range narrowing from 0.809 to 0.39 - mostly due to greatly improved performance at higher dimensions). The F-Races for PSO selected the same configuration for nearly all values of dimensionality, implying that there is no specific parameter that needs adjusting to cope with the increase in dimensionality, but selecting a configuration which provides good exploration allows PSO to perform well as the search space increases exponentially. Dimensionality is still the characteristic where PSO offers the poorest performance, and in which the results show the algorithm to be the least robust, but this not unusual, as this is the same for all other algorithms.

This trend continues across all characteristics, with F-Races selecting the same configurations often, regardless of characteristic values. As with the other swarming algorithms, it is possible that once a good configuration has been found, it is able to deal with a wide range of problems of a similar nature, regardless of the specific characteristics. The configurations selected are all varied in their parameters, and it is unexpected to see that there is no pattern to maximum velocity as boundary constraint range increases. This is likely that because maximum velocity is a maximum, and there are particles with randomly generated velocities below the maximum, this parameter is less significant than it may initially appear. It would be interesting to consider the effect of having a *minimum* velocity on the increase in boundary constraint range, although this would also severely hamper exploitation.

When tested with a paired, two-tailed t-test, the post-tuning results were found to be statistically significantly different at a significance level of 5% for all characteristics, excepting number of local optima - which, as discussed above, showed virtually no improvements (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). The PSO does contain does parameters which control local optima avoidance (in the form of weightings which control reliance on a particle's own best solution versus the population's best solution), so there was the potential for tuning to make a difference here. As previously suggested, the likely explanation for a lack of improvement post-tuning, especially given the strong initial results, is that the 'default' parameters for PSO already offer strong local optima avoidance, and that tweaking these particular parameters does not make a significant difference.

5.4.7 Stochastic Hill Climbing

With only a single parameter - the range at which new solutions are generated - the stochastic hill-climbing algorithm does not offer a large amount of customisation where parameters are concerned. This one parameter is directly linked to the search pattern and nothing else, and as there are no other parameters there is no interplay between parameters to consider, so arguably stochastic hill-climbing should prove the easiest algorithm to tune.

Box plots depicting the average errors across the characteristic ranges are shown in Fig. 5.9, allowing for a thorough examination of the spread of average errors pre- and post-tuning. Examining the errors in this way shows that there is an improvement in average error for all characteristics except dimensionality (where performance actually worsens, similar to the GA). The spread of results narrows slightly for number of optima, ratio of optima and smoothness coefficient, and there is a bigger improvement in the spread of results for boundary constraint range. As with the GA, SHC shows that tuning is always a benefit (excepting the cases where tuning has failed, once again likely attributed to the tuning technique rather than the algorithm itself).

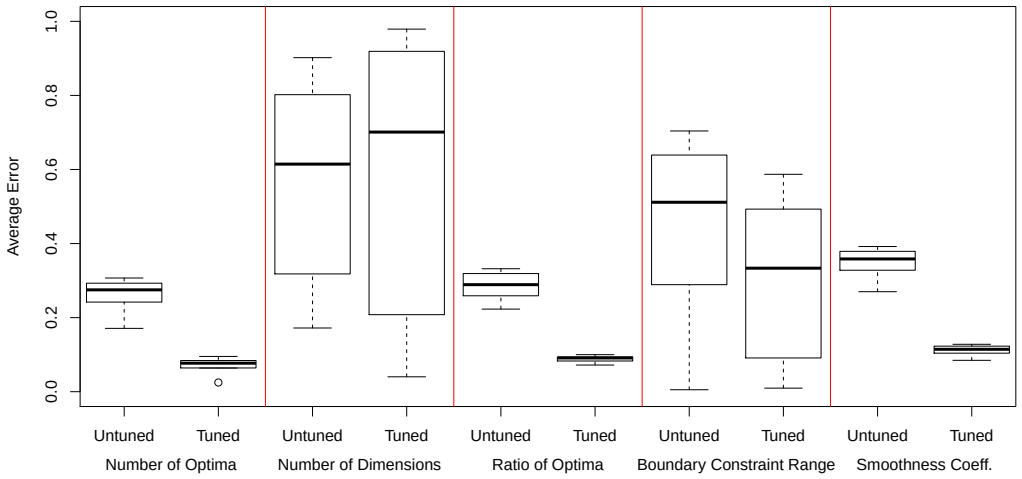


Figure 5.9: Box plots depicting the average errors of **stochastic hill-climbing** across ranges of characteristic values, for each selected characteristic. A narrower spread of errors indicates that the algorithm is more robust to this characteristic, i.e. the performance remains consistent regardless of the changes to the fitness landscape.

As shown, all characteristics, barring dimensionality, show an improvement post-tuning, which is counter-intuitive given the number and nature of the parameters of the algorithm. As the neighbourhood size is intrinsically linked with the range from which new solutions are generated it is of no surprise that performance post-tuning is affected when boundary constraint range changes. Examining the configurations determined through F-Racing, there is a clear correlation: As the boundary constraint range increases, so does the neighbourhood size (the neighbourhood size is consistently 50%-60% of the boundary constraint range). As the number of objective function calculations is limited, despite having a larger neighbourhood size, the ability of the

algorithm to effectively explore larger environments is still limited, hence the average error not decreasing as much as may be expected (from 0.446 to 0.305), but the ability of the algorithm to deal with increasing search space sizes does also improve slightly (the range narrows from 0.699 to 0.578).

Dimensionality shows poorer performance post-tuning with SHC, similar to that observed with the GA. The average error remains roughly consistent post-tuning (with a pre-tuning error of 0.577 and a post-tuning error of 0.589), but the range broadens quite vastly from 0.73 to 0.94. As with the GA, better performance is obtained when the landscape comprises of fewer dimensions, but this is offset by poorer performance when the dimensionality is high. This could, once again, be a case of the F-Race failing to find a satisfactory parameter configuration, and producing less desirable configurations/configurations which are not well suited for generalised landscapes.

Number of local optima, ratio of local optima to global optimum and smoothness coefficient all show very similar improvements in terms of both increased average error and decreased spread of results. Taking the parameters of the algorithm into account, it is surprising to observe clear improvements in both average error and error ranges for the characteristics other than boundary constraint range. The number of local optima demonstrate a large increase in performance and a greater ability to cope with more optima (a reduced standard deviation). The parameter configurations selected for the number of local optima, the ratio of local optima *and* the smoothness all have a neighbourhood size of around 50% the search space size. Performance improvement for all of these characteristics could be attributed to the algorithm having configured itself properly for the search space size used as a default for all other characteristics, rather than tuning itself to perform most significantly with the characteristic in question.

When tested with a paired, two-tailed t-test, all characteristics were found to be significantly different at a significance level of 5% post-tuning, excepting number of dimensions (H_0 = Tuning the parameters of the algorithm using an F-Race has no effect on an algorithm's performance, H_A = Tuning the parameters of the algortihm using an F-Race has an effect on an algorithm's performance). As previously discussed, it is somewhat surprising to see tuning making a difference to the SHC algorithm, given the lack of parameters to tune. There is a possibility that the performance improvements have been obtained from optimising the algorithm's search strategy to the default boundary constraint range (i.e. using a range parameter of approximately 15 instead of 10, to better fit the default boundary constraint range of 30).

While tuning makes an improvement to performance of SHC, the noticeable trend in the parameterisation, and the simplified set of parameters perhaps means that an automated tuning methodology may be more than is necessary to obtain maximum performance from the algorithm.

5.5 Summary

In this Chapter, the experiment from Chapter 4 has been developed further. A significant limitation of the previous work pertained to the use of 'out of the box' parameter configurations for the algorithms, which is unlike the way algorithms are used in the real world. By adding an automated parameter tuning component (F-Racing) to the methodology, the effect of tuning on different algorithms has been studied, contributing significantly to the debate on when it is beneficial to tune algorithms.

It is noted that algorithms broadly fall into three categories with regard to their response to tuning: (1) Algorithms which do not benefit from tuning (ES), (2) Algorithms which benefit from tuning when the problem is "difficult" enough to require tuning (i.e. when untuned performance is poor) (BA, HS, PSO) and (3) Algorithms which always benefit from tuning (BFOA, GA, SHC). Dimensionality often offers the most significant improvement post-tuning in algorithms that have parameters relevant to increasing the *breadth* of search space (swarming algorithms are significantly better here than evolutionary algorithms).

The answer to the question 'To what extent does tuning alter the performance profile of an algorithm with regard to each of the landscape characteristics defined?' therefore depends on both the algorithm and characteristic in question. While there are some algorithms for which tuning has no effect, at least in this study, there are many more where there is an effect - and that effect varies, depending on the characteristic, and the combination of characteristic values. Developing a portfolio of the algorithm's performance, both pre- and post-tuning in this manner, is a helpful strategy for describing the behaviour, strengths and weaknesses of an algorithm, and for exploring the way in which an algorithm is capable of dealing with challenging landscapes. With a performance profile such as those described here, a practitioner can identify what the particular strengths, and weaknesses, of an algorithm are, and whether or not it requires tuning in a specific implementation.

Another way of looking at the results is to consider how tuning works on a per-characteristic basis. Box plots depicting the pre- and post-tuning results of each characteristic, by algorithm, are shown in Figs. 5.10 and 5.11. Examining the results in this way shows that despite the algorithms having unique and varied performance portfolios, with varied reactions to the tuning process, there is some consistency with regard to characteristics. These findings mirror those found in Chapter 4, for example, dimensionality is clearly the most difficult both pre- and post-tuning, with tuning offering the strongest improvements as dimensionality improves, but algorithms still suffering from the poorest performance despite this.

Number of local optima and ratio of local optima are also fairly consistent across algorithms. The likelihood here is that both of these characteristics control the landscape's attractiveness of local optima, and as such, if an algorithm improves post-tuning when there are more local optima present, it likely also improves if these local optima are more attractive.

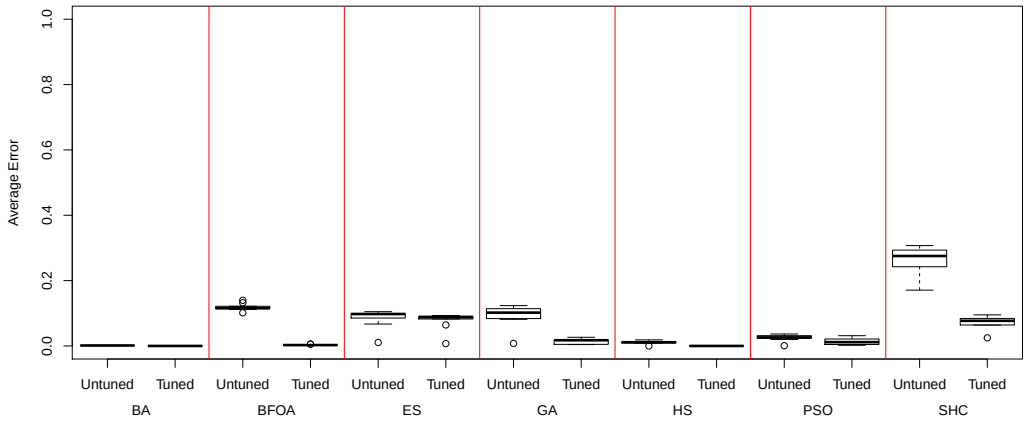
The methodology presented here is computationally inexpensive, which makes it ideal for use as a 'benchmark' process for better defining novel algorithms, with the potential for discussing performance of algorithms both pre- and post-tuning. Additionally, it offers the possibly of relating this to when it is appropriate to tune based on estimated landscape characteristics. To further this work, it would be useful to investigate further which *features* of specific algorithms make them more or less amenable to tuning, and positions algorithms to better cope with difficult characteristics. It is expected that to do so would follow a similar methodology: By 'flagging' certain features off and on, and comparing the performance with relation to landscape characteristics, an algorithm designer can highlight the features of algorithms necessary for good performance, and potentially transfer these features from existing algorithms to new algorithms to cover their weaknesses to certain landscape characteristics.

It is of note that this methodology is applicable to *any* optimisation algorithm, and further studies could continue to add algorithms to form a "complete" picture of algorithm performance, both with untuned parameters and tuned parameters, across the nature-inspired optimisation field.

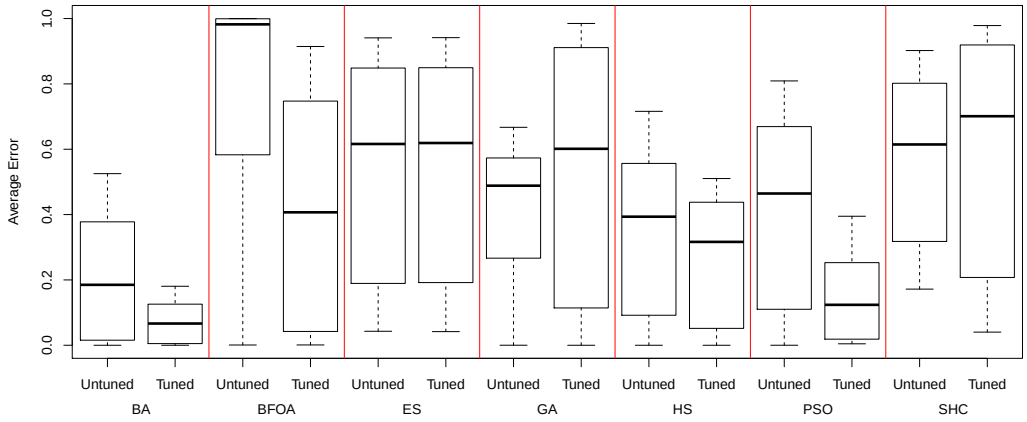
While the work in this Chapter presents the findings using one automated parameter tuning methodology, there are several more that exist in the literature. It could be argued, for example, that the work presented here demonstrates how amenable algorithms are *to the F-Racing process* rather than how they are to the process of tuning more generally. To further this study, it would be beneficial to look at these other tuning methodologies and compare the results; The benefits here would be twofold - such studies would firstly offer additional insights into the performance of the algorithms post-tuning, with more configurations tested, and secondly, different automated tuning methodologies could offer advantages and disadvantages under different problem configurations. It may be the case that certain automated parameter tuning methodologies work well at extremes, i.e. when there is little differentiation between performance data from algorithm runs, and others when there is greater variation. If this is the case, a methodology presented such as this could also be used to identify the strengths and weaknesses of parameter tuning methodologies, in addition to algorithms for optimisation themselves.

The work so far, from Chapter 4 and this Chapter, has focused on methods of trying to offer practitioners more information about nature-inspired algorithms to aid with the algorithm selection problem. One of the problems that remains is that this information is still difficult to use; While information has been presented as clearly as possible, it is still not clear which algorithm is "best" to use in a given circumstance. For example, if you have a problem with high dimensionality, but with low modality, it is now known that the BA performs well against high dimensional problems, but so do GAs, and then adding in the extra factor of number of local optima further complicates matters. To aid in the decision making process further, it would be useful to *automate* the process of algorithm selection. Having identified suitable characteristics for "dividing" algorithm performance, these could be used as inputs for a machine learning technique, and it is

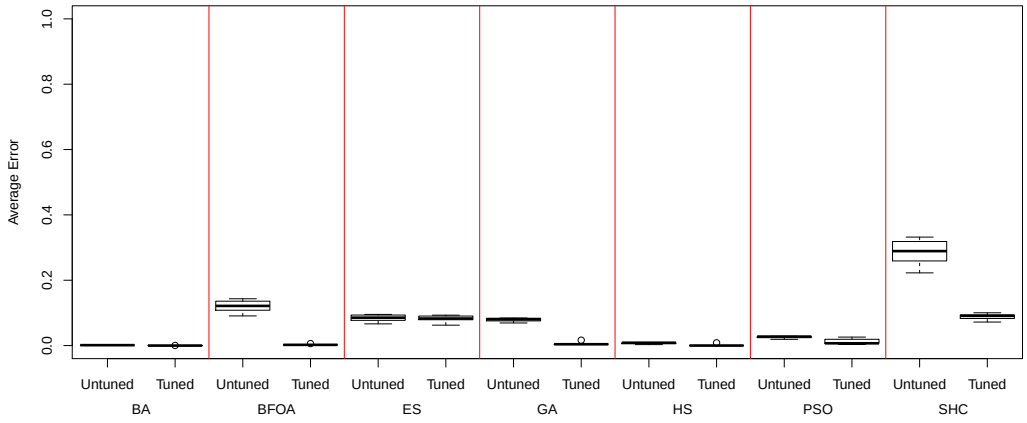
this approach that is trialled in the following Chapter.



(a) Number of local optima.

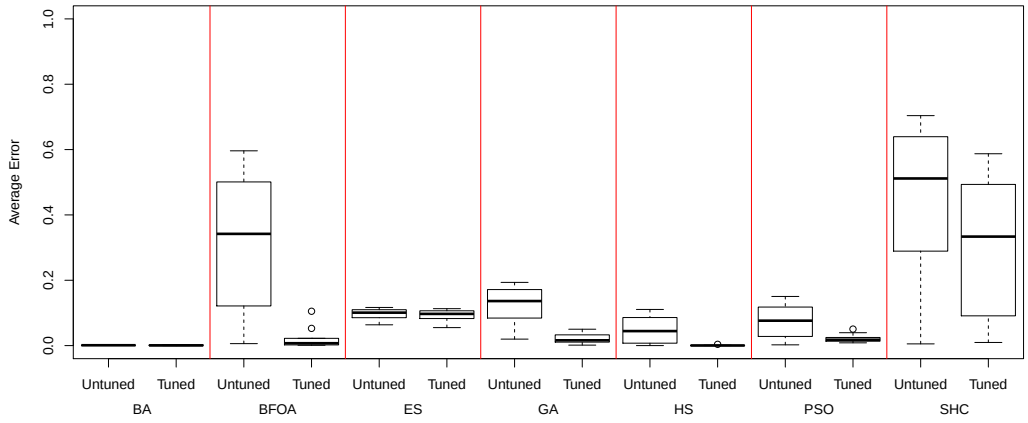


(b) Number of dimensions.

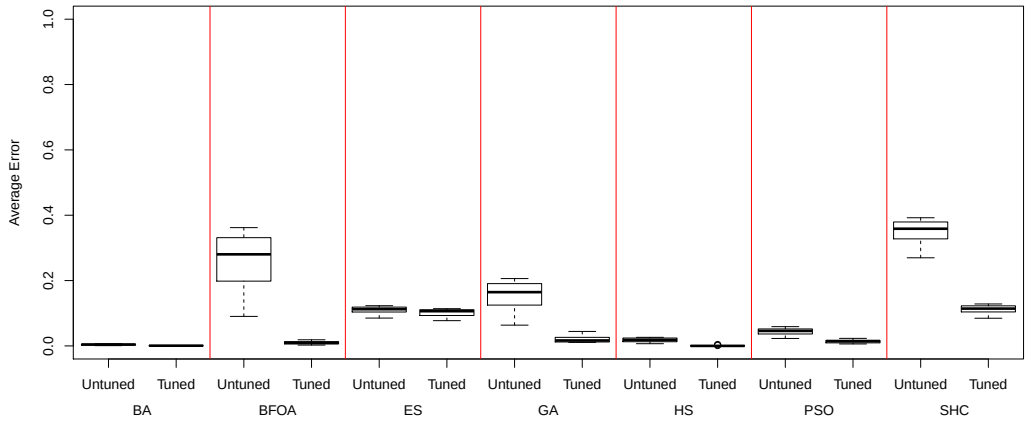


(c) Ratio of local optima to global optimum.

Figure 5.10: Box plots depicting the average error of all algorithms across the complete range of characteristic values for all characteristics (number of local optima, number of dimensions and ratio of local optima to global optimum).



(a) Boundary constraint range.



(b) Smoothness coefficient.

Figure 5.11: Box plots depicting the average error of all algorithms across the complete range of characteristic values for all characteristics (boundary constraint range and smoothness coefficient).

Chapter 6

Using Landscape Characteristics as a Performance Predictor

6.1 Introduction

The *algorithm selection problem* (Rice, 1976; Smith-Miles, 2009) involves finding the "best" algorithm to solve a specific problem for a given case. One approach to this problem may be to test different algorithms on public benchmark data sets, but, such experimental studies are limited in that they generally only consider a relatively small number of algorithms and/or problem instances. The No Free Lunch Theorem (NFLT) (Wolpert and Macready, 1997) suggests that no single algorithm can out-perform another (including random search) on *all* types of problem (Christensen and Oppacher, 2001). The natural implication of this is that benchmark-based comparisons have limited utility, as there may well exist problem instances on which algorithms that are found to be "superior" perform less well (Smith-Miles and Lopes, 2012). It is natural, therefore, to focus on using the *features of the problem* under consideration to inform the choice of algorithm. There now exists a wide and diverse range of algorithms and methods for computationally hard problems, and finding the "best fit" between the characteristics of an algorithm and those of a particular problem is, itself, a difficult task (Kotthoff et al., 2012).

As an ever increasing number of nature-inspired algorithms, and variants of these algorithms, are described, the process of algorithm selection becomes increasingly difficult. The field of *hyper-heuristics*[1] has evolved alongside that of heuristics and meta-heuristics[2] to combat this problem, but as more novel algorithms are proposed new hyper-heuristic techniques need to be developed in an attempt to minimise the problems posed by the process of algorithm selection (Burke et al., 2003, 2010; Özcan et al., 2008).

In the previous Chapters, and associated publications, it has been shown that the performance of various algorithms can be related to the *characteristics* of the fitness landscape of the problem to be solved (Crossley et al., 2013a,b; Pham and Castellani,

[1] Defined here as the process of selecting meta-heuristics.
[2] Defined here as the process of selecting heuristics.

2013). This provides insight into the strengths and weaknesses of an algorithm with regard to certain values of landscape characteristics, but there is still considerable difficulty in interpreting this information when considering which algorithm to select for a given problem. This leads to the third, and final, of the questions posed at the outset: *To what extent can algorithm performance by predicted by using the defined landscape characteristics as inputs to a classification algorithm?* This Chapter addresses that question by trialling two classification algorithms which automatically "rank" heuristic choices by making assumptions on algorithm performance based on landscape characteristics of a given problem, using the data sets generated using the methodology proposed in Chapter 3.

Two systems for predicting algorithm performance were developed, one using artificial neural networks and one using random forests (Breiman, 2001). These two learning algorithms were trained using data sets generated across a range of landscape characteristics. The results analyse whether data sets with *average error* or *expected ranking* as the target outputs produce the best prediction performance for each learning algorithm.

The remainder of this Chapter is organised as follows: There is first an exploration of the background of fitness landscape analysis for the algorithm selection problem in Section 6.2, followed by a description of the approach in Section 6.3, before the experimental findings are presented in Section 6.4. Finally, there is a concluding discussion of the implications of the results, and suggestions for further work in Section 6.5.

6.2 Background

Finding a way to reliably relate fitness landscape analysis to problem characteristics, and then to algorithm performance, is an essential component of solving the algorithm selection problem. One of the originators of the idea that this might be possible for a wide range of algorithms using characteristics of fitness landscapes is Merz (2001). Merz uses a variety of different techniques for describing fitness landscapes to classify problems. In Merz and Freisleben (2000), existing measures of 'hardness' are used to classify landscapes and attempts are made to relate this to algorithm performance leading to the selection of components of an algorithm (in this case, recombination and mutation operators). In later work, Merz proposes new measures of landscape hardness designed to help relate fitness landscapes to algorithm performance (Merz, 2004). The observation is made, once again, that these measures provide valuable insight into the behaviour of algorithms on different *kinds* of landscapes, and specific features of landscapes (such as the size of the basins of attractiveness of local optima) are important in relating fitness landscape analysis to performance. The techniques proposed in this work are said to improve on previous techniques (such as the fitness distance correlation coefficient) which are not feasible for real-world problems.

In Morgan and Gallagher (2012a), the authors also acknowledge the difficulty of understanding the relationship between problems and algorithms, and turn their attention

to continuous optimisation problems unlike previous efforts which have mainly focused on discrete optimisation proposing a technique for capturing information about problem structure. With Morgan and Gallagher (2012b), the authors once again investigate continuous optimisation and the effect ridge structures in a landscape have on optimisers, proposing a methodology for studying the behaviour of optimisation algorithms. The authors acknowledge that more precisely categorising the relationship between landscape structure and algorithm behaviour is the long-term goal beyond this work, and that broader experimentation (including differing performance measures) may lead to this.

Working with particle swarm optimisation, Malan and Engelbrecht (2013) suggest that previous measures of hardness are not generally useful for relating fitness analysis to algorithm performance for a variety of reasons. To summarise, these reasons include (1) Some require knowledge of the global optima which is not always known, (2) Many assume the problem is discrete which is not always the case, and, (3) Many are as computationally intensive as solving the problem. The authors of this work propose three new measures, specifically with continuous optimisation in mind, which measure ruggedness, predict the presence of funnels and provide an estimated measure of the fitness gradient. These novel techniques were tested on benchmark functions, and an attempt to relate performance of PSO to these measures was made. The conclusion, as perhaps can be expected, states that each measure gives *some* information about performance, and could be used as a part-predictor, but none is useful as a complete predictor. This sentiment is echoed in future work, too. In Malan and Engelbrecht (2014b), an opposite approach is taken to the standard - instead of trying to find *good* algorithms, fitness landscape analysis is used to try and find where the PSO algorithm *fails* to optimise. This time, several traditional discrete optimisation landscape analysis techniques are adapted for use in the continuous domain and, once again, the conclusion is that these techniques give partial information individually, but in order to build up a complete picture of algorithm performance (or, rather, failure), several measures need to be used. This is further reinforced in Malan and Engelbrecht (2014a), where similar results are obtained once again, emphasising the need for multiple measures of different fitness landscape features.

The process of mapping a 'real-world' problem to a fitness landscape (and therefore an algorithm) is a complex and intractable task, even with the series of landscape analysis measures the above authors have used to relate fitness landscapes to algorithm performance. This is observed in all of the work described above, and also in the work of Sun et al. (2014), who tackle the problem of mapping fitness landscape analysis measures to problem characteristics as a way of easing this difficulty. This collection of metrics demonstrates that there are several ways to determine different problem characteristics in terms of fitness landscape analysis, making the problem somewhat more tricky, and, this work posits itself as a way of helping algorithm designers looking to solve the problem of algorithm selection choose which fitness landscape analysis metric to use.

Bischl et al. (2012) demonstrate some success in using landscape analysis for automated algorithm selection, based on exploratory analysis. Using large data sets obtained from workshops, support-vector regression is used as the learning methodology to choose between a number of algorithms on a set of functions. The authors acknowledge that there is some difficult in the experimental set-up: The functions used are very different from each other (making accurately testing the learning algorithm's learning capabilities challenging). Based on low-level features (such as an estimation of the number of peaks in the distribution of function values) the authors state they can predict the optimal, or close to optimal, candidate from a small portfolio of algorithms for a given function. They do observe that they have poor worst-case scenario performance and express a desire for more test functions.

6.3 Methodology

The fundamental question this Chapter addresses is this: is it possible to use fitness landscape characteristics in order to predict the performance of a set of algorithms? As Kotthoff *et al.* point out, "While there has been some small-scale work to compare the performance of different machine learning algorithms, there has been no comparison of the machine learning methodologies available for algorithm selection and large-scale evaluation of their performance to date" (Kotthoff et al., 2012). In this paper, the authors evaluate a number of machine learning algorithms (and a large number of machine learning methods for algorithm *selection*) on a number of *discrete optimisation* public data sets, to give a broad overview. In this Chapter, two of the algorithm selection methods they use (*random forests* and *neural network*) are selected for investigation into their prediction performance across different general landscape characteristics. This approach extends that of Muñoz et al. (2012), who use a neural net-based regression approach to rank a number of different *configurations* of a base algorithm.

To investigate the feasibility of fitness landscape characteristics as a predictor of algorithm performance, a methodology is proposed that uses two different machine learning techniques to predict the "performance rankings" of seven different optimisation algorithms. The methodology is as follows: (1) select a number of nature-inspired algorithms, and obtain consistent source code for their implementations; (2) generate *training data* for each algorithm, describing the performance of each algorithm across a range of characteristics; (3) train two learning algorithms (neural net and random forest) using the training data, and then test the predictive capabilities of each.

The predictors are tested using ten-fold cross-validation (Kohavi, 1995), and the learning algorithms are compared against each other. In order to get an idea for how best to prepare the data sets, the learning algorithms are trained using the data in its raw state (that is, with the average error used as the target output) and also with the data *pre-ranked* (that is, with the expected rank as the target output). In the following sections, the methodology is in described in more detail.

Table 6.1: Ranges of characteristics used with the Max-Set of Gaussians landscape generator to generate the training data for the predictors.

Characteristic	Start	Step	End
Number of Optima	1	100	501
Ratio of Local Optima	0.1	0.4	0.9
Dimensions	2	0	2
Boundary Constraint Range	1	50	1151

6.3.1 Training Data Generation

The methodology from Chapter 3 forms the basis of the process for generating training data. The same set of algorithms are used, the same fitness landscape generator is used, and the same landscape characteristics are used. However, in order to ensure a sufficient amount of data for training the learning algorithms, the range of characteristics used has been adjusted for this particular experiment. The ranges used for training data are shown in Table 6.1. Note that in this case, there are no defaults. Instead, a *complete* set of training data (that is to say, all possible combinations of characteristics within the ranges) are generated. Each algorithm was executed on each possible combination **64** times (each a different landscape; 64 chosen as this is the number of simultaneous threads capable of execution on the cluster used for data set generation in this particular experiment), resulting in 29,440 training elements per algorithm. Note that smoothness has been excluded from the range table (and also was not considered as a characteristic in this experiment), as it is not possible to make reasonable estimates about the smoothness coefficient of a 'real-world' landscape.

6.3.2 Learning Algorithms

The comparison between two different machine learning technique's ability to predict the "performance rankings" of the selected algorithms forms the backbone of this experiment. Two learning techniques commonly used for regression analysis were selected: (1) artificial neural networks and (2) random forests. To avoid implementation bias, standard implementations of each were selected. Each method is then trained using a number of different landscapes, with the values for the landscape characteristics (see Table 6.1) forming the input to each. Desired outputs for each algorithm and landscape characteristic combination are determined based on the average performance of an algorithm for a given combination of characteristics. For one set of trials, the mean error is used in its raw form (with the idea that this would capture any nuances in the training data, which may be lost if trained on only ranked data) and in another, the mean errors post-ranking (i.e. the expected rank of the algorithm) are used. Using the expected ranking instead of the mean error incurs the significant drawback that the training data, and setup, need to be readjusted to incorporate any additional algorithms, but does mean the desired outputs used for training more accurately mirror

the final rankings. The predictors then assess a given set of landscape characteristics, making a prediction *for each algorithm* based on whichever training data was used. The outputs of these are ranked across algorithms to give the final rankings of the algorithms. The two predictor methods are not briefly described.

Artificial Neural Network

The *Neural Network Toolbox* [3] provided by *Matlab* was chosen for the artificial neural network implementation. Neural networks were generated and trained with the default parameters offered by the *Matlab* toolbox: **85%** of the generated training set was used for training, **15%** of the training set was used for validation. As the results of the regression-based Neural Network are interpreted as rankings, a script implementing **k**-fold cross-validation to compare the predicted rankings with the actual rankings was created, based on the estimated error produced by the ANN. Scaled conjugate gradient backpropagation training was used (Møller, 1993), to allow for GPU parallelisation of ANN training. An ANN was trained for each algorithm, configured with **four inputs** (one for each landscape characteristic), and **one output** and a **single hidden layer** comprised of **five neurons**. Inputs were normalised between -1 and 1 prior to training, as is common practice when using gradient based training functions (LeCun et al., 1998).

Random Forests

The *Random Forest* package for *R* (Liaw and Wiener, 2002) was chosen for the random forest implementation. Random forests are generated with **fifty** trees, **two** variables tried at each split and a sample size of **twenty**. Unlike ANNs, there is no gradient based training function, and so there is no need to normalise the inputs. As such, no normalisation was performed on the data set prior to use as training data for the random forests. As with the ANNs, one random forest was generated to specialise in making a prediction for each algorithm, and the output from the forests were then compared and ranked, to create the predicted ranking of algorithm performance for a given set of characteristics.

6.3.3 Testing the Predictors

In order to thoroughly test the performance of the learning algorithms, the accepted method of k-fold cross validation (with a k of 10) (Kohavi, 1995) was used to obtain performance data from each learning algorithm. **Five** of each model was created, each trained using a different random seed and tested using a different selection of cross-validation indices, to further ensure a thorough testing strategy.

[3] http://www.mathworks.co.uk/products/neural-network/

Table 6.2: Overall classification accuracy, and correlation coefficients for each of the learning algorithms and training data configurations.

	Accuracy	Spearman's Rho	Kappa
No learning alg. used	55%	0.83	0.472
ANN trained on mean values	51%	0.82	0.43
ANN trained on rankings	64%	0.91	0.58
RF trained on mean values	63%	0.91	0.57
RF trained on rankings	72%	0.91	0.67

6.4 Results

In order to analyse and present the performance of the predictors, three measures of performance are used: (1) The percentage of algorithms placed in the correct ranking position, visualised using a confusion matrix (Stehman, 1997), (2) Spearman's rank correlation coefficient (Spearman's rho, see the original 1904 paper reprinted in Spearman (2010)), commonly used to determine the level of "agreement" between two judges. In this instance, the two "judges" are the *actual* ordering of the nature-inspired algorithms, and the *predicted* ordering. A strong positive coefficient (≥ 0.7) indicates successful prediction of the algorithm ranking, a coefficient close to zero indicates no correlation, and a strong negative correlation indicates that the ranking has predicted an *opposite* ordering (and the "correct" ordering can be obtained by merely reversing the predictions), and (3) Cohen's Kappa, which also measures agreement between two judges, but adjusts agreement that could happen by chance (Cohen, 1960).

Overall accuracy, Spearman's rho and Cohen's Kappa were calculated for each of five tests: No learning algorithm used - that is, taking the average ranking across all characteristics, and using these rankings in all cases without consideration for landscape characteristics - ANNs, using targets of both the raw mean errors and the mean errors ranked, and RFs, again using targets of both the raw mean errors and mean errors ranked. These results are depicted in Table 6.2. The best overall accuracy, and highest Cohen's Kappa value, were obtained from the random forest implementations, trained using data sets which had been pre-ranked. Pre-ranking the training data has the unfortunate side effect of "closing" the system, meaning that additional learning algorithms cannot be added without re-training, but the performance benefit for doing so is large (a 13% accuracy increase for ANNs, and a 9% accuracy increase for RFs).

Confusion matrices for each set are also shown. Not using a learning algorithm (Fig. 6.1) provides a correct prediction of the best algorithm 73.5% of the time, but this is improved upon by just under 10% when learning algorithms are trained using rankings in both ANNs (Fig. 6.2b) and RFs (Fig. 6.3b). The performance in terms of correctly identifying the best algorithm is not improved upon by either the ANNs or RFs trained on mean errors (Fig. 6.2a and Fig. 6.3a respectively), despite the overall accuracy and kappa value of the RF being better than in the 'no learning algorithm' prediction.

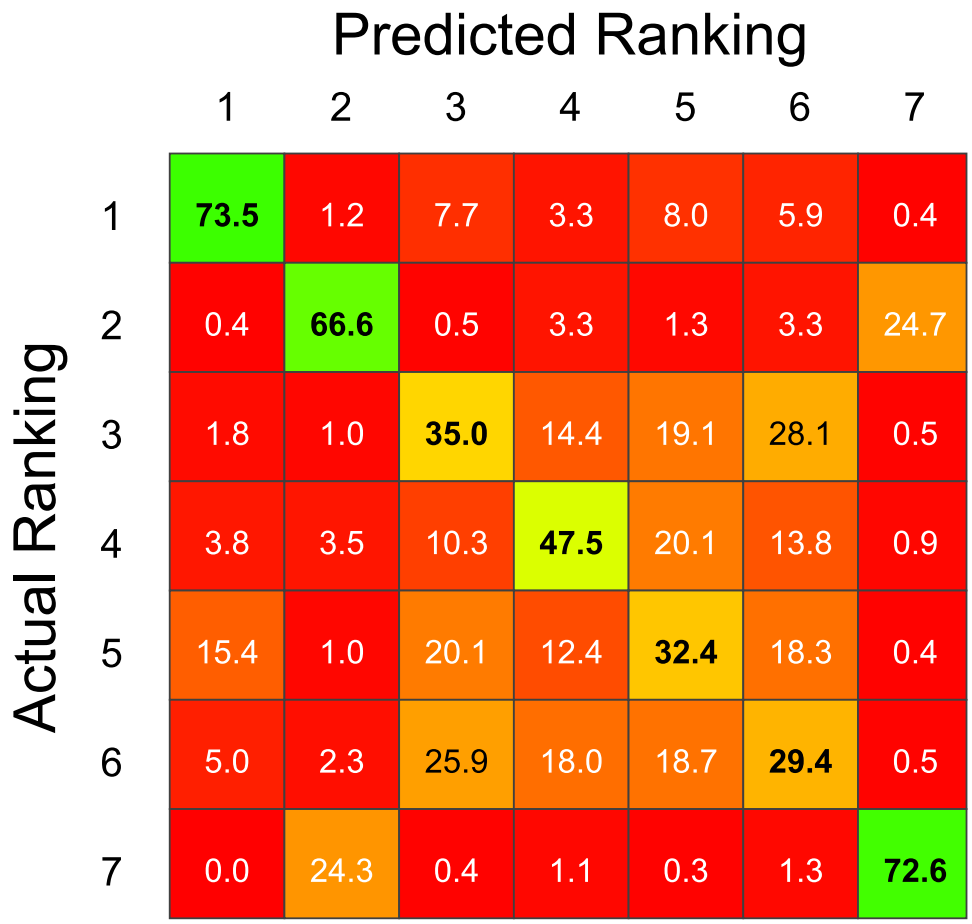


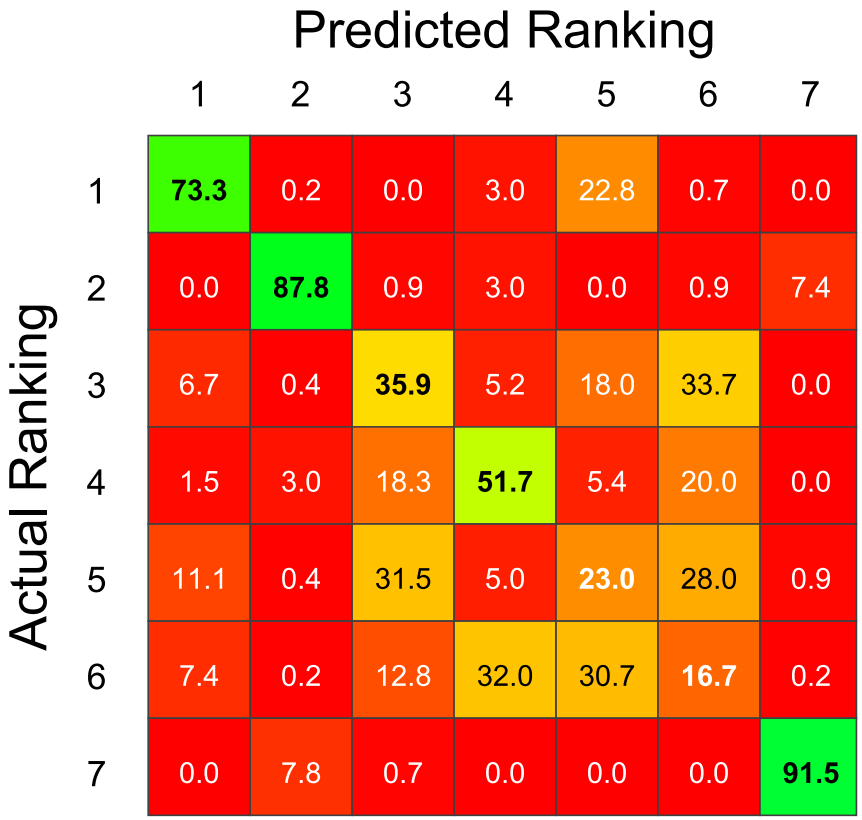
Figure 6.1: Cross-validated confusion matrix showing the performance when not using a learning algorithm, merely using the average ranking of an algorithm with no characteristic-based ranking estimation. Each value is as a percentage of 460 samples.

In all cases of using a learning algorithm, identifying the *worst performing algorithm* is greatly improved by using landscape characteristics as a predictor, with an increase of approximately 20% correctly identified cases. Both the extremities, and also the second place, algorithm score highly in all cases, and the difficulty occurs in identifying algorithms which are performing in the mid-range of the rankings. This is likely due to larger variability in the placement of algorithms at these rankings, and, for the algorithms trained on mean values, less differentiation between the average error.

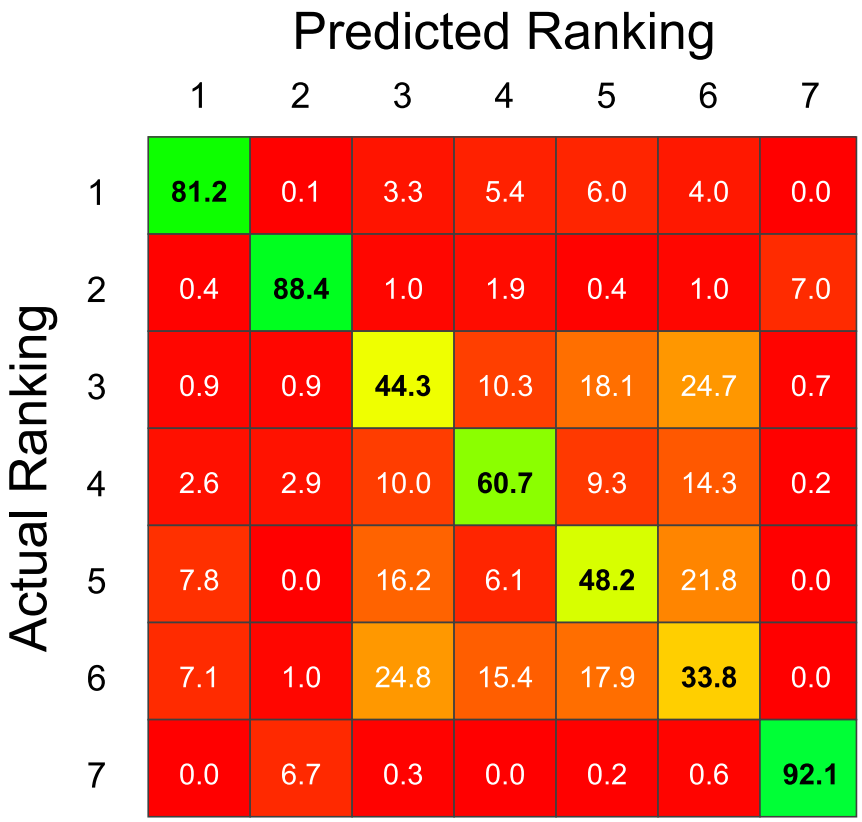
6.5 Conclusions

In this Chapter, the feasibility of using machine learning techniques to "rank" the performance of continuous optimisation algorithms based on the fitness landscape characteristics of the problem has been explored. Ranking algorithms in this way provides a method for informing the choice of the "best" algorithm to use for a given problem, and also offers suggestions as to the *approximate ordering* of the performance of the remaining algorithms. In this way, the question of whether or not a data set generated for profiling an algorithm using the methodology from Chapter 3 has been answered.

Improved prediction accuracy was obtained when using landscape characteristics as a predictor in 3 out of 4 methods, with random forests proving to be the better

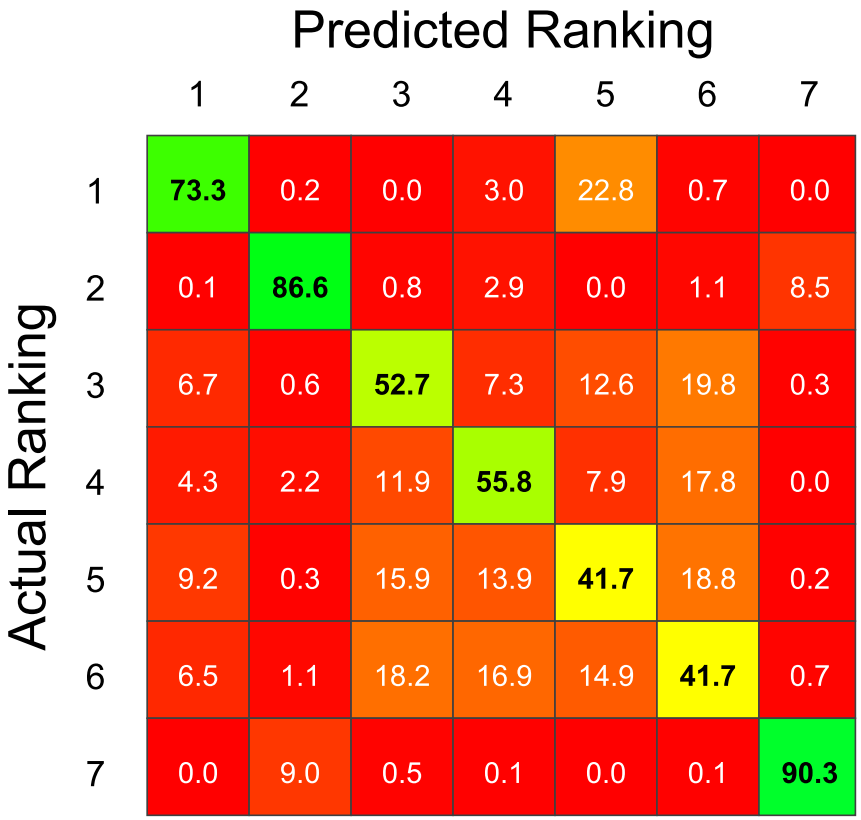


(a) Trained using raw mean errors.

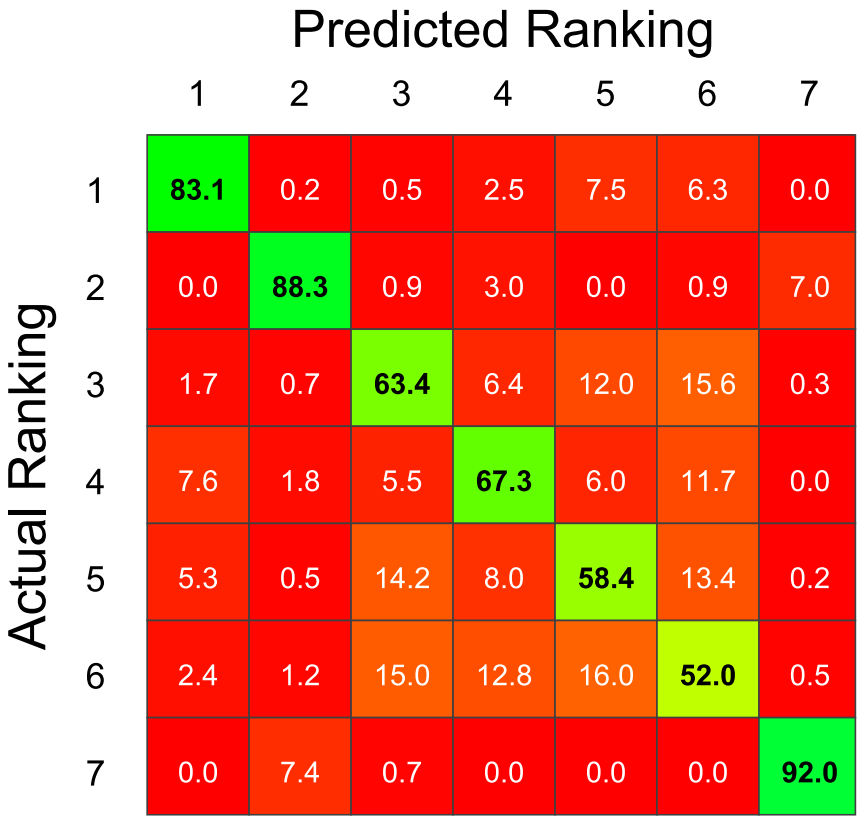


(b) Trained using ranked mean errors.

Figure 6.2: Cross-validated confusion matrix showing the performance of five artificial neural network predictors, trained both on the mean exact error values and trained on ranked mean exact error values. Each value is as a percentage of 2300 samples.



(a) Trained using raw mean errors.



(b) Trained using ranked mean errors.

Figure 6.3: Cross-validated confusion matrix showing the performance of five random forest predictors, trained both on the mean exact error values and on ranked mean exact error values. Each value is as a percentage of 2300 samples.

of the two learning algorithms examined in this study, for this particular case. Also explored was whether ranking the training data helped, or hindered, the process (since pre-ranking would hide intricacies in the data, but is a closer natural fit to the targets being predicted), and found that there are performance improvements from doing so. This is, however, at the cost of creating a system which cannot be as easily extended with additional algorithms.

While this work provides a foundation for showing that characteristics can be used as predictors, there are still open questions. There is ongoing discussion regarding which characteristics are important to the process. Here, those used are are a natural fit to these particular problem instances, they may not have as natural a fit in a broader problem scenario (i.e with landscapes not artificially generated). Further investigation into more possible characteristics, and how these inform the prediction process, are key to developing predictors which are both more accurate, and which have broader applicability would likely prove fruitful.

Similarly, another possible extension is to continue exploring the interaction between different algorithms, on broader problems (and broader *ranges* of characteristics), and how these fit into the rankings. With a broader range of problems, and more algorithms included, the need for characteristic-based prediction may increase, as the simple 'no learning algorithm' method struggles with the increased diversity in the data set.

It is hoped that this work provides a foundation for future work on the prediction of algorithm performance in terms of fitness landscape characteristics, particularly using novel methods of characterising problem features and additional learning techniques.

Chapter 7

Conclusion

7.1 Summary

This study was inspired by the problem of an increasing number of nature-inspired algorithms being developed. The overall aim was to investigate whether fitness landscape characteristics could be used as the cornerstone for rigorously characterising these algorithms in a way that is useful to algorithm designers. Many algorithms are presented with limited details of their performance on a small subset of problem classes, offering little insight into the performance of the algorithm in general. In Chapter 3, a novel method for characterising algorithms based on characteristics of a problem's associated fitness landscape was presented. This method, based on a landscape generation technique, is easy to implement, scalable and algorithm-independent, making it an ideal technique for examining the strengths and weaknesses of algorithms when applied to continuous optimisation problems.

Three research questions were proposed at the outset of this thesis:

1. To what extent can fitness landscape characteristics be used to establish a performance profile of an algorithm, and thus distinguish between different algorithms in terms of performance?

2. To what extent does tuning alter the performance profile of an algorithm with regard to each of the landscape characteristics defined?

3. To what extent can algorithm performance be predicted by using the defined landscape characteristics as input to a classification algorithm?

In order to answer the first of these questions, performance profiles of six different nature-inspired algorithms were established in Chapter 4, using the methodology described. To assess the extent to which these profiles can be used to differentiate between algorithms, the profiles were examined primarily on two key criteria; Each algorithm's capability to produce accurate solutions, and the algorithm's tendency to produce accurate solutions *consistently* as a characteristic changed, representing that algorithm's robustness to that particular characteristic. Statistical analysis was also

used to identify where algorithm performance did, or did not, significantly change as a landscape's characteristics changed. Across the algorithms included in this study, a varied set of results, and also tolerances to landscape characteristics, were identified. Some algorithms were much better at coping with certain characteristics than others, and some algorithms offered much better performance in certain areas than others. Importantly, the ideas in the No Free Lunch Theorem (Wolpert and Macready, 1997) were also reaffirmed - there was no one algorithm able to perform consistently, irrespective of the landscape characteristics. Obtaining unique performance profiles for even this small subset of nature-inspired algorithms suggests that fitness landscape characteristics are a useful tool for profiling the performance of algorithms.

Having answered the first question, using algorithms with 'out of the box' parameter settings, the second question, explored in Chapter 5, required investigation into the extent to which parameter tuning affects the performance profile of an algorithm. Of particular interest, is whether or not it is possible to tell whether an algorithm does, or does not, benefit from tuning in terms of fitness landscape characteristics. Using an automated parameter tuning methodology (F-Racing), 'tuned' performance profiles of the same six nature-inspired algorithms were obtained, and these were compared to the performance profiles of the algorithms pre-tuning, to identify where tuning had, or had not, had a significant impact. As with the first question, results were varied across the algorithms. Broadly, algorithms fit into three categories - those that did not benefit from tuning, those that benefit from tuning only when a problem is 'difficult', and those that always benefit from tuning. As tuning is a time-consuming process, having an idea, in advance of implementation, of whether an algorithm benefits from tuning (and if so, when it benefits from tuning), is a substantial boon. The results of this Chapter show that once again landscape characteristics are a useful tool for describing the performance profile of an algorithm with respect to improvement (or lack thereof) when parameter tuning.

To address the third question, investigation into the performance of two classification algorithms using data sets generated using the methodology as training sets was carried out in Chapter 6. The advantage here is significant; While the technique presented offers insight into algorithm performance, the interpretation of the results can still be a difficult task, particularly when a problem does not neatly match up to a particular set of trialled characteristics. By using machine learning techniques, and automatically ranking algorithm performance, there is an alternative to manually interpreting the data and essentially "interpolating" those characteristic combinations not investigating explicitly. It was found that when used to predict the performance of algorithms on randomly generated landscapes, the prediction technique proved accurate, again showing that the method presented in Chapter 4 is reasonable. Particularly noteworthy was the observation that *pre-ranking* the data before training, and using random forests (over artificial neural networks) offers better prediction performance. This offers a *starting point* for using landscape characteristics as a performance predictor, although there are still outstanding questions, the most important of which being:

How could the technique be applied to 'real world' problems, where the characteristics are less easily established? This question is one of the key concerns with most techniques for predicting algorithm performance, and requires significant further study to work towards resolution.

In answering these questions, this thesis provides the first steps towards novel usage of fitness landscape characteristics for profiling, characterising and analysing nature-inspired continuous optimisation algorithms. The three questions show that fitness landscape characteristics are an appropriate, and useful, way to describe the performance of an algorithm, and that they can easily be used to differentiate between the performance of several algorithms. Particularly useful is the ability to show an algorithm's robustness to certain characteristics.

The method for profiling algorithms presented here is not dependent on algorithm or problem, the main benefit being that it can be used to highlight the profile of any algorithm that can be applied to continuous optimisation. If an algorithm analysis technique, such as the one presented in this thesis, becomes one of the standard elements that accompanies the presentation of a novel algorithm the clarity of algorithm performance will, over time, improve.

By presenting an algorithm in this way, the relative strengths and weaknesses are instantly visible and comparable between algorithms "at a glance". The particular usefulness here is that as the space of algorithms increase, it is important to highlight precisely *why* a novel algorithm occupies a niche space, and what that niche space is; Using the landscape characteristic analysis technique, it is entirely possible to show that a novel nature-inspired algorithm presents a unique profile in terms of performance across the range of characteristics – something that is not immediately obvious when only presented with the results of benchmark problem results. Conversely, as more algorithms are profiled using this technique, the inherent weaknesses in the field will become apparent, and this will help to focus the attention of algorithm designers on precisely which characteristics need attention and, with further profiling, where the current algorithms are falling short. It is hoped that this is of particular usefulness to algorithm designers, both in the design of new algorithms and the analysis and proposal of novel optimisation algorithms.

7.2 Contributions

The contributions of this work are as follows:

A novel methodology for analysing algorithm performance using landscape characteristics has been proposed and tested. This methodology could be used by algorithm designers to showcase a novel algorithm's strengths, weaknesses and how it fits into a niche of the algorithm space. Existing work largely requires algorithm designers to present their algorithms using benchmark problems, which do not offer insight into the *broad* applicability of an algorithm and can be difficult to interpret or generalise and, as such, this methodology in part solves this problem.

In Chapter 4, portfolios of six existing nature-inspired algorithms are presented, enhancing the understanding of these algorithms with regard to their performance against landscape characteristics. This is, to my knowledge, the first broad-scale study of this kind which attempts to characterise algorithms throughout the entire nature-inspired algorithms field with respect to fitness landscape analysis. By establishing that fitness landscape characteristics can be used to identify a performance profile of algorithms, it is suggested that they are a suitable cornerstone for developing further techniques for analysing and characterising algorithms. The need for techniques to classify, characterise and differentiate algorithms is known (Woodward, 2010; Sörensen, 2013), and existing approaches acknowledge short-comings in existing approaches with a need to encapsulate *more* information about the problem (Malan and Engelbrecht, 2013). While further work is necessary to fully explore the notion of fitness-landscape based analysis of algorithms, the identification of fitness landscape characteristics as a profiling technique offers an alternative to existing techniques, which could prove especially promising when partnered with those techniques as a means of incorporating additional information about a particular problem.

In Chapter 5, insight into when it is advantageous (or not) to tune specific algorithms, given landscape characteristics, is presented. The use of this methodology when presenting or developing an algorithm is useful. To my knowledge, this is again the first *broad* study into the effect of parameter tuning, encapsulating algorithms both in and outside of the evolutionary algorithm field. In observing that algorithms react differently to tuning - based on the profiles obtained in this study - and differently with regard to varying landscape characteristics, there is further evidence that fitness landscape characteristics are a suitable way to analyse algorithm performance. Having acknowledged that these algorithms respond differently to tuning, the importance of discussing how an algorithm reacts to tuning is highlighted, reaffirming the discussion of Eiben and Smit (2011). With this knowledge, algorithm designers can obtain performance profiles and quantitatively describe the robustness of their algorithm using both standard parameters, and when tuned, to better describe the performance of a novel algorithm.

Finally, in chapter 6, it is shown that algorithm performance prediction is improved when using landscape characteristics as inputs to a classification algorithm over naïve algorithm selection. In the studies shown in this work, three out of four classification algorithm configurations predicted the best performing algorithm with greater accuracy than simply choosing the best performing algorithm for all problems. Most notably, the observation is made that *random forests* appear to be the better classifier when compared to artificial neural networks and *pre-ranking* the training data gives better results when seeking algorithm rankings. Previous attempts to predict algorithm performance have primarily focused on using measures of hardness as inputs to classification algorithms, and this alternative approach may provide a complementary technique which, partnered with measures of hardness, could provide improved accuracy in algorithm selection. The field of automated algorithm selection is still in its

infancy, and it is hoped that insights such as these may help to guide the development of further work in this area.

7.3 Further Work

In undertaking the trialling of this methodology, the process of *cataloguing* existing algorithms using this technique has begun. By thoroughly analysing the strengths and weaknesses of six existing nature-inspired algorithms, some well explored and some less so, insight into when and why these algorithms may be applicable to problems exhibiting certain characteristics has been provided, plus insight into the effect of how they are affected by the process of parameter tuning.

The catalogue so far only contains a small selection of potential algorithms, and only one version of each of those. The first stage of further work would broaden this study by increasing the algorithm count this study includes, examining a much wider range of algorithms. The benefits of this would be threefold; Firstly, algorithm designers could have a much greater picture of the strengths and weaknesses of specific algorithms, offering a good starting point for choosing algorithms given problem characteristics. Secondly, having a larger overview of the strengths and weaknesses of the field generally would highlight *which* characteristics pose the most significant challenges for the algorithms currently available. For the algorithms included in this study, it was found that both increasing numbers of local optima and increasing dimensionality were significant problems, but different algorithms coped better with the two different characteristics. Having an overview of more algorithms may indicate that there are more algorithms able to cope with local optima, and that resources would be best spent on algorithms able to cope with increasing dimensionality, for example. Thirdly, the increased usage of this technique would offer refinements to the technique itself, perhaps suggesting new characteristics - or variations on the landscape generation methodology - which may offer additional insight which may prove invaluable.

Another avenue of interest, rather than simply adding additional algorithms to the study, is to look at the *features* of algorithms that have particular benefits with respect to different characteristics. This would prove especially interesting to the concept of a meta-heuristic, where rather than selecting a particular algorithm particular features that can be used are selected, and an algorithm is built from these "components." This analysis methodology provides an ideal test bed for such a study; An algorithm can be executed both with, and without, a feature, and the results across each characteristic can be compared, to see precisely the effect the feature has on the given characteristic. In this way, it is hoped that algorithm designers could identify the specific features that are useful for avoiding local optima, for example, rather than viewing algorithms as a whole.

Analysing all the existing nature-inspired algorithms and classifying these on a wide variety of different problems sets is a huge task. It is hoped that by presenting an easy to implement, and reasonably quick to compute, methodology for presenting the strengths

and weaknesses of an algorithm, and showing that automated algorithm selection is possible, this thesis offers a starting point towards making the algorithm selection problem somewhat easier. Once again, it is noted that presenting novel algorithms using the methodology proposed in this thesis would offer practical insight into the behaviour of algorithms, and a standardised method for introducing algorithms would greatly reduce the problems a reader faces when choosing to use a nature-inspired algorithm.

Bibliography

Adler, J. (1966). Chemotaxis in bacteria. *Science*, 153(3737):708–716.

Akay, B. and Karaboga, D. (2009). Parameter tuning for the artificial bee colony algorithm. In Nguyen, N., Kowalczyk, R., and Chen, S.-M., editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, volume 5796 of *Lecture Notes in Computer Science*, pages 608–619. Springer Berlin Heidelberg.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Bäck, T. and Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23.

Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. *Hybrid Metaheuristics*, pages 108–122.

Barnett, L. (1998). Ruggedness and neutrality: The nkp family of fitness landscapes. In *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, pages 18–27.

Barr, R., Golden, B., and Kelly, J. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32.

Bartz-Beielstein, T. (2003). Experimental analysis of evolution strategies: Overview and comprehensive introduction.

Baykasoglu, A., Ozbakir, L., and Tapkan, P. (2009). The bees algorithm for workload balancing in examination job assignment. *European Journal of Industrial Engineering*, 3(4):424–435.

Bellman, R. (1972). *Dynamic Programming*. Princeton University Press.

Berg, H. C. (2000). Motile behavior of bacteria. *Physics Today*, 53(1):24–30.

Beyer, H.-g. and Schwefel, H.-p. (2002). Evolution strategies. *Natural Computing*, 1:3–52.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 11–18, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-race and iterated f-race: An overview. *Experimental methods for the analysis of optimization algorithms*, pages 311–336.

Bischl, B., Mersmann, O., Trautmann, H., and Preuß, M. (2012). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 313–320. ACM.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. *Nature*, 406(6791):39–42.

Brabazon, A. and O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Springer-Verlag.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brownlee, J. (2011). *Clever Algorithms*. LuLu, Australia, 1st edition.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology. *International series in operations research and management science*, pages 457–474.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, pages 449–468. Springer.

Chen, H., Zhu, Y., and Hu, K. (2008). Self-adaptation in bacterial foraging optimization algorithm. In *3rd Int. Conf. Intell. System and Knowledge Eng.*, volume 1, pages 1026–1031. IEEE.

Chen, T. and Chen, H. (2009). Mixed–discrete structural optimization using a rank-niche evolution strategy. *Engineering Optimization*, 41(1):39–58.

Cheng, R., Gen, M., and Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithmsi. representation. *Computers & Industrial Engineering*, 30(4):983–997.

Christensen, S. and Oppacher, F. (2001). What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2001, pages 1219–1226.

Coelho, L. d. S. and Mariani, V. C. (2009). An improved harmony search algorithm for power economic load dispatch. *Energy Conversion and Management*, 50(10):2522–2526.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Crossley, M., Nisbet, A., and Amos, M. (2013a). Quantifying the impact of parameter tuning on nature-inspired algorithms. In Lio, P., Miglino, O., Nicosia, G., Nolfi, S., and Pavone, M., editors, *Advances in Artificial Life, ECAL 2013*, pages 925–932. MIT Press.

Crossley, M., Nisbet, A., and Amos, M. (2013b). Quantifying the impact of parameter tuning on nature-inspired algorithms. In Lio, P., Miglino, O., Nicosia, G., Nolfi, S., and Pavone, M., editors, *Advances in Artificial Life, ECAL 2013*, pages 925–932. MIT Press.

Dantzig, G. B. (1965). *Linear programming and extensions*. Princeton university press.

Das, T., Venayagamoorthy, G., and Aliyu, U. (2008). Bio-inspired algorithms for the design of multiple optimal power system stabilizers: Sppso and bfa. *Industry Applications, IEEE Transactions on*, 44(5):1445–1457.

Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J.-C., and Harley, R. G. (2008). Particle swarm optimization: basic concepts, variants and applications in power systems. *Evolutionary Computation, IEEE Transactions on*, 12(2):171–195.

Donelli, M. and Massa, A. (2005). Computational approach based on a particle swarm optimizer for microwave imaging of two-dimensional dielectric scatterers. *Microwave Theory and Techniques, IEEE Transactions on*, 53(5):1761–1776.

Eberhart, R. and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *Proc. 2001 Congr. Evol. Computat.*, volume 1, pages 81–86. Ieee.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141.

Eiben, A. E. and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31.

Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43–53.

Erdal, F., Doğan, E., and Saka, M. P. (2011). Optimum design of cellular beams using harmony search and particle swarm optimizers. *Journal of Constructional Steel Research*, 67(2):237–247.

Eshelman, L. J., Caruana, R. A., and Schaffer, J. D. (1989). Biases in the crossover landscape. In *Proceedings of the third international conference on Genetic algorithms*, pages 10–19. Morgan Kaufmann Publishers Inc.

Fahmy, A., Kalyoncu, M., and Castellani, M. (2012). Automatic design of control systems for robot manipulators using the bees algorithm. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 226(4):497–508.

Fang, H.-L., Ross, P., and Corne, D. (1993). *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems*. University of Edinburgh, Department of Artificial Intelligence.

Fesanghary, M., Damangir, E., and Soleimani, I. (2009). Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm. *Applied Thermal Engineering*, 29(5):1026–1031.

Fisher, R. A. (1999). *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press.

Forsgren, A., Gill, P. E., and Wright, M. H. (2002). Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597.

Gaing, Z.-L. (2003). Particle swarm optimization to solving the economic dispatch considering the generator constraints. *Power Systems, IEEE Transactions on*, 18(3):1187–1195.

Gaing, Z.-L. (2004). A particle swarm optimization approach for optimum design of pid controller in avr system. *Energy Conversion, IEEE Transactions on*, 19(2):384–391.

Gallagher, M. and Yuan, B. (2006). A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603.

Geard, N., Wiles, J., Hallinan, J., Tonkes, B., and Skellett, B. (2002). A comparison of neutral landscapes-nk, nkp and nkq. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 205–210. IEEE.

Geem, Z. and Kim, J. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68.

Geem, Z. W. (2006). Optimal cost design of water distribution networks using harmony search. *Engineering Optimization*, 38(03):259–277.

Gen, M. and Cheng, R. (2000). *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons.

Gill, P. E., Murray, W., and Wright, M. H. (1981). Practical optimization.

Goldberg, D. and Holland, J. (1988). Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128.

Grünbaum, D. (1998). Schooling as a strategy for taxis in a noisy environment. *Evolutionary Ecology*, 12(5):503–522.

Gudise, V. G. and Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 110–117. IEEE.

Haddad, O. B., Afshar, A., and Marino, M. A. (2006). Honey-bees mating optimization (hbmo) algorithm: a new heuristic approach for water resources optimization. *Water Resources Management*, 20(5):661–680.

Hansen, N. and Kern, S. (2004). Evaluating the cma evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE.

Hasançebi, O. (2008). Adaptive evolution strategies in structural optimization: Enhancing their computational performance with applications to large-scale structures. *Computers & structures*, 86(1):119–132.

Hassan, L. H., Moghavvemi, M., Almurib, H. A., and Steinmayer, O. (2013). Application of genetic algorithm in optimization of unified power flow controller parameters and its location in the power system network. *International Journal of Electrical Power & Energy Systems*, 46:89–97.

Hendtlass, T. (2009). Particle swarm optimisation and high dimensional problem spaces. In *2009 IEEE Congress on Evolutionary Computation, CEC'09.*, pages 1988–1994. IEEE.

Heo, J. S., Lee, K. Y., and Garduno-Ramirez, R. (2006). Multiobjective control of power plants using particle swarm optimization techniques. *Energy Conversion, IEEE Transactions on*, 21(2):552–561.

Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*, 12(4):265–319.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, pages 13–30.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314.

Hooker, J. (1995). Testing heuristics: we have it all wrong. *Journal of Heuristics*.

Horn, J. and Goldberg, D. (1994). Genetic algorithm difficulty and the modality of fitness landscapes. In *Foundations of Genetic Algorithms 3*.

Hou, E. S., Ansari, N., and Ren, H. (1994). A genetic algorithm for multiprocessor scheduling. *Parallel and Distributed Systems, IEEE Transactions on*, 5(2):113–120.

Hu, X., Shonkwiler, R., and Spruill, M. (2009). Random restarts in global optimization.

Jani, R. (2008). A Generator for Multimodal Test Functions. In Li, X., editor, *Proc. SEAL 2008: LNCS 5361*, volume 3, pages 239–248. SPringer-Verlag.

Janson, S., Middendorf, M., and Beekman, M. (2005). Honeybee swarms: how do scouts guide a swarm of uninformed bees? *Animal Behaviour*, 70(2):349–358.

Jin, Y. (2004). Constructing dynamic optimization test problems using the multi-objective optimization concept. In Raidl, G., editor, *Applications of Evolutionary Computing: LNCS 3005*, volume LNCS 3005, pages 525–536. Springer-Verlag.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192.

Joya, G., Atencia, M., and Sandoval, F. (2002). Hopfield neural networks for optimization: study of the different dynamics. *Neurocomputing*, 43(1):219–237.

Kannan, S., Slochanal, S., Subbaraj, P., and Padhy, N. P. (2004). Application of particle swarm optimization technique and its variants to generation expansion planning problem. *Electric Power Systems Research*, 70(3):203–210.

Kantorovich, L. V. (1940). A new method of solving some classes of extremal problems. In *Doklady Akad Sci USSR*, volume 28, pages 211–214.

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471.

Kauffman, S. and Levin, S. (1987). Towards a general theory of adaptive walks on rugged landscapes. *Journal of theoretical Biology*, 128(1):11–45.

Kauffman, S. A. and Weinberger, E. D. (1989). The nk model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of theoretical biology*, 141(2):211–245.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings. ...*, pages 1942–1948.

Khorram, E. and Jaberipour, M. (2011). Harmony search algorithm for solving combined heat and power economic dispatch problems. *Energy Conversion and Management*, 52(2):1550–1554.

Kim, H.-K., Chong, J.-K., Park, K.-Y., and Lowther, D. A. (2007). Differential evolution strategy for constrained global optimization and application to practical engineering problems. *Magnetics, IEEE Transactions on*, 43(4):1565–1568.

Klee, V. and Minty, G. J. (1970). How good is the simplex algorithm. Technical report, DTIC Document.

Knowles, J. and Corne, D. (1999). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koster, C. and Beney, J. (2007). On the importance of parameter tuning in text categorization. *Perspectives of Systems Informatics*, pages 270–283.

Kotthoff, L., Gent, I. P., and Miguel, I. (2012). An evaluation of machine learning in algorithm selection for search problems. *AI Communications*, 25(3):257–270.

Kukkonen, S. and Lampinen, J. (2005). GDE3: The third Evolution Step of Generalized Differential Evolution. *2005 IEEE Congress on Evolutionary Computation*, pages 443–450.

Kumar, K. S. and Jayabarathi, T. (2012). Power system reconfiguration and loss minimization for an distribution systems using bacterial foraging optimization algorithm. *International Journal of Electrical Power & Energy Systems*, 36(1):13 – 17.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

Lee, K. and Geem, Z. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194(36-38):3902–3933.

Lee, K. S. and Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9):781–798.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Liu, Y. and Passino, K. (2002). Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors. *Journal of Optimization Theory and Applications*, 115(3):603–628.

Lucas, S. M. (2005). Evolving a neural network location evaluator to play ms. pac-man. In *CIG*. Citeseer.

Malan, K. and Engelbrecht, A. (2013). Ruggedness, funnels and gradients in fitness landscapes and the effect on pso performance. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 963–970.

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

Malan, K. M. and Engelbrecht, A. P. (2009). Quantifying ruggedness of continuous landscapes using entropy. In *2009 IEEE Congress on Evolutionary Computation*, pages 1440–1447. IEEE.

Malan, K. M. and Engelbrecht, A. P. (2014b). Particle swarm optimisation failure prediction based on fitness landscape characteristics. In *Swarm Intelligence (SIS), 2014 IEEE Symposium on*, pages 1–9. IEEE.

Mandischer, M. (2002). A comparison of evolution strategies and backpropagation for neural network training. *Neurocomputing*, 42(14):87 – 117. Evolutionary neural systems.

Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Ser, J. D., Bilbao, M., Salcedo-Sanz, S., and Geem, Z. (2013). A survey on applications of the harmony search algorithm. *Engineering Applications of Artificial Intelligence*, 26(8):1818 – 1831.

Maron, O. and Moore, A. (1993). Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263.

Maron, O. and Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11:193–225.

Merz, P. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *Evolutionary Computation, IEEE*, 4(4):337–352.

Merz, P. (2001). Memetic algorithms for combinatorial optimization problems: fitness landscapes and effective search strategies.

Merz, P. (2004). Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325.

Merz, P. and Freisleben, B. (1998). On the effectiveness of evolutionary search in high-dimensional nk-landscapes. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 741–745.

Merz, P. and Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Trans. Evol. Computat.*, 4(4):337–352.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34(10):2964–2975.

Mester, D., Bräysy, O., and Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications*, 32(2):508–517.

Michalewicz, Z., Deb, K., and Schmidt, M. (2000). Test-case generator for nonlinear continuous parameter optimization techniques. *IEEE Transactions on Evolutionary Computation*, 4(3):197–215.

Michalewicz, Z. and Janikow, C. (1992). A modified genetic algorithm for optimal control problems. *Computers and Mathematics with Applications*, 23(12):83–94.

Mishra, S. and Bhende, C. N. (2007). Bacterial foraging technique-based optimized active power filter for load compensation. *Power Delivery, IEEE Transactions on*, 22(1):457–465.

Mitchell, M., Forrest, S., and Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the first european conference on artificial life*, pages 245–254. Cambridge: The MIT Press.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533.

Morgan, R. and Gallagher, M. (2010). When does dependency modelling help? Using a randomized landscape generator to compare algorithms in terms of problem structure. In et al Schaefer, R., editor, *PPSN XI*, pages 94–103. Springer-Verlag.

Morgan, R. and Gallagher, M. (2012a). Length scale for characterising continuous optimization problems. In *Parallel Problem Solving from Nature-PPSN XII*, pages 407–416. Springer.

Morgan, R. and Gallagher, M. (2012b). Using landscape topology to compare continuous metaheuristics: A framework and case study on edas and ridge structure. *Evolutionary computation*, 20(2):277–299.

Muñoz, M. A., Kirley, M., and Halgamuge, S. K. (2012). A meta-learning prediction model of algorithm performance for continuous optimization problems. In *Parallel Problem Solving from Nature-PPSN XII*, pages 226–235. Springer.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary computation*, 1(1):25–49.

Muller, S. D., Marchetto, J., Airaghi, S., and Kournoutsakos, P. (2002). Optimization based on bacterial chemotaxis. *Evolutionary Computation, IEEE Transactions on*, 6(1):16–29.

Nannen, V., Smit, S. K., and Eiben, A. E. (2008). Costs and benefits of tuning parameters of evolutionary algorithms. In *Parallel Problem Solving from Nature–PPSN X*, pages 528–538. Springer.

Nara, K., Shiose, A., Kitagawa, M., and Ishihara, T. (1992). Implementation of genetic algorithm for distribution systems loss minimum re-configuration. *Power Systems, IEEE Transactions on*, 7(3):1044–1051.

Nasseri, M., Asghari, K., and Abedini, M. (2008). Optimized scenario for rainfall forecasting using genetic algorithm coupled with artificial neural network. *Expert Systems with Applications*, 35(3):1415–1421.

Newman, M. E. J. and Engelhardt, R. (1998). Effects of selective neutrality on the evolution of molecular species. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1403):1333–1338.

Özbakır, L. and Tapkan, P. (2011). Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications*, 38(9):11947–11957.

Özcan, E., Bilgin, B., and Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23.

Pan, I., Das, S., and Gupta, A. (2011). Tuning of an optimal fuzzy pid controller with stochastic algorithms for networked control systems with random time delay. *ISA transactions*, 50(1):28–36.

Papadrakakis, M., Lagaros, N. D., and Tsompanakis, Y. (1998). Structural optimization using evolution strategies and neural networks. *Computer methods in applied mechanics and engineering*, 156(1):309–333.

Park, J.-B., Lee, K.-S., Shin, J.-R., and Lee, K. Y. (2005). A particle swarm optimization for economic dispatch with nonsmooth cost functions. *Power Systems, IEEE Transactions on*, 20(1):34–42.

Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67.

Pearl, J. (1984). Heuristics: intelligent search strategies for computer problem solving.

Pelikan, M., Sastry, K., Goldberg, D. E., Butz, M. V., and Hauschild, M. (2009). Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 851–858. ACM.

Pham, D. and Castellani, M. (2009). The bees algorithm: modelling foraging behaviour to solve continuous optimization problems. *Proc. Inst. Mech. Engineers, Part C: J. Mech. Eng. Sci.*, 223(12):2919.

Pham, D. and Castellani, M. (2013). Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms. *Soft Computing*, pages 1–33.

Pham, D. and Ghanbarzadeh, A. (2007). Multi-objective optimisation using the bees algorithm. In *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007): Whittles, Dunbeath, Scotland*, volume 242, pages 111–116.

Pham, D., Ghanbarzadeh, A., and Koc, E. (2006a). The Bees Algorithm A Novel Tool for Complex Optimisation Problems. In Pham, D., Eldukhri, E., and Soroka, A., editors, *Intelligent Production Machines and Systems*, pages 454–459.

Pham, D., Ghanbarzadeh, A., and Koc, E. (2006b). The Bees Algorithm: A Novel Tool for Complex Optimisation Problems. In Pham, D., Eldukhri, E., and Soroka, A., editors, *Intelligent Production Machines and Systems*, pages 454–459.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M. (2005). The bees algorithm. technical note. Technical report.

Pham, D., Koç, E., Ghanbarzadeh, A., and Otri, S. (2006c). Optimisation of the weights of multi-layered perceptrons using the bees algorithm. In *Proceedings of 5th international symposium on intelligent manufacturing systems*, pages 38–46.

Pham, D., Koc, E., Lee, J., and Phrueksanant, J. (2007a). Using the bees algorithm to schedule jobs for a machine. In *Proceedings of eighth international conference on laser metrology, CMM and machine tool performance*, pages 430–439.

Pham, D., Soroka, A., Koc, E., Ghanbarzadeh, A., and Otri, S. (2007b). Some applications of the bees algorithm in engineering design and manufacture. In *Proceedings of international conference on manufacturing automation (ICMA 2007), Singapore*, pages 782–794.

Pham, D., Soroka, A. J., Ghanbarzadeh, A., Koc, E., Otri, S., and Packianather, M. (2006d). Optimising neural networks for identification of wood defects using the bees algorithm. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 1346–1351. IEEE.

Pham, D. T., Afify, A. A., and Koç, E. (2007c). Manufacturing cell formation using the Bees Algorithm. In *Innovative Production Machines and Systems Virtual Conference*.

Pham, D. T., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koç, E., and Otri, S. (2007d). Using the Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification. In *Innovative Production Machines and Systems Virtual Conference*.

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

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1):33–57.

Rechenberg, I. (1971). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dr.-Ing.* PhD thesis, Thesis, Technical University of Berlin, Department of Process Engineering.

Repoussis, P. P., Tarantilis, C. D., Bräysy, O., and Ioannou, G. (2010). A hybrid evolution strategy for the open vehicle routing problem. *Computers & Operations Research*, 37(3):443–455.

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

Rice, J. R. (1976). The algorithm selection problem. *Advances in computers*, 15:65–118.

Riley, J., Greggers, U., Smith, A., Reynolds, D., and Menzel, R. (2005). The flight paths of honeybees recruited by the waggle dance. *Nature*, 435(7039):205–207.

Ross, P., Hart, E., and Corne, D. (2003). Genetic algorithms and timetabling. In Ghosh, A. and Tsutsui, S., editors, *Advances in Evolutionary Computing*, Natural Computing Series, pages 755–771. Springer Berlin Heidelberg.

Saka, M. (2009). Optimum design of steel sway frames to bs5950 using harmony search algorithm. *Journal of Constructional Steel Research*, 65(1):36–43.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie*. Birkhäuser.

Schwefel, H.-P. (1981). *Numerical optimization of computer models*. John Wiley & Sons, Inc.

Sedighizadeh, D. and Masehian, E. (2009). Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5):486–502.

Seeley, T. D. (2009). *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press.

Selvakumar, A. I. and Thanushkodi, K. (2007). A new particle swarm optimization solution to nonconvex economic dispatch problems. *Power Systems, IEEE Transactions on*, 22(1):42–51.

Slade, W. H., Ressom, H. W., Musavi, M. T., and Miller, R. L. (2004). Inversion of ocean color observations using particle swarm optimization. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(9):1915–1923.

Smit, S. K. and Eiben, A. E. (2009). Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 399–406. IEEE.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1):6:1–6:25.

Sörensen, K. (2013). Metaheuristicsthe metaphor exposed. *International Transactions in Operational Research*.

Sousa, T., Silva, A., and Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, 30(5):767–783.

Spearman, C. (2010). The proof and measurement of association between two things. *Int J Epidemiol*, 39(5):1137–50.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

Sun, Y., Halgamuge, S. K., Kirley, M., and Munoz, M. A. (2014). On the selection of fitness landscape analysis metrics for continuous optimization problems. In *Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on*, pages 1–6. IEEE.

Syswerda, G. (1989). Uniform crossover in genetic algorithms.

Tamer Ayvaz, M. (2009). Application of harmony search algorithm to the solution of groundwater management models. *Advances in Water Resources*, 32(6):916–924.

Tangpattanakul, P. and Artrit, P. (2009). Minimum-time trajectory of robot manipulator using harmony search algorithm. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 1, pages 354–357. IEEE.

Tangpattanakul, P., Meesomboon, A., and Artrit, P. (2010). Optimal trajectory of robot manipulator using harmony search algorithms. In *Recent Advances In Harmony Search Algorithm*, pages 23–36. Springer.

Tavares, J., Pereira, F. B., and Costa, E. (2008). Multidimensional knapsack problem: a fitness landscape analysis. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, 38(3):604–16.

Uludag, G. and Sima Uyar, A. (2009). Fitness landscape analysis of differential evolution algorithms. In *Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009.*, pages 1–4.

Van der Merwe, D. and Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 215–220. IEEE.

Vasebi, A., Fesanghary, M., and Bathaee, S. (2007). Combined heat and power economic dispatch by harmony search algorithm. *International Journal of Electrical Power & Energy Systems*, 29(10):713–719.

Wachowiak, M. P., Smolíková, R., Zheng, Y., Zurada, J. M., and Elmaghraby, A. S. (2004). An approach to multimodal biomedical image registration utilizing particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):289–301.

Walters, D. C. and Sheble, G. B. (1993). Genetic algorithm solution of economic dispatch with valve point loading. *Power Systems, IEEE Transactions on*, 8(3):1325–1332.

Wang, C. and Huang, Y. (2010). Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37(4):2826–2837.

Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336.

Winston, M. L. (1991). *The biology of the honey bee*. Harvard University Press.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Woodward, J. (2010). Why classifying search algorithms is essential. *2010 International Conference on Progress in Informatics and Computing*.

Wright, A. H., Thompson, R. K., and Zhang, J. (2000). The computational complexity of nk fitness functions. *Evolutionary Computation, IEEE Transactions on*, 4(4):373–379.

Wright, S. (1931). Evolution in mendelian populations. *Genetics*, 16(2):97.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc of the 6th Int. Congr. of Genetics*, volume 1, pages 356–366.

Wright, S. and Nocedal, J. (1999). *Numerical optimization*, volume 2. Springer New York.

Yang, X.-S. (2009). Harmony search as a metaheuristic algorithm. In Geem, Z., editor, *Music-Inspired Harmony Search Algorithm*, volume 191 of *Studies in Computational Intelligence*, pages 1–14. Springer Berlin / Heidelberg.

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver press.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., and Nakanishi, Y. (2000). A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *Power Systems, IEEE Transactions on*, 15(4):1232–1239.

Yuan, B. and Gallagher, M. (2004). Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 172–181. Springer.

Zarei, O., Fesanghary, M., Farshi, B., Saffar, R. J., and Razfar, M. (2009). Optimization of multi-pass face-milling via harmony search algorithm. *Journal of materials processing technology*, 209(5):2386–2392.

Appendices

Appendix A

p-Values for Untuned Performance Data

This appendix includes the p-values of two-tailed, unpaired T-Tests for each algorithm, across each characteristic, for the complete characteristic value range. Significance suggests that the change in characteristic value is statistically significant. Values that do not have a significance level less than 5% (i.e. where the statistical tests suggest that the change in characteristic value has not generated statistically significantly different results) are highlighted in **bold.**

p-Values of the Untuned **Bees Algorithm** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	1.79E-58	5.61E-35	5.84E-58	5.12E-42	1.13E-71	4.34E-72	1.59E-87	1.06E-64	3.54E-51
1	1.79E-58	-	1.69E-06	**5.39E-01**	4.84E-04	1.52E-09	**1.20E-01**	3.07E-03	**5.25E-01**	3.73E-03
2	5.61E-35	1.69E-06	-	1.36E-07	**1.77E-01**	3.96E-23	1.08E-10	1.13E-15	9.18E-06	1.10E-11
3	5.84E-58	**5.39E-01**	1.36E-07	-	5.90E-05	5.24E-08	**3.72E-01**	2.44E-02	**2.07E-01**	1.94E-02
4	5.12E-42	4.84E-04	**1.77E-01**	5.90E-05	-	5.41E-19	2.79E-07	2.70E-11	2.32E-03	7.95E-09
5	1.13E-71	1.52E-09	3.96E-23	5.24E-08	5.41E-19	-	1.41E-06	1.74E-04	1.71E-11	4.52E-03
6	4.34E-72	**1.20E-01**	1.08E-10	**3.72E-01**	2.79E-07	1.41E-06	-	**1.63E-01**	2.27E-02	**1.02E-01**
7	1.59E-87	3.07E-03	1.13E-15	2.44E-02	2.70E-11	1.74E-04	**1.63E-01**	-	1.61E-04	**6.08E-01**
8	1.06E-64	**5.25E-01**	9.18E-06	**2.07E-01**	2.32E-03	1.71E-11	2.27E-02	1.61E-04	-	4.47E-04
9	3.54E-51	3.73E-03	1.10E-11	1.94E-02	7.95E-09	4.52E-03	**1.02E-01**	**6.08E-01**	4.47E-04	-

p-Values of the Untuned **Bees Algorithm** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	1.03E-267	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.03E-267	-	7.38E-186	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.38E-186	-	4.29E-82	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.29E-82	-	1.05E-245	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.05E-245	-	6.23E-34
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.23E-34	-

p-Values of the Untuned **Bees Algorithm** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	2.09E-25	1.00E-08	4.30E-03	**2.86E-01**	**2.63E-01**	**2.10E-01**	**3.77E-01**	7.70E-03	1.14E-03
20	2.09E-25	-	1.53E-04	1.51E-06	7.53E-17	3.42E-13	5.07E-12	1.10E-08	3.58E-15	3.92E-17
30	1.00E-08	1.53E-04	-	**1.19E-01**	4.95E-07	7.20E-06	1.39E-05	3.90E-04	4.42E-08	1.74E-09
40	4.30E-03	1.51E-06	1.19E-01	-	2.18E-03	3.89E-03	3.81E-03	1.78E-02	5.19E-05	5.13E-06
50	**2.86E-01**	7.53E-17	4.95E-07	2.18E-03	-	**8.31E-01**	**6.85E-01**	**8.33E-01**	**8.83E-02**	2.63E-02
60	**2.63E-01**	3.42E-13	7.20E-06	3.89E-03	**8.31E-01**	-	**8.50E-01**	**9.71E-01**	**1.63E-01**	**6.17E-02**
70	**2.10E-01**	5.07E-12	1.39E-05	3.81E-03	**6.85E-01**	**8.50E-01**	-	**9.00E-01**	**2.48E-01**	**1.08E-01**
80	**3.77E-01**	1.10E-08	3.90E-04	1.78E-02	**8.33E-01**	**9.71E-01**	**9.00E-01**	-	**2.44E-01**	**1.16E-01**
90	7.70E-03	3.58E-15	4.42E-08	5.19E-05	**8.83E-02**	**1.63E-01**	**2.48E-01**	**2.44E-01**	-	**6.63E-01**
100	1.14E-03	3.92E-17	1.74E-09	5.13E-06	2.63E-02	**6.17E-02**	**1.08E-01**	**1.16E-01**	**6.63E-01**	-

p-Values of the Untuned **Bees Algorithm** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	1.57E-125	4.79E-266	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	1.57E-125	-	1.64E-45	6.52E-121	3.91E-209	7.78E-274	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	4.79E-266	1.64E-45	-	8.09E-28	4.01E-81	1.09E-137	9.27E-193	5.63E-263	0.00E+00	0.00E+00
40	0.00E+00	6.52E-121	8.09E-28	-	7.78E-16	6.21E-49	1.77E-89	1.31E-136	1.51E-182	7.66E-228
50	0.00E+00	3.91E-209	4.01E-81	7.78E-16	-	1.52E-12	8.62E-38	5.09E-69	8.41E-107	9.68E-145
60	0.00E+00	7.78E-274	1.09E-137	6.21E-49	1.52E-12	-	3.51E-09	3.62E-25	4.92E-51	4.68E-79
70	0.00E+00	0.00E+00	9.27E-193	1.77E-89	8.62E-38	3.51E-09	-	2.33E-05	1.81E-19	6.07E-38
80	0.00E+00	0.00E+00	5.63E-263	1.31E-136	5.09E-69	3.62E-25	2.33E-05	-	5.40E-07	2.40E-19
90	0.00E+00	0.00E+00	0.00E+00	1.51E-182	8.41E-107	4.92E-51	1.81E-19	5.40E-07	-	7.53E-05
100	0.00E+00	0.00E+00	0.00E+00	7.66E-228	9.68E-145	4.68E-79	6.07E-38	2.40E-19	7.53E-05	-

p-Values of the Untuned **Bacterial Foraging Optimisation Algorithm** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	**6.89E-02**	5.42E-06	7.07E-13	1.84E-10	4.19E-11	3.86E-11	6.99E-10	6.34E-09	2.96E-24
1	**6.89E-02**	-	4.40E-03	1.98E-08	2.07E-06	6.42E-07	6.05E-07	6.65E-06	3.83E-05	1.50E-18
2	5.42E-06	4.40E-03	-	5.60E-03	**6.40E-02**	3.80E-02	3.73E-02	**1.20E-01**	**2.51E-01**	2.23E-09
3	7.07E-13	1.98E-08	5.60E-03	-	**3.29E-01**	**4.45E-01**	**4.47E-01**	**1.80E-01**	**7.68E-02**	1.18E-03
4	1.84E-10	2.07E-06	**6.40E-02**	**3.29E-01**	-	**8.25E-01**	**8.21E-01**	**7.28E-01**	**4.36E-01**	1.79E-05
5	4.19E-11	6.42E-07	3.80E-02	**4.45E-01**	**8.25E-01**	-	**9.96E-01**	**5.64E-01**	**3.12E-01**	4.20E-05
6	3.86E-11	6.05E-07	3.73E-02	**4.47E-01**	**8.21E-01**	**9.96E-01**	-	**5.61E-01**	**3.09E-01**	4.13E-05
7	6.99E-10	6.65E-06	**1.20E-01**	**1.80E-01**	**7.28E-01**	**5.64E-01**	**5.61E-01**	-	**6.61E-01**	2.08E-06
8	6.34E-09	3.83E-05	**2.51E-01**	**7.68E-02**	**4.36E-01**	**3.12E-01**	**3.09E-01**	**6.61E-01**	-	1.99E-07
9	2.96E-24	1.50E-18	2.23E-09	1.18E-03	1.79E-05	4.20E-05	4.13E-05	2.08E-06	1.99E-07	-

p-Values of the Untuned **Bacterial Foraging Optimisation Algorithm** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	0.00E+00	-	8.30E-246	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	8.30E-246	-	2.25E-65	4.18E-119	1.05E-136	3.08E-141	8.61E-144
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.25E-65	-	2.77E-28	3.90E-51	4.15E-58	2.35E-62
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.18E-119	2.77E-28	-	2.67E-12	1.43E-20	4.51E-27
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.05E-136	3.90E-51	2.67E-12	-	2.38E-04	2.67E-11
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.08E-141	4.15E-58	1.43E-20	2.38E-04	-	9.86E-04
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.61E-144	2.35E-62	4.51E-27	2.67E-11	9.86E-04	-

p-Values of the Untuned **Bacterial Foraging Optimisation Algorithm** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	1.27E-104	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	1.27E-104	-	2.10E-139	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	0.00E+00	2.10E-139	-	3.26E-108	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
40	0.00E+00	0.00E+00	3.26E-108	-	1.19E-72	1.97E-236	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	0.00E+00	0.00E+00	0.00E+00	1.19E-72	-	9.22E-51	2.30E-158	2.74E-290	0.00E+00	0.00E+00
60	0.00E+00	0.00E+00	0.00E+00	1.97E-236	9.22E-51	-	3.12E-32	5.51E-102	5.54E-207	0.00E+00
70	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.30E-158	3.12E-32	-	5.00E-22	5.46E-80	4.67E-159
80	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.74E-290	5.51E-102	5.00E-22	-	1.44E-20	1.61E-66
90	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.54E-207	5.46E-80	1.44E-20	-	2.29E-15
100	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.67E-159	1.61E-66	2.29E-15	-

p-Values of the Untuned **Bacterial Foraging Optimisation Algorithm** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	3.35E-67	8.50E-163	7.39E-271	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	3.35E-67	-	1.70E-24	1.21E-77	4.18E-131	3.11E-179	3.00E-227	4.59E-275	0.00E+00	0.00E+00
30	8.50E-163	1.70E-24	-	1.42E-17	1.67E-46	4.13E-77	4.36E-110	4.72E-145	1.69E-181	1.26E-211
40	7.39E-271	1.21E-77	1.42E-17	-	6.82E-09	5.90E-24	1.83E-43	1.67E-66	1.27E-91	1.47E-113
50	0.00E+00	4.18E-131	1.67E-46	6.82E-09	-	1.61E-05	8.04E-16	1.60E-30	5.98E-48	3.77E-64
60	0.00E+00	3.11E-179	4.13E-77	5.90E-24	1.61E-05	-	1.84E-04	6.88E-13	1.39E-24	2.05E-36
70	0.00E+00	3.00E-227	4.36E-110	1.83E-43	8.04E-16	1.84E-04	-	5.58E-04	8.37E-11	7.54E-19
80	0.00E+00	4.59E-275	4.72E-145	1.67E-66	1.60E-30	6.88E-13	5.58E-04	-	2.47E-03	6.74E-08
90	0.00E+00	0.00E+00	1.69E-181	1.27E-91	5.98E-48	1.39E-24	8.37E-11	2.47E-03	-	1.73E-02
100	0.00E+00	0.00E+00	1.26E-211	1.47E-113	3.77E-64	2.05E-36	7.54E-19	6.74E-08	1.73E-02	-

p-Values of the Untuned **Evolution Strategies** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	9.37E-140	3.50E-199	1.82E-228	5.82E-258	6.55E-267	8.32E-279	2.32E-295	5.44E-276	1.91E-283
1	9.37E-140	-	3.38E-09	4.19E-13	1.62E-21	6.95E-25	1.22E-26	4.74E-33	2.77E-24	7.73E-25
2	3.50E-199	3.38E-09	-	**2.31E-01**	5.39E-04	2.19E-05	5.05E-06	4.52E-09	6.04E-05	4.44E-05
3	1.82E-228	4.19E-13	**2.31E-01**	-	2.10E-02	1.87E-03	6.17E-04	2.02E-06	4.30E-03	3.49E-03
4	5.82E-258	1.62E-21	5.39E-04	2.10E-02	-	**4.20E-01**	**2.72E-01**	1.50E-02	**6.05E-01**	**5.71E-01**
5	6.55E-267	6.95E-25	2.19E-05	1.87E-03	**4.20E-01**	-	**7.75E-01**	**1.06E-01**	**7.65E-01**	**7.98E-01**
6	8.32E-279	1.22E-26	5.05E-06	6.17E-04	**2.72E-01**	**7.75E-01**	-	**1.79E-01**	**5.54E-01**	**5.82E-01**
7	2.32E-295	4.74E-33	4.52E-09	2.02E-06	1.50E-02	**1.06E-01**	**1.79E-01**	-	**5.23E-02**	**5.62E-02**
8	5.44E-276	2.77E-24	6.04E-05	4.30E-03	**6.05E-01**	**7.65E-01**	**5.54E-01**	**5.23E-02**	-	**9.64E-01**
9	1.91E-283	7.73E-25	4.44E-05	3.49E-03	**5.71E-01**	**7.98E-01**	**5.82E-01**	**5.62E-02**	**9.64E-01**	-

p-Values of the Untuned **Evolution Strategies** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	7.16E-49	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	7.16E-49	-	3.69E-189	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	3.69E-189	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	1.15E-205	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.15E-205	-	3.66E-259	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.66E-259	-	2.56E-120	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.56E-120	-	3.97E-197	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.97E-197	-	6.08E-100
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.08E-100	-

p-Values of the Untuned **Evolution Strategies** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	7.63E-06	2.08E-12	3.92E-20	4.68E-28	6.60E-35	5.76E-40	1.83E-46	8.70E-53	4.37E-60
20	7.63E-06	-	1.08E-02	2.57E-06	9.10E-11	5.57E-15	3.01E-18	1.47E-22	7.03E-27	5.09E-32
30	2.08E-12	1.08E-02	-	3.15E-02	8.69E-05	1.51E-07	7.95E-10	6.08E-13	3.60E-16	4.00E-20
40	3.92E-20	2.57E-06	3.15E-02	-	**7.66E-02**	1.97E-03	6.69E-05	4.82E-07	2.23E-09	2.35E-12
50	4.68E-28	9.10E-11	8.69E-05	**7.66E-02**	-	**1.86E-01**	2.68E-02	1.12E-03	2.67E-05	1.70E-07
60	6.60E-35	5.57E-15	1.51E-07	1.97E-03	**1.86E-01**	-	**3.73E-01**	**5.34E-02**	4.07E-03	9.60E-05
70	5.76E-40	3.01E-18	7.95E-10	6.69E-05	2.68E-02	**3.73E-01**	-	**2.99E-01**	4.79E-02	2.66E-03
80	1.83E-46	1.47E-22	6.08E-13	4.82E-07	1.12E-03	**5.34E-02**	**2.99E-01**	-	**3.48E-01**	4.92E-02
90	8.70E-53	7.03E-27	3.60E-16	2.23E-09	2.67E-05	4.07E-03	4.79E-02	**3.48E-01**	-	**3.04E-01**
100	4.37E-60	5.09E-32	4.00E-20	2.35E-12	1.70E-07	9.60E-05	2.66E-03	4.92E-02	**3.04E-01**	-

p-Values of the Untuned **Evolution Strategies** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	2.86E-03	5.41E-08	1.11E-10	3.47E-14	1.75E-16	5.81E-20	1.94E-22	1.47E-24	5.95E-28
20	2.86E-03	-	1.41E-02	5.60E-04	4.83E-06	1.76E-07	9.25E-10	1.97E-11	6.55E-13	2.53E-15
30	5.41E-08	1.41E-02	-	**3.25E-01**	3.57E-02	6.05E-03	2.76E-04	2.50E-05	2.71E-06	6.29E-08
40	1.11E-10	5.60E-04	**3.25E-01**	-	**2.63E-01**	**7.74E-02**	7.85E-03	1.21E-03	2.03E-04	9.22E-06
50	3.47E-14	4.83E-06	3.57E-02	**2.63E-01**	-	**5.19E-01**	**1.25E-01**	3.47E-02	9.64E-03	9.41E-04
60	1.75E-16	1.76E-07	6.05E-03	**7.74E-02**	**5.19E-01**	-	**3.73E-01**	**1.42E-01**	**5.19E-02**	7.73E-03
70	5.81E-20	9.25E-10	2.76E-04	7.85E-03	**1.25E-01**	**3.73E-01**	-	**5.65E-01**	**2.93E-01**	**7.64E-02**
80	1.94E-22	1.97E-11	2.50E-05	1.21E-03	3.47E-02	**1.42E-01**	**5.65E-01**	-	**6.32E-01**	**2.31E-01**
90	1.47E-24	6.55E-13	2.71E-06	2.03E-04	9.64E-03	**5.19E-02**	**2.93E-01**	**6.32E-01**	-	**4.71E-01**
100	5.95E-28	2.53E-15	6.29E-08	9.22E-06	9.41E-04	7.73E-03	**7.64E-02**	**2.31E-01**	**4.71E-01**	-

p-Values of the Untuned **Genetic Algorithm** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	9.55E-298	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
1	9.55E-298	-	**8.42E-01**	**3.87E-01**	4.13E-27	2.13E-08	8.87E-37	1.23E-44	1.20E-21	1.64E-15
2	0.00E+00	**8.42E-01**	-	**2.80E-01**	1.44E-26	5.24E-08	3.21E-36	4.44E-44	4.01E-21	5.02E-15
3	0.00E+00	**3.87E-01**	**2.80E-01**	-	5.29E-34	1.09E-11	1.84E-45	2.00E-54	7.50E-28	1.44E-20
4	0.00E+00	4.13E-27	1.44E-26	5.29E-34	-	2.77E-08	**7.73E-02**	1.26E-03	**1.20E-01**	9.55E-04
5	0.00E+00	2.13E-08	5.24E-08	1.09E-11	2.77E-08	-	9.70E-14	6.08E-19	3.74E-05	1.67E-02
6	0.00E+00	8.87E-37	3.21E-36	1.84E-45	**7.73E-02**	9.70E-14	-	**1.36E-01**	6.95E-04	2.16E-07
7	0.00E+00	1.23E-44	4.44E-44	2.00E-54	1.26E-03	6.08E-19	**1.36E-01**	-	1.09E-06	2.58E-11
8	0.00E+00	1.20E-21	4.01E-21	7.50E-28	**1.20E-01**	3.74E-05	6.95E-04	1.09E-06	-	**7.45E-02**
9	0.00E+00	1.64E-15	5.02E-15	1.44E-20	9.55E-04	1.67E-02	2.16E-07	2.58E-11	**7.45E-02**	-

p-Values of the Untuned **Genetic Algorithm** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	0.00E+00	-	1.50E-185	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	1.50E-185	-	1.11E-46	1.11E-85	1.96E-253	7.80E-242	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	1.11E-46	-	2.95E-07	2.61E-87	1.75E-78	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	1.11E-85	2.95E-07	-	1.73E-50	2.86E-43	1.37E-283	4.74E-270
7	0.00E+00	0.00E+00	0.00E+00	1.96E-253	2.61E-87	1.73E-50	-	**1.56E-01**	1.43E-94	3.90E-85
8	0.00E+00	0.00E+00	0.00E+00	7.80E-242	1.75E-78	2.86E-43	**1.56E-01**	-	3.44E-112	5.90E-102
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.37E-283	1.43E-94	3.44E-112	-	**1.54E-01**
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.74E-270	3.90E-85	5.90E-102	**1.54E-01**	-

p-Values of the Untuned **Genetic Algorithm** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	4.33E-92	7.24E-202	4.55E-304	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	4.33E-92	-	4.80E-31	1.48E-89	2.33E-148	1.72E-188	3.79E-234	1.29E-289	0.00E+00	0.00E+00
30	7.24E-202	4.80E-31	-	1.64E-18	1.12E-50	3.93E-76	4.16E-108	1.45E-149	2.04E-187	4.68E-216
40	4.55E-304	1.48E-89	1.64E-18	-	3.41E-10	1.00E-22	1.09E-41	2.44E-69	1.94E-96	6.78E-118
50	0.00E+00	2.33E-148	1.12E-50	3.41E-10	-	4.15E-04	3.45E-13	4.57E-30	1.11E-48	2.36E-64
60	0.00E+00	1.72E-188	3.93E-76	1.00E-22	4.15E-04	-	1.69E-04	2.77E-15	4.64E-29	2.09E-41
70	0.00E+00	3.79E-234	4.16E-108	1.09E-41	3.45E-13	1.69E-04	-	3.40E-05	1.02E-13	2.13E-22
80	0.00E+00	1.29E-289	1.45E-149	2.44E-69	4.57E-30	2.77E-15	3.40E-05	-	9.90E-04	2.21E-08
90	0.00E+00	0.00E+00	2.04E-187	1.94E-96	1.11E-48	4.64E-29	1.02E-13	9.90E-04	-	2.12E-02
100	0.00E+00	0.00E+00	4.68E-216	6.78E-118	2.36E-64	2.09E-41	2.13E-22	2.21E-08	2.12E-02	-

p-Values of the Untuned **Genetic Algorithm** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	8.49E-49	2.88E-101	1.22E-158	5.54E-195	5.03E-250	2.20E-278	0.00E+00	0.00E+00	0.00E+00
20	8.49E-49	-	5.51E-12	1.44E-36	1.76E-56	1.47E-89	1.67E-108	3.61E-133	3.92E-146	6.89E-175
30	2.88E-101	5.51E-12	-	7.47E-09	1.40E-19	1.39E-40	7.00E-54	1.04E-71	1.71E-81	1.08E-103
40	1.22E-158	1.44E-36	7.47E-09	-	9.51E-04	2.67E-14	1.66E-22	2.44E-34	3.11E-41	1.73E-57
50	5.54E-195	1.76E-56	1.40E-19	9.51E-04	-	1.71E-05	1.09E-10	5.30E-19	3.71E-24	8.08E-37
60	5.03E-250	1.47E-89	1.39E-40	2.67E-14	1.71E-05	-	3.02E-02	3.92E-06	4.75E-09	4.36E-17
70	2.20E-278	1.67E-108	7.00E-54	1.66E-22	1.09E-10	3.02E-02	-	1.47E-02	2.33E-04	4.78E-10
80	0.00E+00	3.61E-133	1.04E-71	2.44E-34	5.30E-19	3.92E-06	1.47E-02	-	**2.14E-01**	1.47E-04
90	0.00E+00	3.92E-146	1.71E-81	3.11E-41	3.71E-24	4.75E-09	2.33E-04	**2.14E-01**	-	1.07E-02
100	0.00E+00	6.89E-175	1.08E-103	1.73E-57	8.08E-37	4.36E-17	4.78E-10	1.47E-04	1.07E-02	-

p-Values of the Untuned **Harmony Search** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	5.02E-30	4.44E-33	2.63E-47	1.21E-49	2.87E-42	1.95E-73	1.41E-72	1.36E-54	5.85E-46
1	5.02E-30	-	**7.36E-01**	1.92E-02	3.69E-03	**2.65E-01**	4.88E-14	6.55E-12	7.47E-05	**5.31E-02**
2	4.44E-33	**7.36E-01**	-	5.45E-03	7.94E-04	**1.30E-01**	7.20E-16	1.29E-13	9.12E-06	1.82E-02
3	2.63E-47	1.92E-02	5.45E-03	-	**5.52E-01**	**1.94E-01**	4.47E-08	2.25E-06	**8.72E-02**	**6.62E-01**
4	1.21E-49	3.69E-03	7.94E-04	**5.52E-01**	-	**5.86E-02**	1.03E-06	3.63E-05	**2.65E-01**	**3.02E-01**
5	2.87E-42	**2.65E-01**	**1.30E-01**	**1.94E-01**	**5.86E-02**	-	1.67E-11	1.69E-09	2.68E-03	**3.87E-01**
6	1.95E-73	4.88E-14	7.20E-16	4.47E-08	1.03E-06	1.67E-11	-	**4.02E-01**	1.53E-04	3.50E-09
7	1.41E-72	6.55E-12	1.29E-13	2.25E-06	3.63E-05	1.69E-09	**4.02E-01**	-	2.64E-03	2.27E-07
8	1.36E-54	7.47E-05	9.12E-06	**8.72E-02**	**2.65E-01**	2.68E-03	1.53E-04	2.64E-03	-	3.16E-02
9	5.85E-46	**5.31E-02**	1.82E-02	**6.62E-01**	**3.02E-01**	**3.87E-01**	3.50E-09	2.27E-07	3.16E-02	-

p-Values of the Untuned **Harmony Search** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	2.71E-33	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	2.71E-33	-	1.72E-262	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	1.72E-262	-	3.47E-269	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	3.47E-269	-	4.76E-180	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	4.76E-180	-	3.27E-21	2.71E-277	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.27E-21	-	6.77E-160	3.01E-211	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.71E-277	6.77E-160	-	2.85E-02	2.55E-289	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.01E-211	2.85E-02	-	0.00E+00	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.55E-289	0.00E+00	-	4.89E-09
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.89E-09	-

p-Values of the Untuned **Harmony Search** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	2.38E-03	1.28E-28	1.44E-60	2.27E-113	1.50E-173	2.19E-231	2.32E-288	0.00E+00	0.00E+00
20	2.38E-03	-	8.42E-23	2.43E-55	2.56E-108	2.27E-168	3.59E-226	4.09E-283	0.00E+00	0.00E+00
30	1.28E-28	8.42E-23	-	4.40E-17	6.19E-62	6.84E-117	3.01E-172	1.56E-227	1.53E-268	0.00E+00
40	1.44E-60	2.43E-55	4.40E-17	-	8.52E-20	1.34E-58	3.37E-104	1.15E-152	2.69E-189	4.57E-235
50	2.27E-113	2.56E-108	6.19E-62	8.52E-20	-	3.17E-13	1.43E-39	2.36E-73	8.49E-101	6.18E-138
60	1.50E-173	2.27E-168	6.84E-117	1.34E-58	3.17E-13	-	2.13E-09	1.73E-28	1.20E-46	6.90E-74
70	2.19E-231	3.59E-226	3.01E-172	3.37E-104	1.43E-39	2.13E-09	-	3.07E-07	4.13E-17	6.44E-35
80	2.32E-288	4.09E-283	1.56E-227	1.15E-152	2.36E-73	1.73E-28	3.07E-07	-	1.00E-03	4.27E-13
90	0.00E+00	0.00E+00	1.53E-268	2.69E-189	8.49E-101	1.20E-46	4.13E-17	1.00E-03	-	6.91E-05
100	0.00E+00	0.00E+00	0.00E+00	4.57E-235	6.18E-138	6.90E-74	6.44E-35	4.27E-13	6.91E-05	-

p-Values of the Untuned **Harmony Search** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	3.61E-03	2.50E-08	5.16E-15	9.21E-19	8.86E-24	4.36E-30	2.43E-37	5.39E-43	9.44E-41
20	3.61E-03	-	6.56E-03	5.09E-07	1.10E-09	1.78E-13	2.10E-18	2.16E-24	2.40E-29	2.08E-27
30	2.50E-08	6.56E-03	-	2.04E-02	6.33E-04	2.30E-06	8.34E-10	2.46E-14	2.34E-18	8.63E-17
40	5.16E-15	5.09E-07	2.04E-02	-	**2.66E-01**	1.45E-02	1.08E-04	7.21E-08	6.01E-11	9.66E-10
50	9.21E-19	1.10E-09	6.33E-04	**2.66E-01**	-	**1.82E-01**	5.78E-03	1.85E-05	4.92E-08	5.12E-07
60	8.86E-24	1.78E-13	2.30E-06	1.45E-02	**1.82E-01**	-	**1.55E-01**	3.18E-03	3.53E-05	2.15E-04
70	4.36E-30	2.10E-18	8.34E-10	1.08E-04	5.78E-03	**1.55E-01**	-	**1.25E-01**	6.21E-03	2.16E-02
80	2.43E-37	2.16E-24	2.46E-14	7.21E-08	1.85E-05	3.18E-03	**1.25E-01**	-	**2.25E-01**	**4.40E-01**
90	5.39E-43	2.40E-29	2.34E-18	6.01E-11	4.92E-08	3.53E-05	6.21E-03	**2.25E-01**	-	**6.60E-01**
100	9.44E-41	2.08E-27	8.63E-17	9.66E-10	5.12E-07	2.15E-04	2.16E-02	**4.40E-01**	**6.60E-01**	-

p-Values of the Untuned **Particle Swarm Optimisation** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	2.13E-74	5.52E-85	3.53E-102	9.21E-119	4.78E-76	2.75E-124	3.29E-141	6.34E-107	1.70E-91
1	2.13E-74	-	**9.00E-01**	**8.68E-01**	2.78E-03	4.80E-04	7.71E-04	3.93E-08	**3.47E-01**	**1.57E-01**
2	5.52E-85	**9.00E-01**	-	**9.71E-01**	3.13E-03	1.65E-04	8.44E-04	3.24E-08	**4.02E-01**	**1.08E-01**
3	3.53E-102	**8.68E-01**	**9.71E-01**	-	2.28E-03	5.79E-05	5.58E-04	1.04E-08	**4.00E-01**	**8.37E-02**
4	9.21E-119	2.78E-03	3.13E-03	2.28E-03	-	3.04E-12	**7.06E-01**	8.17E-03	2.67E-02	2.02E-06
5	4.78E-76	4.80E-04	1.65E-04	5.79E-05	3.04E-12	-	1.36E-13	6.47E-22	1.28E-06	2.10E-02
6	2.75E-124	7.71E-04	8.44E-04	5.58E-04	**7.06E-01**	1.36E-13	-	2.26E-02	9.07E-03	2.43E-07
7	3.29E-141	3.93E-08	3.24E-08	1.04E-08	8.17E-03	6.47E-22	2.26E-02	-	1.04E-06	1.16E-13
8	6.34E-107	**3.47E-01**	**4.02E-01**	**4.00E-01**	2.67E-02	1.28E-06	9.07E-03	1.04E-06	-	1.03E-02
9	1.70E-91	**1.57E-01**	**1.08E-01**	**8.37E-02**	2.02E-06	2.10E-02	2.43E-07	1.16E-13	1.03E-02	-

p-Values of the Untuned **Particle Swarm Optimisation** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	6.82E-89	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.82E-89	-	9.92E-221	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	9.92E-221	-	2.14E-296	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	2.14E-296	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	3.22E-90	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.22E-90	-	8.70E-223	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.70E-223	-	3.23E-24	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.23E-24	-	1.57E-272	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.57E-272	-	2.02E-53
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.02E-53	-

p-Values of the Untuned **Particle Swarm Optimisation** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	0.00E+00	**1.58E-01**	8.68E-25	1.83E-76	1.55E-125	3.17E-165	6.54E-232	1.06E-300	0.00E+00
20	0.00E+00	-	1.67E-71	2.31E-138	1.19E-207	3.29E-264	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	**1.58E-01**	1.67E-71	-	1.29E-21	1.93E-65	3.87E-109	1.66E-145	1.46E-207	6.23E-273	0.00E+00
40	8.68E-25	2.31E-138	1.29E-21	-	3.41E-15	1.90E-40	2.60E-65	8.84E-112	8.06E-165	1.68E-200
50	1.83E-76	1.19E-207	1.93E-65	3.41E-15	-	3.06E-08	5.68E-21	1.71E-50	4.28E-89	8.61E-117
60	1.55E-125	3.29E-264	3.87E-109	1.90E-40	3.06E-08	-	1.02E-04	2.60E-21	2.25E-48	1.63E-69
70	3.17E-165	0.00E+00	1.66E-145	2.60E-65	5.68E-21	1.02E-04	-	2.16E-08	5.23E-27	2.92E-43
80	6.54E-232	0.00E+00	1.46E-207	8.84E-112	1.71E-50	2.60E-21	2.16E-08	-	2.13E-07	1.74E-16
90	1.06E-300	0.00E+00	6.23E-273	8.06E-165	4.28E-89	2.25E-48	5.23E-27	2.13E-07	-	2.23E-03
100	0.00E+00	0.00E+00	0.00E+00	1.68E-200	8.61E-117	1.63E-69	2.92E-43	1.74E-16	2.23E-03	-

p-Values of the Untuned **Particle Swarm Optimisation** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	2.66E-03	9.15E-13	6.52E-19	1.45E-23	1.13E-34	2.32E-37	5.74E-42	1.49E-46	3.01E-58
20	2.66E-03	-	2.95E-05	2.74E-09	1.41E-12	4.92E-21	4.19E-23	8.45E-27	1.76E-30	2.28E-40
30	9.15E-13	2.95E-05	-	**7.38E-02**	3.37E-03	1.15E-07	7.28E-09	3.67E-11	1.63E-13	1.90E-20
40	6.52E-19	2.74E-09	**7.38E-02**	-	**2.53E-01**	4.31E-04	6.34E-05	1.33E-06	2.24E-08	6.15E-14
50	1.45E-23	1.41E-12	3.37E-03	**2.53E-01**	-	1.71E-02	4.22E-03	2.16E-04	8.35E-06	1.76E-10
60	1.13E-34	4.92E-21	1.15E-07	4.31E-04	1.71E-02	-	**6.38E-01**	**1.90E-01**	3.92E-02	6.23E-05
70	2.32E-37	4.19E-23	7.28E-09	6.34E-05	4.22E-03	**6.38E-01**	-	**4.00E-01**	**1.11E-01**	3.96E-04
80	5.74E-42	8.45E-27	3.67E-11	1.33E-06	2.16E-04	**1.90E-01**	**4.00E-01**	-	**4.53E-01**	6.82E-03
90	1.49E-46	1.76E-30	1.63E-13	2.24E-08	8.35E-06	3.92E-02	**1.11E-01**	**4.53E-01**	-	4.97E-02
100	3.01E-58	2.28E-40	1.90E-20	6.15E-14	1.76E-10	6.23E-05	3.96E-04	6.82E-03	4.97E-02	-

p-Values of the Untuned **Stochastic Hill-Climbing** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	4.99E-40	9.23E-110	1.89E-46	3.93E-147	1.92E-128	3.69E-94	7.13E-162	2.14E-103	4.15E-82
1	4.99E-40	-	3.11E-21	**6.02E-01**	1.45E-36	4.73E-27	6.01E-12	8.37E-43	8.69E-15	2.15E-07
2	9.23E-110	3.11E-21	-	8.85E-21	6.12E-03	**3.98E-01**	1.49E-03	2.88E-04	1.50E-02	4.90E-07
3	1.89E-46	**6.02E-01**	8.85E-21	-	7.56E-37	6.72E-27	2.64E-11	1.66E-43	3.22E-14	1.02E-06
4	3.93E-147	1.45E-36	6.12E-03	7.56E-37	-	4.57E-02	3.82E-10	**3.83E-01**	3.32E-08	1.35E-16
5	1.92E-128	4.73E-27	**3.98E-01**	6.72E-27	4.57E-02	-	2.11E-05	3.53E-03	4.93E-04	4.49E-10
6	3.69E-94	6.01E-12	1.49E-03	2.64E-11	3.82E-10	2.11E-05	-	3.08E-13	**3.99E-01**	**5.26E-02**
7	7.13E-162	8.37E-43	2.88E-04	1.66E-43	**3.83E-01**	3.53E-03	3.08E-13	-	5.63E-11	7.48E-21
8	2.14E-103	8.69E-15	1.50E-02	3.22E-14	3.32E-08	4.93E-04	**3.99E-01**	5.63E-11	-	4.69E-03
9	4.15E-82	2.15E-07	4.90E-07	1.02E-06	1.35E-16	4.49E-10	**5.26E-02**	7.48E-21	4.69E-03	-

p-Values of the Untuned **Stochastic Hill-Climbing** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	1.19E-127	1.59E-210	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	1.19E-127	-	2.13E-08	2.21E-179	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	1.59E-210	2.13E-08	-	7.90E-126	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	2.21E-179	7.90E-126	-	8.16E-192	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	8.16E-192	-	1.71E-130	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.71E-130	-	1.63E-162	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.63E-162	-	8.65E-72	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.65E-72	-	2.63E-207	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.63E-207	-	7.73E-84
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.73E-84	-

p-Values of the Untuned **Stochastic Hill-Climbing** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	0.00E+00	-	3.50E-251	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	0.00E+00	3.50E-251	-	1.22E-86	4.27E-237	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
40	0.00E+00	0.00E+00	1.22E-86	-	1.95E-40	2.82E-146	3.87E-260	0.00E+00	0.00E+00	0.00E+00
50	0.00E+00	0.00E+00	4.27E-237	1.95E-40	-	4.47E-35	1.81E-98	8.57E-174	4.29E-259	0.00E+00
60	0.00E+00	0.00E+00	0.00E+00	2.82E-146	4.47E-35	-	1.82E-18	8.78E-57	2.66E-109	1.04E-165
70	0.00E+00	0.00E+00	0.00E+00	3.87E-260	1.81E-98	1.82E-18	-	1.05E-12	2.67E-41	4.98E-78
80	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.57E-174	8.78E-57	1.05E-12	-	2.43E-10	5.49E-31
90	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.29E-259	2.66E-109	2.67E-41	2.43E-10	-	1.50E-07
100	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.04E-165	4.98E-78	5.49E-31	1.50E-07	-

p-Values of the Untuned **Stochastic Hill-Climbing** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	2.20E-12	3.13E-27	6.33E-41	4.54E-53	9.50E-67	3.22E-76	4.47E-88	4.09E-100	1.82E-108
20	2.20E-12	-	1.40E-04	1.42E-10	6.02E-17	7.64E-25	1.08E-30	2.71E-38	2.75E-46	5.35E-52
30	3.13E-27	1.40E-04	-	9.11E-03	5.10E-06	8.93E-11	1.25E-14	6.86E-20	1.17E-25	6.61E-30
40	6.33E-41	1.42E-10	9.11E-03	-	**5.08E-02**	1.08E-04	3.41E-07	7.19E-11	3.98E-15	2.26E-18
50	4.54E-53	6.02E-17	5.10E-06	**5.08E-02**	-	**5.52E-02**	1.67E-03	5.13E-06	3.72E-09	1.17E-11
60	9.50E-67	7.64E-25	8.93E-11	1.08E-04	**5.52E-02**	-	**2.19E-01**	8.19E-03	6.87E-05	1.12E-06
70	3.22E-76	1.08E-30	1.25E-14	3.41E-07	1.67E-03	**2.19E-01**	-	**1.57E-01**	5.96E-03	2.75E-04
80	4.47E-88	2.71E-38	6.86E-20	7.19E-11	5.13E-06	8.19E-03	**1.57E-01**	-	**1.82E-01**	2.62E-02
90	4.09E-100	2.75E-46	1.17E-25	3.98E-15	3.72E-09	6.87E-05	5.96E-03	**1.82E-01**	-	**3.74E-01**
100	1.82E-108	5.35E-52	6.61E-30	2.26E-18	1.17E-11	1.12E-06	2.75E-04	2.62E-02	**3.74E-01**	-

p-Values of the Untuned **Random Search** results across **'number of local optima'** values.

	0	1	2	3	4	5	6	7	8	9
0	-	1.09E-13	7.68E-03	1.31E-04	**4.10E-01**	1.40E-07	2.72E-06	1.11E-08	1.31E-02	**1.00E-01**
1	1.09E-13	-	8.26E-07	5.11E-04	2.99E-11	4.88E-02	3.28E-03	**5.11E-02**	4.32E-07	4.99E-08
2	7.68E-03	8.26E-07	-	**1.92E-01**	**6.62E-02**	5.10E-03	3.96E-02	2.01E-03	**8.70E-01**	**4.02E-01**
3	1.31E-04	5.11E-04	**1.92E-01**	-	2.36E-03	**1.46E-01**	**5.08E-01**	**9.92E-02**	**1.46E-01**	4.26E-02
4	**4.10E-01**	2.99E-11	**6.62E-02**	2.36E-03	-	6.88E-06	1.09E-04	9.93E-07	**9.69E-02**	**3.81E-01**
5	1.40E-07	4.88E-02	5.10E-03	**1.46E-01**	6.88E-06	-	**3.91E-01**	**9.08E-01**	3.32E-03	6.04E-04
6	2.72E-06	3.28E-03	3.96E-02	**5.08E-01**	1.09E-04	**3.91E-01**	-	**3.05E-01**	2.74E-02	5.81E-03
7	1.11E-08	**5.11E-02**	2.01E-03	**9.92E-02**	9.93E-07	**9.08E-01**	**3.05E-01**	-	1.24E-03	1.94E-04
8	1.31E-02	4.32E-07	**8.70E-01**	**1.46E-01**	**9.69E-02**	3.32E-03	2.74E-02	1.24E-03	-	**4.97E-01**
9	**1.00E-01**	4.99E-08	**4.02E-01**	4.26E-02	**3.81E-01**	6.04E-04	5.81E-03	1.94E-04	**4.97E-01**	-

p-Values of the Untuned **Random Search** results across **'dimensions'** values.

	1	2	3	4	5	6	7	8	9	10
1	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	3.62E-201
10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.62E-201	-

p-Values of the Untuned **Random Search** results across **'boundary constraints'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	0.00E+00	-	1.20E-135	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	0.00E+00	1.20E-135	-	2.76E-66	3.96E-176	3.65E-276	0.00E+00	0.00E+00	0.00E+00	0.00E+00
40	0.00E+00	0.00E+00	2.76E-66	-	1.46E-38	2.19E-111	1.79E-188	2.67E-258	0.00E+00	0.00E+00
50	0.00E+00	0.00E+00	3.96E-176	1.46E-38	-	4.12E-25	7.13E-76	7.17E-135	3.25E-193	2.44E-247
60	0.00E+00	0.00E+00	3.65E-276	2.19E-111	4.12E-25	-	9.34E-18	8.36E-55	1.01E-100	6.03E-149
70	0.00E+00	0.00E+00	0.00E+00	1.79E-188	7.13E-76	9.34E-18	-	2.68E-13	2.21E-41	8.02E-78
80	0.00E+00	0.00E+00	0.00E+00	2.67E-258	7.17E-135	8.36E-55	2.68E-13	-	2.13E-10	2.27E-32
90	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.25E-193	1.01E-100	2.21E-41	2.13E-10	-	1.94E-08
100	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.44E-247	6.03E-149	8.02E-78	2.27E-32	1.94E-08	-

p-Values of the Untuned **Random Search** results across **'smoothness coefficient'** values.

	10	20	30	40	50	60	70	80	90	100
10	-	3.11E-214	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	3.11E-214	-	1.43E-81	1.47E-215	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
30	0.00E+00	1.43E-81	-	4.89E-42	1.05E-121	4.88E-209	1.41E-292	0.00E+00	0.00E+00	0.00E+00
40	0.00E+00	1.47E-215	4.89E-42	-	2.33E-25	1.50E-76	1.55E-137	4.07E-200	1.86E-260	0.00E+00
50	0.00E+00	0.00E+00	1.05E-121	2.33E-25	-	6.25E-17	6.13E-52	6.51E-96	3.43E-143	1.12E-190
60	0.00E+00	0.00E+00	4.88E-209	1.50E-76	6.25E-17	-	4.11E-12	3.25E-37	6.41E-70	2.83E-106
70	0.00E+00	0.00E+00	1.41E-292	1.55E-137	6.13E-52	4.11E-12	-	3.91E-09	7.72E-28	7.66E-53
80	0.00E+00	0.00E+00	0.00E+00	4.07E-200	6.51E-96	3.25E-37	3.91E-09	-	3.48E-07	1.49E-21
90	0.00E+00	0.00E+00	0.00E+00	1.86E-260	3.43E-143	6.41E-70	7.72E-28	3.48E-07	-	7.81E-06
100	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.12E-190	2.83E-106	7.66E-53	1.49E-21	7.81E-06	-

p-Values of the Untuned **Bees Algorithm** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	6.04E-05	7.88E-16	3.76E-31	8.74E-54
0.3	6.04E-05	-	2.63E-05	5.32E-17	1.44E-35
0.5	7.88E-16	2.63E-05	-	3.45E-06	5.90E-19
0.7	3.76E-31	5.32E-17	3.45E-06	-	2.76E-05
0.9	8.74E-54	1.44E-35	5.90E-19	2.76E-05	-

p-Values of the Untuned **Bacterial Foraging Optimisation Algorithm** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	**7.91E-02**	5.62E-08	1.12E-19	4.70E-45
0.3	**7.91E-02**	-	1.89E-04	7.47E-14	6.58E-37
0.5	5.62E-08	1.89E-04	-	1.30E-04	1.38E-20
0.7	1.12E-19	7.47E-14	1.30E-04	-	1.63E-08
0.9	4.70E-45	6.58E-37	1.38E-20	1.63E-08	-

p-Values of the Untuned **Evolution Strategy** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	**6.13E-01**	6.08E-03	1.67E-07	4.14E-18
0.3	**6.13E-01**	-	2.10E-02	9.66E-07	1.72E-17
0.5	6.08E-03	2.10E-02	-	8.17E-03	1.35E-10
0.7	1.67E-07	9.66E-07	8.17E-03	-	1.01E-04
0.9	4.14E-18	1.72E-17	1.35E-10	1.01E-04	-

p-Values of the Untuned **Genetic Algorithm** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	7.66E-05	5.47E-08	6.73E-08	1.21E-02
0.3	7.66E-05	-	**1.71E-01**	**2.38E-01**	**5.32E-02**
0.5	5.47E-08	**1.71E-01**	-	**8.06E-01**	3.87E-04
0.7	6.73E-08	**2.38E-01**	**8.06E-01**	-	5.68E-04
0.9	1.21E-02	**5.32E-02**	3.87E-04	5.68E-04	-

p-Values of the Untuned **Harmony Search** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	1.16E-03	1.52E-10	1.11E-15	3.31E-16
0.3	1.16E-03	-	2.06E-03	1.39E-05	7.98E-05
0.5	1.52E-10	2.06E-03	-	**2.93E-01**	**7.51E-01**
0.7	1.11E-15	1.39E-05	**2.93E-01**	-	**3.73E-01**
0.9	3.31E-16	7.98E-05	**7.51E-01**	**3.73E-01**	-

p-Values of the Untuned **Particle Swarm Optimisation** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	6.78E-03	6.56E-06	1.88E-07	3.72E-07
0.3	6.78E-03	-	**8.26E-02**	2.10E-02	4.25E-02
0.5	6.56E-06	**8.26E-02**	-	**6.11E-01**	**8.94E-01**
0.7	1.88E-07	2.10E-02	**6.11E-01**	-	**6.74E-01**
0.9	3.72E-07	4.25E-02	**8.94E-01**	**6.74E-01**	-

p-Values of the Untuned **Stochastic Hill-Climbing** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	2.76E-02	2.74E-13	5.84E-38	1.28E-87
0.3	2.76E-02	-	1.34E-07	1.20E-28	3.37E-76
0.5	2.74E-13	1.34E-07	-	2.50E-09	1.45E-42
0.7	5.84E-38	1.20E-28	2.50E-09	-	2.20E-15
0.9	1.28E-87	3.37E-76	1.45E-42	2.20E-15	-

p-Values of the Untuned **Random Search** results across **'ratio of local optima to global optimum'** values.

	0.1	0.3	0.5	0.7	0.9
0.1	-	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
0.3	**1.00E+00**	-	**1.00E+00**	**1.00E+00**	**1.00E+00**
0.5	**1.00E+00**	**1.00E+00**	-	**1.00E+00**	**1.00E+00**
0.7	**1.00E+00**	**1.00E+00**	**1.00E+00**	-	**1.00E+00**
0.9	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	-

Appendix B

Tuned Parameter Configurations

This appendix includes the complete parameter configurations for all six nature-inspired algorithms, after the conclusion of the F-Racing process.

Best found **Bacterial Foraging Optimisation** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Pop. Size	Step Size	Repro Steps	Chem Steps	Swim Length	Elim. Prob.	d_attr	w_attr	h_rep	w_rep
0	9	1.42	9	36	4	0.01	0.10	0.36	0.14	0.21
1	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
2	49	1.74	6	65	4	0.78	0.94	1.26	1.32	1.93
3	49	1.74	6	65	4	0.78	0.94	1.26	1.32	1.93
4	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
5	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
6	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
7	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
8	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
9	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86

Best found **Bacterial Foraging Optimisation** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dims.	Pop. Size	Step Size	Repro Steps	Chem Steps	Swim Length	Elim. Prob.	d_attr	w_attr	h_rep	w_rep
1	34	0.31	8	39	5	0.39	0.11	0.07	0.20	1.06
2	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
3	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
4	22	2.83	8	75	7	0.87	0.69	1.32	0.98	0.62
5	22	2.83	8	75	7	0.87	0.69	1.32	0.98	0.62
6	33	3.46	2	83	6	0.67	0.07	1.47	1.07	1.34
7	33	3.46	2	83	6	0.67	0.07	1.47	1.07	1.34
8	33	3.46	2	83	6	0.67	0.07	1.47	1.07	1.34
9	33	3.46	2	83	6	0.67	0.07	1.47	1.07	1.34
10	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77

Best found **Bacterial Foraging Optimisation** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Pop. Size	Step Size	Repro Steps	Chem Steps	Swim Length	Elim. Prob.	d_attr	w_attr	h_rep	w_rep
0.1	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
0.2	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
0.3	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
0.4	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
0.5	49	1.74	6	65	4	0.78	0.94	1.26	1.32	1.93
0.6	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
0.7	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
0.8	49	1.74	6	65	4	0.78	0.94	1.26	1.32	1.93
0.9	49	1.74	6	65	4	0.78	0.94	1.26	1.32	1.93

Best found **Bacterial Foraging Optimisation** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Constraint	Pop. Size	Step Size	Repro Steps	Chem Steps	Swim Length	Elim. Prob.	d_attr	w_attr	h_rep	w_rep
10	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
20	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
30	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
40	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
50	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
60	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
70	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
80	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
90	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
100	32	6.29	6	70	1	0.26	0.03	0.44	0.07	1.95

Best found **Bacterial Foraging Optimisation** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness	Pop. Size	Step Size	Repro Steps	Chem Steps	Swim Length	Elim. Prob.	d_attr	w_attr	h_rep	w_rep
10	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
20	11	1.93	4	37	4	0.96	0.04	0.12	0.22	1.86
30	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
40	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
50	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
60	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
70	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
80	21	2.97	8	80	7	0.59	0.74	0.95	1.15	1.77
90	53	0.92	7	36	9	0.22	0.57	1.66	1.06	1.90
100	32	6.29	6	70	1	0.26	0.03	0.44	0.07	1.95

Best found **Bees algorithm** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Num Bees	Num Sites	Elite Sites	Patch Size	Elite Bees	Other Bees
0	108	43	9	0.369758663	9	3
1	108	43	9	0.369758663	9	3
2	108	43	9	0.369758663	9	3
3	108	43	9	0.369758663	9	3
4	108	43	9	0.369758663	9	3
5	108	43	9	0.369758663	9	3
6	108	43	9	0.369758663	9	3
7	108	43	9	0.369758663	9	3
8	108	43	9	0.369758663	9	3
9	108	43	9	0.369758663	9	3

Best found **Bees algorithm** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Num Bees	Num Sites	Elite Sites	Patch Size	Elite Bees	Other Bees
1	13	11	1	0.092645364	6	3
2	108	43	9	0.369758663	9	3
3	13	11	1	0.092645364	6	3
4	97	10	5	1.569463987	35	7
5	97	10	5	1.569463987	35	7
6	97	10	5	1.569463987	35	7
7	97	10	5	1.569463987	35	7
8	97	10	5	1.569463987	35	7
9	97	10	5	1.569463987	35	7
10	97	10	5	1.569463987	35	7

Best found **Bees algorithm** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Num Bees	Num Sites	Elite Sites	Patch Size	Elite Bees	Other Bees
0.1	13	11	1	0.092645364	6	3
0.2	13	11	1	0.092645364	6	3
0.3	135	56	7	0.060748275	4	1
0.4	135	56	7	0.060748275	4	1
0.5	108	43	9	0.369758663	9	3
0.6	13	11	1	0.092645364	6	3
0.7	13	11	1	0.092645364	6	3
0.8	13	11	1	0.092645364	6	3
0.9	135	56	7	0.060748275	4	1

Best found **Bees algorithm** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Num Bees	Num Sites	Elite Sites	Patch Size	Elite Bees	Other Bees
10	13	11	1	0.092645364	6	3
20	13	11	1	0.092645364	6	3
30	108	43	9	0.369758663	9	3
40	13	11	1	0.092645364	6	3
50	13	11	1	0.092645364	6	3
60	135	56	7	0.060748275	4	1
70	13	11	1	0.092645364	6	3
80	13	11	1	0.092645364	6	3
90	13	11	1	0.092645364	6	3
100	13	11	1	0.092645364	6	3

Best found **Bees algorithm** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Num Bees	Num Sites	Elite Sites	Patch Size	Elite Bees	Other Bees
10	13	11	1	0.092645364	6	3
20	13	11	1	0.092645364	6	3
30	135	56	7	0.060748275	4	1
40	13	11	1	0.092645364	6	3
50	13	11	1	0.092645364	6	3
60	13	11	1	0.092645364	6	3
70	13	11	1	0.092645364	6	3
80	13	11	1	0.092645364	6	3
90	13	11	1	0.092645364	6	3
100	135	56	7	0.060748275	4	1

Best found **Evolution Strategies** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Population Size	Number of Children
0	5	2
1	84	38
2	94	92
3	78	61
4	90	88
5	97	91
6	88	80
7	98	92
8	86	84
9	90	76

Best found **Evolution Strategies** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Population Size	Number of Children
1	86	42
2	94	92
3	91	33
4	8	6
5	47	34
6	65	14
7	49	26
8	100	23
9	41	10
10	29	19

Best found **Evolution Strategies** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Population Size	Number of Children
0.1	98	83
0.2	96	52
0.3	98	65
0.4	71	49
0.5	94	92
0.6	90	88
0.7	54	32
0.8	99	84
0.9	65	63

Best found **Evolution Strategies** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Population Size	Number of Children
10	90	80
20	99	95
30	94	92
40	91	56
50	88	60
60	50	26
70	95	43
80	98	86
90	80	55
100	80	61

Best found **Evolution Strategies** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Population Size	Number of Children
10	59	42
20	98	66
30	95	81
40	87	57
50	77	61
60	81	52
70	72	67
80	95	92
90	71	65
100	97	70

Best found **Genetic Algorithm** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Bits Per Dimension	Population Size	Crossover Chance	Mutation Chance
0	112	227	0.89331401	0.039475776
1	50	191	0.052663763	0.051492694
2	50	191	0.052663763	0.051492694
3	50	191	0.052663763	0.051492694
4	50	191	0.052663763	0.051492694
5	58	243	0.294221318	0.064708823
6	50	191	0.052663763	0.051492694
7	58	243	0.294221318	0.064708823
8	50	191	0.052663763	0.051492694
9	50	191	0.052663763	0.051492694

Best found **Genetic Algorithm** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Bits Per Dimension	Population Size	Crossover Chance	Mutation Chance
1	112	227	0.89331401	0.039475776
2	50	191	0.052663763	0.051492694
3	50	191	0.052663763	0.051492694
4	24	221	0.162093651	0.020109175
5	24	221	0.162093651	0.020109175
6	24	221	0.162093651	0.020109175
7	24	221	0.162093651	0.020109175
8	118	234	0.73610524	0.01388368
9	42	238	0.26651476	0.014879794
10	87	198	0.810279389	0.011274037

Best found **Genetic Algorithm** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Bits Per Dimension	Population Size	Crossover Chance	Mutation Chance
0.1	112	227	0.89331401	0.039475776
0.2	112	227	0.89331401	0.039475776
0.3	58	243	0.294221318	0.064708823
0.4	50	191	0.052663763	0.051492694
0.5	50	191	0.052663763	0.051492694
0.6	50	191	0.052663763	0.051492694
0.7	58	243	0.294221318	0.064708823
0.8	50	191	0.052663763	0.051492694
0.9	58	243	0.294221318	0.064708823

Best found **Genetic Algorithm** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Bits Per Dimension	Population Size	Crossover Chance	Mutation Chance
10	112	227	0.89331401	0.039475776
20	112	227	0.89331401	0.039475776
30	50	191	0.052663763	0.051492694
40	50	191	0.052663763	0.051492694
50	50	191	0.052663763	0.051492694
60	50	191	0.052663763	0.051492694
70	58	243	0.294221318	0.064708823
80	50	191	0.052663763	0.051492694
90	111	234	0.294221318	0.064708823
100	58	243	0.294221318	0.064708823

Best found **Genetic Algorithm** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Bits Per Dimension	Population Size	Crossover Chance	Mutation Chance
10	50	191	0.052663763	0.051492694
20	50	191	0.052663763	0.051492694
30	50	191	0.052663763	0.051492694
40	50	191	0.052663763	0.051492694
50	58	243	0.294221318	0.064708823
60	58	243	0.294221318	0.064708823
70	50	191	0.052663763	0.051492694
80	58	243	0.294221318	0.064708823
90	58	243	0.294221318	0.064708823
100	58	243	0.294221318	0.064708823

Best found **Harmony Search** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Memory Size	Consideration Rate	Adjustment Rate	Range
0	61	0.522940998	0.361405982	0.02686435
1	106	0.594938408	0.241110384	0.018747458
2	38	0.58512274	0.119371722	0.041315692
3	106	0.594938408	0.241110384	0.018747458
4	61	0.522940998	0.361405982	0.02686435
5	61	0.522940998	0.361405982	0.02686435
6	61	0.522940998	0.361405982	0.02686435
7	61	0.522940998	0.361405982	0.02686435
8	61	0.522940998	0.361405982	0.02686435
9	61	0.522940998	0.361405982	0.02686435

Best found **Harmony Search** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Memory Size	Consideration Rate	Adjustment Rate	Range
1	130	0.432413263	0.874201206	0.028459016
2	184	0.556426407	0.716830534	0.071875809
3	61	0.522940998	0.361405982	0.02686435
4	2	0.448981918	0.233814989	2.321073256
5	2	0.448981918	0.233814989	2.321073256
6	2	0.448981918	0.233814989	2.321073256
7	110	0.92527344	0.07807093	0.658149792
8	110	0.92527344	0.07807093	0.658149792
9	1	0.885656584	0.049619375	13.6648148
10	46	0.97452116	0.376275727	0.657018353

Best found **Harmony Search** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Memory Size	Consideration Rate	Adjustment Rate	Range
0.1	28	0.565082183	0.753860822	0.185551216
0.2	139	0.514641192	0.146908781	0.05675386
0.3	58	0.650182633	0.110585621	0.078901814
0.4	121	0.864147077	0.476129388	0.022442529
0.5	38	0.58512274	0.119371722	0.041315692
0.6	101	0.679165869	0.245691678	0.412331579
0.7	101	0.470476815	0.453765271	0.052569231
0.8	150	0.479425233	0.241871074	0.171575701
0.9	124	0.90089792	0.243349931	0.513187438

Best found **Harmony Search** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Memory Size	Consideration Rate	Adjustment Rate	Range
10	128	0.957354308	0.659578273	0.050942962
20	29	0.783704707	0.923156195	0.051675768
30	38	0.58512274	0.119371722	0.041315692
40	16	0.694140202	0.377870876	0.530903565
50	21	0.489778087	0.112641114	0.242556224
60	91	0.468706715	0.531816432	0.658843222
70	77	0.505062739	0.251941775	0.91844589
80	160	0.614716811	0.369631483	1.387309958
90	54	0.358521337	0.8326134	0.267781121
100	21	0.499594146	0.897245597	0.939176505

Best found **Harmony Search** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Memory Size	Consideration Rate	Adjustment Rate	Range
10	26	0.631935846	0.428215124	0.104445226
20	49	0.564021172	0.408215819	0.513555773
30	41	0.666408546	0.657556726	0.091365312
40	215	0.739566626	0.490167435	0.344831117
50	218	0.817422046	0.665090485	0.202434332
60	19	0.408050675	0.90265884	0.151420418
70	119	0.472864497	0.63521141	0.16787071
80	79	0.577783975	0.378317385	0.388423256
90	25	0.377836064	0.660113722	0.859643887
100	24	0.3403247	0.110464253	0.30650878

Best found **Particle Swarm Optimisation** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Population Size	Maximum Velocity	Personal Best Weight	Global Best Weight
0	35	0.118316261	3.605395251	0.71177902
1	35	2.808731759	3.66606597	0.154792824
2	33	2.463940386	3.611013362	1.282039577
3	36	29.06492595	2.08535032	0.552357329
4	34	3.02885491	3.978257479	0.799743307
5	34	3.02885491	3.978257479	0.799743307
6	35	2.808731759	3.66606597	0.154792824
7	36	29.06492595	2.08535032	0.552357329
8	34	3.02885491	3.978257479	0.799743307
9	34	3.02885491	3.978257479	0.799743307

Best found **Particle Swarm Optimisation** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Population Size	Maximum Velocity	Personal Best Weight	Global Best Weight
1	35	0.118316261	3.605395251	0.71177902
2	29	20.1549109	1.774936052	3.370919399
3	35	2.808731759	3.66606597	0.154792824
4	35	2.808731759	3.66606597	0.154792824
5	35	2.808731759	3.66606597	0.154792824
6	35	2.808731759	3.66606597	0.154792824
7	35	2.808731759	3.66606597	0.154792824
8	35	2.808731759	3.66606597	0.154792824
9	35	2.808731759	3.66606597	0.154792824
10	35	2.808731759	3.66606597	0.154792824

Best found **Particle Swarm Optimisation** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Population Size	Maximum Velocity	Personal Best Weight	Global Best Weight
0.1	33	2.463940386	3.611013362	1.282039577
0.2	33	2.463940386	3.611013362	1.282039577
0.3	37	29.45821309	2.115935328	2.386092361
0.4	34	3.02885491	3.978257479	0.799743307
0.5	34	3.02885491	3.978257479	0.799743307
0.6	34	3.02885491	3.978257479	0.799743307
0.7	34	3.02885491	3.978257479	0.799743307
0.8	34	3.02885491	3.978257479	0.799743307
0.9	34	3.02885491	3.978257479	0.799743307

Best found **Particle Swarm Optimisation** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Population Size	Maximum Velocity	Personal Best Weight	Global Best Weight
10	36	0.391012818	2.321464975	2.720356571
20	29	20.1549109	1.774936052	3.370919399
30	33	2.463940386	3.611013362	1.282039577
40	25	39.15311803	2.002129093	2.906120979
50	34	1.382664143	3.476081122	0.234521474
60	33	3.254507546	3.115778704	0.126284832
70	33	3.254507546	3.115778704	0.126284832
80	33	3.254507546	3.115778704	0.126284832
90	33	3.254507546	3.115778704	0.126284832
100	33	3.254507546	3.115778704	0.126284832

Best found **Particle Swarm Optimisation** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Population Size	Maximum Velocity	Personal Best Weight	Global Best Weight
10	35	1.531565201	3.919579713	0.400994985
20	35	1.531565201	3.919579713	0.400994985
30	34	1.382664143	3.476081122	0.234521474
40	34	1.382664143	3.476081122	0.234521474
50	35	1.531565201	3.919579713	0.400994985
60	34	1.382664143	3.476081122	0.234521474
70	35	1.531565201	3.919579713	0.400994985
80	33	3.254507546	3.115778704	0.126284832
90	34	1.382664143	3.476081122	0.234521474
100	33	3.254507546	3.115778704	0.126284832

Best found **Stochastic Hill Climbing** parameter configurations as the **'number of local optima'** characteristic of the fitness landscape changes.

Curves	Neighbourhood Size
0	15.75686374
1	13.57374624
2	16.19450084
3	19.00952014
4	21.18612078
5	16.05003502
6	16.05003502
7	16.05003502
8	15.82807067
9	15.8692342

Best found **Stochastic Hill Climbing** parameter configurations as the **'dimensions'** characteristic of the fitness landscape changes.

Dimensions	Neighbourhood Size
1	10.20867642
2	16.19450084
3	14.64677071
4	14.1933461
5	7.07795945
6	6.491854803
7	6.904605006
8	5.895783364
9	5.672806904
10	4.552774007

Best found **Stochastic Hill Climbing** parameter configurations as the **'ratio of local optima to global optimum'** characteristic of the fitness landscape changes.

Ratio	Neighbourhood Size
0.1	14.86405173
0.2	15.23523792
0.3	15.57096951
0.4	15.68013727
0.5	16.19450084
0.6	16.4536341
0.7	16.37892354
0.8	16.9389166
0.9	17.37204712

Best found **Stochastic Hill Climbing** parameter configurations as the **'boundary constraints'** characteristic of the fitness landscape changes.

Boundary Constraints	Neighbourhood Size
10	4.773961648
20	10.31928631
30	16.19450084
40	22.34021944
50	28.32556178
60	33.58694741
70	40.62169524
80	45.57602055
90	52.98651083
100	59.17991985

Best found **Stochastic Hill Climbing** parameter configurations as the **'smoothness coefficient'** characteristic of the fitness landscape changes.

Smoothness Coefficient	Neighbourhood Size
10	15.72213281
20	16.75488111
30	17.40161114
40	17.06126638
50	17.471051
60	17.70691888
70	16.71207413
80	17.8007314
90	17.6963382
100	17.95917259

Appendix C

Untuned and Tuned Performance Data

This appendix includes the complete performance data, in terms of average error and standard deviation, of all six nature-inspired algorithms, as characteristics varied across the given ranges.

The average error of the selected algorithms as the number of local optima in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
# Optima	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
1	9.22E-04	6.64E-06	1.40E-01	5.84E-03	1.06E-02	7.15E-03	7.76E-03	4.29E-03	3.03E-05	5.71E-09	7.03E-04	1.90E-03	1.71E-01	2.49E-02
2	1.40E-03	8.43E-06	1.32E-01	2.67E-03	6.70E-02	6.43E-02	8.37E-02	1.86E-02	8.96E-03	7.34E-05	2.58E-02	2.73E-03	2.40E-01	7.72E-02
3	1.23E-03	7.87E-06	1.21E-01	2.07E-03	8.54E-02	8.28E-02	8.43E-02	1.67E-02	8.60E-03	3.02E-05	2.61E-02	1.90E-02	2.89E-01	9.11E-02
4	1.42E-03	8.58E-06	1.12E-01	2.23E-03	8.93E-02	8.25E-02	8.15E-02	1.67E-02	1.16E-02	2.35E-05	2.61E-02	4.44E-03	2.42E-01	8.42E-02
5	1.27E-03	8.56E-06	1.15E-01	2.37E-03	9.68E-02	9.22E-02	1.14E-01	2.65E-02	1.22E-02	1.59E-05	3.15E-02	1.39E-02	3.03E-01	9.52E-02
6	1.69E-03	9.50E-06	1.14E-01	2.73E-03	9.95E-02	9.12E-02	9.88E-02	4.47E-03	1.02E-02	7.60E-09	1.99E-02	2.12E-02	2.93E-01	6.53E-02
7	1.45E-03	9.29E-06	1.14E-01	2.50E-03	1.00E-01	9.00E-02	1.19E-01	2.58E-02	1.86E-02	7.11E-05	3.22E-02	9.41E-03	2.73E-01	6.39E-02
8	1.51E-03	9.76E-06	1.16E-01	2.88E-03	1.05E-01	9.31E-02	1.23E-01	4.84E-03	1.75E-02	2.31E-07	3.66E-02	8.99E-03	3.07E-01	8.09E-02
9	1.37E-03	9.29E-06	1.17E-01	2.66E-03	9.85E-02	9.01E-02	1.10E-01	1.83E-02	1.36E-02	3.63E-07	2.76E-02	3.14E-02	2.77E-01	7.68E-02
10	1.53E-03	9.88E-06	1.02E-01	5.55E-03	9.86E-02	8.56E-02	1.05E-01	1.82E-02	1.11E-02	2.81E-06	2.34E-02	2.58E-02	2.65E-01	6.41E-02

The standard deviation of the selected algorithms as the number of local optima in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
# Optima	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
1	1.21E-03	9.02E-06	3.12E-01	7.07E-02	7.69E-02	5.33E-02	4.60E-02	5.51E-03	1.50E-04	5.20E-08	9.67E-04	4.35E-02	3.74E-01	1.34E-01
2	2.66E-03	1.32E-05	2.78E-01	5.89E-03	2.08E-01	2.03E-01	1.94E-01	6.99E-02	7.82E-02	1.41E-03	1.37E-01	4.13E-02	3.62E-01	2.19E-01
3	2.18E-03	1.20E-05	2.57E-01	4.40E-03	2.31E-01	2.29E-01	1.91E-01	6.58E-02	7.13E-02	1.91E-03	1.29E-01	1.09E-01	3.73E-01	2.36E-01
4	2.83E-03	1.43E-05	2.34E-01	1.03E-02	2.26E-01	2.18E-01	1.74E-01	6.47E-02	7.95E-02	6.91E-04	1.17E-01	4.99E-02	3.31E-01	2.04E-01
5	2.27E-03	1.41E-05	2.32E-01	5.02E-03	2.33E-01	2.27E-01	2.04E-01	8.10E-02	8.20E-02	4.96E-04	1.31E-01	9.09E-02	3.43E-01	2.21E-01
6	4.08E-03	1.79E-05	2.29E-01	6.43E-03	2.36E-01	2.25E-01	1.86E-01	6.87E-03	7.40E-02	9.85E-08	1.03E-01	1.10E-01	3.39E-01	1.86E-01
7	2.68E-03	1.81E-05	2.28E-01	4.56E-03	2.33E-01	2.19E-01	2.01E-01	7.67E-02	1.02E-01	5.35E-03	1.31E-01	7.09E-02	3.26E-01	1.79E-01
8	2.66E-03	1.53E-05	2.24E-01	6.43E-03	2.37E-01	2.22E-01	2.06E-01	6.99E-03	9.60E-02	1.52E-05	1.40E-01	6.43E-02	3.30E-01	2.01E-01
9	2.35E-03	1.57E-05	2.24E-01	5.25E-03	2.29E-01	2.18E-01	1.91E-01	6.37E-02	8.65E-02	1.77E-05	1.21E-01	1.30E-01	3.17E-01	1.89E-01
10	3.84E-03	2.44E-05	2.09E-01	4.78E-02	2.26E-01	2.11E-01	1.84E-01	5.91E-02	7.73E-02	9.29E-05	1.11E-01	1.17E-01	3.12E-01	1.73E-01

The average error of the selected algorithms as the number of dimensions in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Dimensions	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
1	4.15E-07	9.47E-11	7.95E-04	9.66E-04	4.28E-02	4.18E-02	1.09E-04	2.20E-06	9.37E-07	3.41E-11	6.74E-05	4.43E-03	1.72E-01	4.02E-02
2	1.23E-03	7.87E-06	1.21E-01	2.09E-03	8.54E-02	8.28E-02	8.43E-02	1.67E-02	8.60E-03	4.70E-05	2.61E-02	1.87E-02	2.89E-01	9.11E-02
3	1.55E-02	5.09E-03	5.83E-01	4.23E-02	1.89E-01	1.92E-01	2.67E-01	1.14E-01	9.19E-02	5.17E-02	1.10E-01	1.78E-02	3.18E-01	2.08E-01
4	6.75E-02	1.34E-02	8.81E-01	1.53E-01	3.89E-01	3.95E-01	4.04E-01	2.61E-01	2.31E-01	1.09E-01	2.51E-01	4.89E-02	4.34E-01	4.18E-01
5	1.41E-01	5.21E-02	9.72E-01	3.28E-01	5.58E-01	5.57E-01	4.76E-01	5.09E-01	3.70E-01	2.67E-01	4.21E-01	9.97E-02	5.67E-01	6.37E-01
6	2.29E-01	8.10E-02	9.92E-01	4.86E-01	6.74E-01	6.81E-01	5.01E-01	6.93E-01	4.17E-01	3.65E-01	5.08E-01	1.48E-01	6.62E-01	7.65E-01
7	3.17E-01	1.13E-01	9.98E-01	6.40E-01	7.87E-01	7.90E-01	5.73E-01	8.38E-01	5.47E-01	4.81E-01	6.34E-01	2.22E-01	7.52E-01	8.70E-01
8	3.78E-01	1.26E-01	9.99E-01	7.47E-01	8.49E-01	8.50E-01	5.67E-01	9.11E-01	5.57E-01	4.05E-01	6.69E-01	2.53E-01	8.02E-01	9.19E-01
9	4.87E-01	1.59E-01	1.00E+00	8.57E-01	9.10E-01	9.09E-01	6.67E-01	9.66E-01	6.98E-01	5.10E-01	7.73E-01	3.65E-01	8.70E-01	9.62E-01
10	5.25E-01	1.81E-01	1.00E+00	9.14E-01	9.41E-01	9.42E-01	6.61E-01	9.85E-01	7.16E-01	4.38E-01	8.09E-01	3.95E-01	9.02E-01	9.79E-01

The standard deviation of the selected algorithms as the number of dimensions in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Dimensions	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
1	1.62E-06	4.17E-10	2.44E-02	2.57E-02	1.74E-01	1.71E-01	2.06E-03	1.19E-05	3.05E-06	1.22E-10	6.72E-03	5.53E-02	3.08E-01	1.63E-01
2	2.18E-03	1.20E-05	2.57E-01	4.51E-03	2.31E-01	2.29E-01	1.91E-01	6.58E-02	7.13E-02	1.75E-03	1.29E-01	1.09E-01	3.73E-01	2.36E-01
3	1.90E-02	5.78E-02	3.90E-01	6.65E-02	2.63E-01	2.66E-01	3.06E-01	1.98E-01	2.24E-01	1.46E-01	2.27E-01	9.49E-02	3.49E-01	2.37E-01
4	8.22E-02	6.12E-02	2.44E-01	1.67E-01	3.01E-01	3.04E-01	3.54E-01	1.74E-01	3.21E-01	2.40E-01	2.99E-01	1.36E-01	3.37E-01	2.52E-01
5	1.51E-01	1.44E-01	1.07E-01	2.34E-01	2.82E-01	2.78E-01	3.52E-01	2.04E-01	3.57E-01	3.37E-01	3.12E-01	1.81E-01	2.91E-01	2.29E-01
6	1.95E-01	1.76E-01	4.60E-02	2.12E-01	2.48E-01	2.45E-01	3.42E-01	1.78E-01	3.42E-01	3.55E-01	2.91E-01	2.04E-01	2.53E-01	1.87E-01
7	2.26E-01	2.06E-01	2.01E-02	2.16E-01	2.06E-01	2.02E-01	3.40E-01	1.38E-01	3.35E-01	3.86E-01	2.62E-01	2.47E-01	2.12E-01	1.36E-01
8	2.23E-01	2.08E-01	9.28E-03	1.75E-01	1.64E-01	1.65E-01	3.28E-01	9.21E-02	2.84E-01	3.76E-01	2.27E-01	2.49E-01	1.75E-01	1.04E-01
9	2.34E-01	2.21E-01	6.34E-03	1.24E-01	1.17E-01	1.19E-01	2.98E-01	4.83E-02	2.33E-01	3.74E-01	1.80E-01	2.75E-01	1.30E-01	5.91E-02
10	2.05E-01	2.21E-01	1.69E-03	9.49E-02	8.69E-02	8.50E-02	2.91E-01	2.84E-02	2.00E-01	3.66E-01	1.47E-01	2.62E-01	1.03E-01	3.62E-02

The average error of the selected algorithms as the ratio of local optima to the global optimum in the fitness landscape changes, both pre- and post-tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Ratio	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
0.1	1.02E-03	5.04E-07	1.43E-01	6.19E-03	9.54E-02	9.31E-02	6.92E-02	4.58E-03	2.98E-03	1.35E-07	1.91E-02	2.60E-02	3.32E-01	9.38E-02
0.3	1.11E-03	1.03E-06	1.36E-01	2.25E-03	9.34E-02	9.04E-02	8.05E-02	3.88E-03	5.65E-03	5.10E-06	2.42E-02	3.29E-03	3.19E-01	1.00E-01
0.5	1.23E-03	7.87E-06	1.21E-01	2.12E-03	8.54E-02	8.28E-02	8.43E-02	1.66E-02	8.60E-03	6.13E-05	2.75E-02	1.92E-02	2.89E-01	9.11E-02
0.7	1.40E-03	4.25E-04	1.08E-01	2.22E-03	7.72E-02	7.91E-02	8.36E-02	3.99E-03	9.62E-03	9.13E-05	2.84E-02	6.96E-03	2.59E-01	8.29E-02
0.9	1.59E-03	1.18E-06	9.08E-02	2.15E-03	6.64E-02	6.25E-02	7.55E-02	3.93E-03	8.88E-03	8.46E-03	2.77E-02	6.86E-03	2.23E-01	7.20E-02

The standard deviation of the selected algorithms as the ratio of local optima to the global optimum in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Ratio	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
0.1	1.43E-03	2.23E-06	3.10E-01	6.52E-02	2.77E-01	2.74E-01	2.00E-01	6.55E-03	5.12E-02	6.99E-07	1.30E-01	1.53E-01	4.52E-01	2.73E-01
0.3	1.78E-03	1.48E-06	2.89E-01	4.77E-03	2.58E-01	2.54E-01	2.02E-01	5.52E-03	6.42E-02	2.25E-04	1.36E-01	4.97E-02	4.13E-01	2.64E-01
0.5	2.18E-03	1.20E-05	2.57E-01	4.51E-03	2.31E-01	2.29E-01	1.91E-01	6.56E-02	7.13E-02	3.27E-03	1.33E-01	1.11E-01	3.73E-01	2.36E-01
0.7	2.94E-03	1.34E-02	2.30E-01	4.33E-03	2.07E-01	2.10E-01	1.77E-01	5.69E-03	6.50E-02	1.77E-03	1.23E-01	6.49E-02	3.39E-01	2.13E-01
0.9	3.37E-03	8.82E-06	2.05E-01	7.56E-03	1.84E-01	1.80E-01	1.54E-01	5.69E-03	5.10E-02	5.06E-02	1.09E-01	5.92E-02	3.11E-01	1.89E-01

The average error of the selected algorithms as the boundary constraint range in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Range	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
10	1.08E-03	8.73E-07	6.04E-03	1.03E-03	6.37E-02	5.51E-02	1.99E-02	1.43E-03	5.18E-05	1.05E-04	2.81E-02	3.95E-02	5.29E-03	1.65E-02
20	1.35E-03	7.45E-05	4.41E-02	1.77E-03	7.72E-02	7.17E-02	5.58E-02	3.00E-03	6.35E-04	2.76E-04	2.35E-03	1.46E-02	1.29E-01	9.63E-03
30	1.23E-03	7.87E-06	1.21E-01	2.13E-03	8.54E-02	8.28E-02	8.43E-02	1.67E-02	7.44E-03	7.32E-06	2.61E-02	1.89E-02	2.89E-01	9.11E-02
40	1.17E-03	4.67E-04	2.14E-01	6.55E-03	9.26E-02	9.42E-02	1.10E-01	2.52E-02	1.83E-02	3.07E-04	4.66E-02	5.06E-02	3.98E-01	1.98E-01
50	1.05E-03	7.55E-04	3.02E-01	1.46E-02	9.86E-02	9.32E-02	1.30E-01	3.29E-02	3.56E-02	1.66E-05	6.77E-02	2.49E-02	4.75E-01	2.99E-01
60	1.04E-03	1.57E-06	3.82E-01	7.13E-03	1.03E-01	1.00E-01	1.42E-01	4.00E-02	5.32E-02	4.51E-04	8.48E-02	8.47E-03	5.48E-01	3.68E-01
70	1.03E-03	2.07E-03	4.47E-01	5.27E-02	1.06E-01	1.03E-01	1.56E-01	1.07E-02	7.00E-02	7.34E-04	9.78E-02	1.02E-02	5.99E-01	4.47E-01
80	1.04E-03	2.48E-03	5.01E-01	2.22E-02	1.10E-01	1.07E-01	1.71E-01	5.02E-02	8.59E-02	3.97E-03	1.18E-01	1.35E-02	6.39E-01	4.93E-01
90	9.63E-04	3.01E-03	5.53E-01	1.05E-01	1.13E-01	1.08E-01	1.84E-01	1.43E-02	9.68E-02	5.67E-04	1.38E-01	1.70E-02	6.75E-01	5.43E-01
100	9.37E-04	3.25E-03	5.96E-01	7.31E-03	1.17E-01	1.13E-01	1.93E-01	1.56E-02	1.11E-01	1.24E-03	1.50E-01	2.09E-02	7.04E-01	5.87E-01

The standard deviation of the selected algorithms as the boundary constraint range in the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Range	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
10	1.31E-03	1.14E-06	6.11E-02	1.05E-02	2.06E-01	1.91E-01	8.38E-02	2.05E-03	2.31E-04	8.02E-03	5.40E-02	1.57E-01	6.16E-03	1.97E-02
20	2.14E-03	7.38E-03	1.63E-01	2.06E-02	2.23E-01	2.18E-01	1.54E-01	4.23E-03	1.92E-02	1.18E-02	2.84E-02	9.51E-02	2.79E-01	1.19E-02
30	2.18E-03	1.20E-05	2.57E-01	5.00E-03	2.31E-01	2.29E-01	1.91E-01	6.58E-02	6.64E-02	2.04E-04	1.29E-01	1.09E-01	3.73E-01	2.36E-01
40	2.86E-03	1.66E-02	3.26E-01	4.81E-02	2.38E-01	2.40E-01	2.20E-01	8.91E-02	1.10E-01	1.24E-02	1.72E-01	1.77E-01	4.03E-01	3.30E-01
50	2.80E-03	2.11E-02	3.64E-01	7.96E-02	2.42E-01	2.36E-01	2.39E-01	1.04E-01	1.55E-01	2.29E-04	2.06E-01	1.29E-01	4.15E-01	3.80E-01
60	3.58E-03	2.70E-05	3.86E-01	4.68E-02	2.45E-01	2.41E-01	2.48E-01	1.18E-01	1.86E-01	1.17E-02	2.29E-01	7.51E-02	4.12E-01	4.01E-01
70	4.01E-03	3.58E-02	3.93E-01	1.69E-01	2.46E-01	2.41E-01	2.59E-01	1.63E-02	2.10E-01	1.45E-02	2.44E-01	8.22E-02	4.07E-01	4.14E-01
80	4.91E-03	3.89E-02	3.95E-01	1.07E-01	2.48E-01	2.40E-01	2.71E-01	1.36E-01	2.29E-01	3.42E-02	2.64E-01	9.44E-02	4.02E-01	4.20E-01
90	4.36E-03	4.32E-02	3.91E-01	2.39E-01	2.49E-01	2.42E-01	2.79E-01	2.24E-02	2.39E-01	1.07E-02	2.82E-01	1.07E-01	3.93E-01	4.19E-01
100	4.36E-03	4.48E-02	3.84E-01	3.47E-02	2.50E-01	2.44E-01	2.85E-01	2.36E-02	2.54E-01	2.48E-02	2.92E-01	1.20E-01	3.85E-01	4.12E-01

The average error of the selected algorithms as the smoothness coefficient of the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Coefficient	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
10	8.32E-04	1.77E-04	9.03E-02	2.55E-03	8.52E-02	7.73E-02	6.36E-02	1.14E-02	6.78E-03	6.10E-08	2.28E-02	6.02E-03	2.70E-01	8.47E-02
20	1.66E-03	3.54E-04	1.54E-01	8.11E-03	9.52E-02	8.82E-02	1.03E-01	2.09E-02	9.65E-03	1.80E-06	2.82E-02	9.33E-03	3.07E-01	9.63E-02
30	2.39E-03	1.80E-06	1.98E-01	5.38E-03	1.04E-01	9.31E-02	1.25E-01	2.62E-02	1.28E-02	3.01E-04	3.66E-02	8.38E-03	3.28E-01	1.04E-01
40	3.13E-03	6.57E-04	2.39E-01	6.22E-03	1.07E-01	1.00E-01	1.45E-01	3.29E-02	1.58E-02	1.38E-03	4.05E-02	1.07E-02	3.42E-01	1.08E-01
50	3.79E-03	7.93E-04	2.69E-01	7.72E-03	1.11E-01	1.02E-01	1.56E-01	1.07E-02	1.74E-02	4.13E-03	4.31E-02	1.71E-02	3.53E-01	1.13E-01
60	4.45E-03	8.69E-04	2.92E-01	1.15E-02	1.14E-01	1.10E-01	1.73E-01	1.20E-02	1.94E-02	2.19E-06	4.87E-02	1.49E-02	3.64E-01	1.16E-01
70	5.07E-03	1.13E-03	3.12E-01	1.04E-02	1.17E-01	1.11E-01	1.81E-01	4.43E-02	2.16E-02	1.55E-04	4.99E-02	2.10E-02	3.71E-01	1.19E-01
80	5.55E-03	8.82E-04	3.31E-01	1.52E-02	1.19E-01	1.11E-01	1.91E-01	1.51E-02	2.41E-02	1.89E-04	5.20E-02	1.41E-02	3.79E-01	1.23E-01
90	6.14E-03	1.06E-03	3.48E-01	1.31E-02	1.20E-01	1.12E-01	1.96E-01	1.62E-02	2.63E-02	1.50E-05	5.39E-02	2.31E-02	3.87E-01	1.25E-01
100	6.64E-03	4.18E-06	3.62E-01	1.88E-02	1.23E-01	1.14E-01	2.06E-01	1.71E-02	2.55E-02	1.08E-05	5.90E-02	1.68E-02	3.92E-01	1.28E-01

The standard deviation of the selected algorithms as the smoothness coefficient of the fitness landscape changes, both pre- and post- tuning.

	BA		BFOA		ES		GA		HS		PSO		SHC	
Coefficient	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned	Untuned	Tuned
10	1.50E-03	1.03E-02	2.19E-01	3.43E-02	2.33E-01	2.24E-01	1.65E-01	5.23E-02	6.32E-02	4.73E-07	1.21E-01	6.47E-02	3.65E-01	2.30E-01
20	3.10E-03	1.46E-02	2.90E-01	6.80E-02	2.43E-01	2.33E-01	2.12E-01	7.49E-02	7.58E-02	1.33E-05	1.33E-01	8.04E-02	3.82E-01	2.41E-01
30	4.10E-03	2.36E-06	3.26E-01	1.04E-02	2.50E-01	2.35E-01	2.32E-01	8.63E-02	8.77E-02	1.13E-02	1.51E-01	7.58E-02	3.91E-01	2.46E-01
40	5.43E-03	1.99E-02	3.53E-01	2.96E-02	2.50E-01	2.41E-01	2.49E-01	1.02E-01	9.69E-02	1.94E-02	1.58E-01	8.51E-02	3.97E-01	2.47E-01
50	6.04E-03	2.21E-02	3.69E-01	3.42E-02	2.52E-01	2.38E-01	2.60E-01	1.33E-02	1.02E-01	4.00E-02	1.63E-01	1.07E-01	4.02E-01	2.49E-01
60	7.05E-03	2.34E-02	3.80E-01	1.77E-02	2.53E-01	2.47E-01	2.71E-01	1.43E-02	1.08E-01	3.29E-06	1.73E-01	9.98E-02	4.05E-01	2.50E-01
70	7.93E-03	2.68E-02	3.89E-01	3.88E-02	2.53E-01	2.45E-01	2.77E-01	1.19E-01	1.13E-01	6.53E-03	1.74E-01	1.18E-01	4.07E-01	2.51E-01
80	7.94E-03	2.37E-02	3.98E-01	2.49E-02	2.53E-01	2.40E-01	2.82E-01	1.79E-02	1.20E-01	7.99E-03	1.77E-01	9.27E-02	4.09E-01	2.52E-01
90	8.76E-03	2.58E-02	4.02E-01	4.30E-02	2.54E-01	2.40E-01	2.86E-01	1.83E-02	1.26E-01	3.38E-05	1.80E-01	1.24E-01	4.11E-01	2.52E-01
100	9.24E-03	4.85E-06	4.07E-01	2.70E-02	2.55E-01	2.39E-01	2.92E-01	1.91E-02	1.24E-01	2.67E-05	1.89E-01	1.01E-01	4.12E-01	2.54E-01

Appendix D

Published Work

This appendix contains two papers which have been published in peer-reviewed conference proceedings during this research. These conference publications contain much of the work presented in Chapter 4 and Chapter 5.

Details are as follows:

Crossley, M., Nisbet, A., and Amos, M. (2013). Fitness landscape-based characterisation of nature-inspired algorithms. In Tomassini, M., Antonioni, A., Daolio, F., and Buesser, P., editors, *Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA '13), Lausanne, Switzerland, April 4-6, 2013. Lecture Notes in Computer Science, Vol. 7824,* **pages 110-119. Springer.**

Crossley, M., Nisbet, A., and Amos, M. (2013). Quantifying the impact of parameter tuning on nature-inspired algorithms. In Lio, P., Miglino, O., Nicosia, G., Nolfi, S., and Pavone, M., editors, *Advances in Artificial Life, ECAL 2013,* **pages 925-932. MIT Press.**

Fitness Landscape-Based Characterisation of Nature-Inspired Algorithms

Matthew Crossley, Andy Nisbet, and Martyn Amos *

School of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, Manchester M15GD, UK
{m.crossley,a.nisbet,m.amos}@mmu.ac.uk

Abstract. A significant challenge in nature-inspired algorithmics is the identification of specific characteristics of problems that make them harder (or easier) to solve using specific methods. The hope is that, by identifying these characteristics, we may more easily predict which algorithms are best-suited to problems sharing certain features. Here, we approach this problem using fitness landscape analysis. Techniques already exist for measuring the "difficulty" of specific landscapes, but these are often designed solely with evolutionary algorithms in mind, and are generally specific to discrete optimisation. In this paper we develop an approach for comparing a wide range of continuous optimisation algorithms. Using a fitness landscape generation technique, we compare six different nature-inspired algorithms and identify which methods perform best on landscapes exhibiting specific features.

1 Introduction

Inspired by the foundational work of Wolpert and Macready [1], practitioners have long sought to better understand the relationship between problems and solution methods (i.e., algorithms). Here, we are particularly interested in the question of which algorithm is *best-suited* to a particular problem, and the process of addressing this has been described by some as a "black-art" [2].

Although theoretical studies in this area have yielded useful results, the *experimental analysis* of algorithms is receiving increasing attention. As Morgan and Gallagher point out [3], this approach is *scalable* in that it readily admits newly-described algorithms, and it is now an area of research that is supported by a number of high-profile competitions and libraries of benchmark test problems.

The fundamental properties of a problem's *search landscape* underpin much work in experimental analysis, and the use of landscape/test case generators [3–7] has been proposed as one way in which we might effectively assess algorithms against problem instances.

In this paper we examine six different nature-inspired algorithms by testing them against a number of different randomized landscapes with several different

* Matthew Crossley is supported by a Ph.D. studentship from the Dalton Research Institute, MMU. The authors thank David Corne for useful discussions.

properties (e.g., ruggedness). This gives a much richer picture of their relative strengths and weaknesses, compared to simply using the "difficulty" of a landscape [8].

The rest of the paper is organized as follows: in Section 2 we give a brief overview of previous work, before describing our testing methodology in Section 3. We then present our experimental results in Section 4, before concluding in Section 5 with a discussion of our findings.

2 Previous work

The use of algorithms inspired by physical or natural processes is now well-established in the field of optimisation [9]. As the number of such algorithms grows year-on-year, there is a pressing need to better understand their properties, in order that practitioners may make informed decisions about which method is best-suited to a particular problem, under certain conditions. Although analytical methods have been successfully applied to nature-inspired methods [10] [11], their "real world" applicability is not clear, as they often rely on significant assumptions and/or simplifications.

In what follows, we take an *experimental* approach [12] to studying the selected algorithms, using an established landscape generation technique [4]. As Morgan and Gallagher observe, "In a general sense, an algorithm can be expected to perform well if the assumptions that it makes, either explicit or implicit, are well-matched to the properties of the search landscape or solution space of a given problem or set of problems" [3]. We therefore seek to investigate the performance of several algorithms on a number of types of *fitness landscape* with specific properties or characteristics. This approach is preferred by Hooker to the use of benchmark problems, because the latter "differ in so many respects that it is rarely evident why some are harder than others, and they may yet fail to vary over parameters that are key determinants of performance. It is better generate problems in a controlled fashion... The goal is not to generate realistic problems, which random generation cannot do, but to generate several problem sets, each of which is homogeneous with respect to characteristics that are likely to affect performance" [13].

The fitness landscape approach has been successfully applied to the study of various nature-inspired algorithms [14–16]. Indeed, to our knowledge, landscape analysis of nature-inspired algorithms has been largely *restricted* to evolutionary methods. In this paper we broaden this work *considerably*, by considering several classes of natural algorithms (social, evolutionary and physical). Overall, we study six different nature-inspired methods, as well as stochastic hill-climbing as a baseline algorithm. Our empirical approach is informed by previous work [17] [18], which emphasises the need to establish a rigorous framework for experimental algorithmics. In the next Section, we describe in detail our methodology.

3 Methodology

3.1 Algorithm selection

We select, for comparison, a number of nature-inspired algorithms that are commonly applied to continuous function optimisation. These may be classified [19] as either *social*, *evolutionary* or *physical*. The social algorithms we select are Bacterial Foraging Optimisation Algorithm (BFOA) [20], Bees Algorithm (BA) [21], and Particle Swarm Optimisation (PSO) [22]. The evolutionary algorithms selected are Genetic Algorithms (GA) [23] and Evolution Strategies (ES) [24], and physical algorithms are represented by Harmony Search (HS) [25]. We also include random search (RS) and stochastic hill climbing (SHC) as "baseline" algorithms.

We note that the references supplied above for each algorithm may serve simply as an example of their *application*, rather than their precise *implementation*. In terms of implementation, we heed the observation that "Ideally, competing algorithms would be coded by the same expert programmer and run on the same test problems on the same computer configuration" [12]. With that in mind, we use only implementations provided by Brownlee to accompany [26]. The limited space available prevents a complete description of each algorithm, but full implementation details are in [26], which is freely available and contains the source code used here.

3.2 Optimisation problem characteristics

As Morgan and Gallagher explain [3], their Max-Set of Gaussians (MSG) method [4] is a "randomised landscape generator that specifies test problems as a weighted sum of Gaussian functions. By specifying the number of Gaussians and the mean and covariance parameters for each component, a variety of test landscape instances can be generated. The topological properties of the landscapes are intuitively related to (and vary smoothly with) the parameters of the generator." By manipulating these parameters, we obtain landscapes with different *characteristics*. This allows us to investigate the performance of our selected algorithms on landscapes with different features, and to identify which characteristics pose the greatest challenge. As Morgan and Gallagher observe, "Different problem types have their own characteristics, however it is usually the case that complementary insights into algorithm behaviour result from conducting larger experimental studies using a variety of different problem types" [3]. We now describe the different characteristics (corresponding to problem types) under study in this paper.

Ruggedness of a landscape is often linked to its difficulty [8], and factors affecting this include (1) the *number* of local optima [27], and (2) *ratio* of the fitness value of local optima to the global optimal value [28] [14]. Other significant factors concern (3) *dimensionality* [29] (that is, the number of variables in the objective function), (4) *boundary constraints* (that is, the limits imposed on the value of a variable) [30], and (5) *smoothness* of each Gaussian curve (effectively

Table 1. A summary of the ranges selected for the characteristics in our fitness space (F)

Characteristic	Min	Step	Max	Default
Number of local optima	0	1	9	3
Ratio of local optima to global optimum	0.1	0.2	0.9	0.5
Dimensionality	1	1	10	2
Boundary constraints	10	10	100	30
Smoothness	10	10	100	15

the gradient) used to generate the landscape [31] - a smaller value indicates a smoother gradient. A summary of the ranges selected for each characteristic is given in Table 1.

3.3 Performance measurement

In terms of *performance metrics*, we abstract away from algorithm-specific measures, due to the diverse range of methods selected. The following metrics are applied: **(1) Accuracy**: We define this as the mean absolute error of the best solution found on a given set of landscape characteristics, over a number of runs $(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x}))$ (where X is the set of best solutions found, n is the number of runs performed and $\bar{x}$ is the known optimum). This is the most commonly-used assessment metric for optimisation algorithms [4]. The generation technique we use creates landscapes with a known global optimum, in this case zero. **(2) Variance of final solutions**: A measure of variation in best solutions found across differently seeded runs. We use the standard deviation of the best solutions of all runs on a given set of landscape characteristics, defined as $(\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2)^{\frac{1}{2}}$ (where X is our data set, n is the size of the data set and $\bar{x}$ is the mean average). **(3) Success rate**: We measure this as the frequency with which differently-seeded runs of an algorithm are able to find a solution within a specified distance from the optimum [32]. We keep the success tolerance relatively low (error less than 1.0×10^{-4}) in order to ensure that we capture the change in success rate of algorithms which perform poorly.

3.4 Experimental setup

In order to generate the landscapes, we used the Matlab code supplied with [4]. All landscapes were generated using default parameters of three curves, two dimensions, 0.5 average ratio of local minima to global minimum, 30 units in

each dimension with a smoothness coefficient of 15), with only the parameter under investigation changing for each experiment. We ran each algorithm 100 times on each landscape in the set of landscapes generated for each particular characteristic value (when investigating smoothness, for example, we generated 10 different landscapes (smoothness = 10 ... 100), and ran each algorithm 100 times on each landscape).

Parameterisation of algorithms provides a significant challenge when evaluating performance. Our aim is not to perform "competitive testing" [13], but to establish general performance *profiles* for different algorithms over different types of problem. As such, we use the so-called "vanilla" implementation of each algorithm, with general-purpose settings taken from [4]. Where an algorithm has a "population size" parameter, we use a value of 50; where an algorithm has a "range" or "velocity" parameter, we use a value of 10.

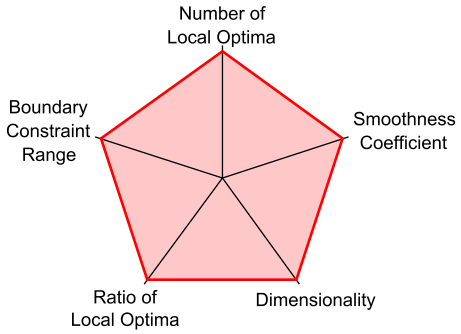
Termination criteria were also standardised. The most objective criterion is the number of objective function evaluations. This means each algorithm has access to the same amount of information from the landscape, and the same amount of *feedback* on potential solutions. Experimentally we determined that the selected algorithms generally converged within 20,000 objective function calculations, so this was used as the termination criterion. The code used for all algorithms, as well as datasets and the landscape generator, is available on request from the authors.

4 Results

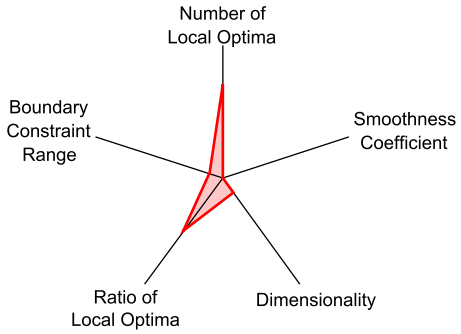
Space prevents a detailed presentation of full experimental plots, but these are available from the project website[1]. To summarise, we plot the *resilience* of each algorithm to changing landscape characteristics, in the form of a radar plot in Figure 1. To assess the resilience of an algorithm we use the standard deviation of the average error across all values of a landscape characteristic, which we normalise on a per-characteristic basis. This "ranking" shows which algorithms do *not* show performance variability versus those which *are* heavily influenced by a characteristic. BFOA shows large deviations in average error for boundary constraint range, smoothness coefficient changes and dimensionality, indicating that BFOA is an algorithm heavily dependent on the landscape of a problem - perhaps because of a heavy reliance on careful parameterisation. SHC also shows large variance - perhaps, in large part again, to a lack of parameters and complicated local optima avoidance techniques. GA and ES show large variation with respect to number of local optima, perhaps supporting the argument that evolutionary algorithms suffer more than most from the problem of becoming "stuck" in local optima.

All algorithms produce the smallest average error when no **local optima** (minima) are present in the fitness landscape. This is expected, as, with only one optimum, there are no alternative solutions to which the algorithms may converge. We observe the greatest average error with only one optimum from SHC,

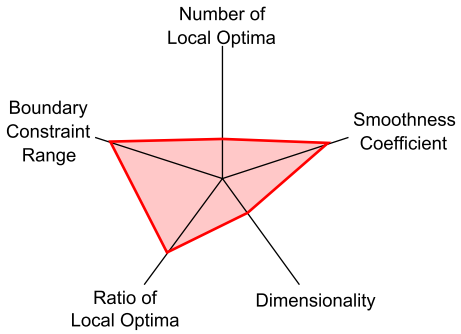
[1] http://www2.docm.mmu.ac.uk/STAFF/M.Amos/Project/Characterisation



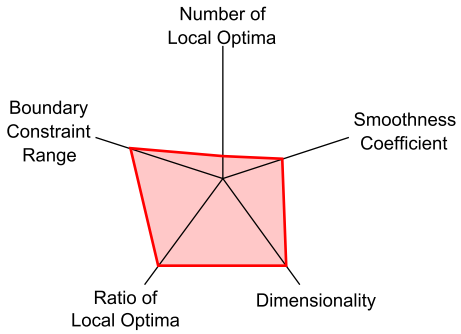
(a) Bees algorithm



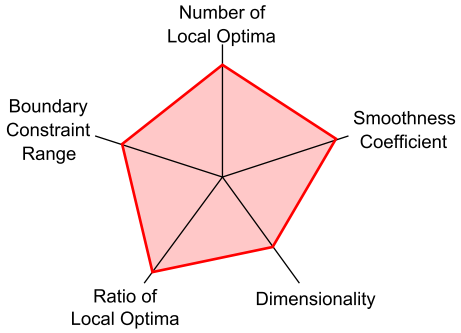
(b) Bacterial foraging optimisation algorithm



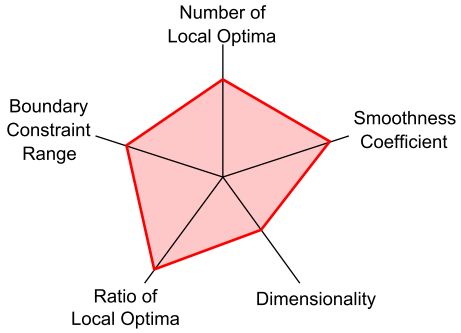
(c) Evolution strategies



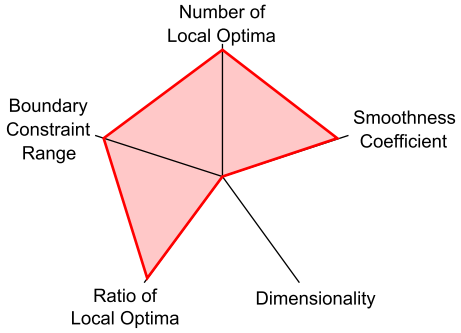
(d) Genetic algorithm



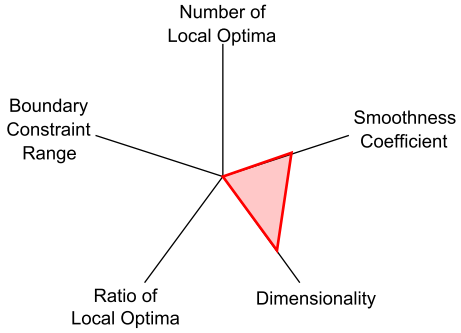
(e) Harmony search



(f) Particle swarm optimisation



(g) Random search



(h) Stochastic hill climbing

Fig. 1. Radar plots depicting the standard deviation of the average error of each algorithm with respect to differing landscape characteristics. Standard deviations are normalised on a per-axis basis. Values close to the centre of the plot indicate a larger variance in average error, indicating these algorithms are more affected by the characteristic. In general, the more robust an algorithm, the larger the plot surface area.

with BFOA (approx. 0.14) also showing a large average error. There are very small average errors (almost zero) from GA, ES, PSO, HS, RS and BA. BFOA also produces the largest variation in final solutions (0.32).With the introduction of only a single local optimum, performance of *most* algorithms degrades significantly. ES and GA suffer significantly, with average error increasing from approximately zero to 0.06 and 0.08 respectively, and the standard deviation of solutions increasing by around 0.15 for each algorithm. SHC also performs poorly, with a similar increase in average error. The least affected are RS (which blindly chooses random solutions, and is therefore unaffected by local minima) and BA, which contains a global search mechanism.

For algorithms which do not directly use the gradient of the landscape, we would expect to see no change in their performance as we adjust the **ratio of local optima** parameter. We observe that RS, which selects new solutions randomly from the entire search space, offers very similar performance in terms of mean error and success rate for all ratio values. Similarly, algorithms which perform a global search should be better at avoiding local minima even when they are attractive - and this is true for BA and HS. PSO shows little change in success rate as the ratio becomes more attractive, owing to the fact that solutions are directed towards the best particle, and their own best solution, regardless of their individual experience with the gradation of the landscape. Interestingly, SHC average error decreases as ratio increases - most likely due to an increased availability of 'better' solutions throughout the landscape. ES demonstrates very poor, yet consistent, performance as the ratio changes. Success rates are very low, and, interestingly, we observe a decrease in the standard deviation of solutions as the ratio increases. This suggests that ES is perhaps more "content" to optimise at a local minima, with the algorithm getting trapped in these more frequently as ratio increases. This could also be true of other algorithms whose deviation decrease, such as BFOA and SHC. GA performs in a similar manner to ES with regard to average error and diversity, although with a considerably better success rate, suggesting that this may be a general problem for algorithms which use an evolutionary approach.

At only one **dimension**, fitness landscapes are trivially easy. The performance of all algorithms reflects this, with all algorithms performing well on landscapes of a single dimension. All algorithms show a success rate (that is, optimisation with an error of under 1.0×10^{-4}) above 90%. As we increase the dimensionality to two, most algorithm performances begin to degrade. Suffering mostly severely is RS, which is to be expected, as random search is our most basic algorithm. Algorithms which also perform poorly at only two dimensions are ES, BA and PSO. It is perhaps surprising, at first, to see BA performing poorly, given that the algorithm contains a randomly sourced global search. However, this global search is *effectively* RS, which performs poorly, so we can assume the global search is not covering enough of the landscape. Coupled with the non-adaptive nature of the algorithm (meaning that solution selection around the current best area is within a relatively large range), poor algorithm performance is easily explained. We propose that PSO and ES suffer from a similar

problem, in that exploration is limited, and neither optimise their current best as accurately as their adaptive variants.

Random search exhibits a similar, yet less extreme, reaction to changes in **boundary constraints** as with the increase in dimensionality. This is to be expected, as the limit on objective function calculations results in random search having less chance to explore the search space. SHC also has an almost linear increase in average error, matching the linear increase in search space size, but produces consistently poor results in terms of success. The social system algorithms (BA and PSO) both exhibit slightly unusual behaviour - as the problem space increases, their success rate also increases. This suggests that their reliance on a parameter to search within a range is hindering the algorithms when the problem space is too small to properly explore. HS provides the best success rate for the entire range of sizes we have selected in this problem, indicating good exploration of the search space irrespective of the range parameter. BFOA also suffers significantly as search space size increases, again implying a heavy reliance on the parameter which controls the range of search for new solutions. The evolutionary algorithms do not cope particularly well with the increase of problem size, with performance in terms of both average error and success rate decreasing consistently as size increases.

The evolutionary algorithms (ES and particularly GA) perform poorly and are most affected by changing the **smoothness coefficient**. BA and PSO all also show decreasing success rate as the curves become steeper, as does BFOA which relies heavily on gradient information. Harmony search suffers similarly to the evolutionary algorithms, and swarm algorithms, as curves become more steep. The similarity in terms of success rate for all algorithms suggests that the availability of gradient information is something which affects all algorithms.

5 Conclusions

In this paper, we have described the results of an extensive study of nature-inspired algorithms, in terms of their performance on fitness landscapes with different characteristics. We studied six nature-based methods (plus two stochastic baseline algorithms), varying a number of landscape features. The most significant characteristic appears to be the number of local minima, where a combination of global and local search appears to be beneficial. On the other hand, the ratio of local optima to the global minimum appears to have little effect on the success of the algorithms under study. As expected, dimensionality proved problematic for all algorithms, whereas landscape smoothness appeared to have little effect.

This work offers a contribution to the empirical study of nature-inspired algorithms, and we hope that it motivates future investigations. To further this work, it may be useful to examine a larger collection of nature-inspired algorithms over a greater range of values for the characteristics, in order to more fully capture a wider variety of algorithmic performance. The current work provides a firm foundation for this.

References

[1] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[2] J. Woodward, "Why classifying search algorithms is essential," *2010 International Conference on Progress in Informatics and Computing*, 2010.

[3] R. Morgan and M. Gallagher, "When does dependency modelling help? Using a randomized landscape generator to compare algorithms in terms of problem structure," in *PPSN XI*, R. et al Schaefer, Ed. Springer-Verlag, 2010, pp. 94–103.

[4] M. Gallagher and B. Yuan, "A general-purpose tunable landscape generator," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 590–603, 2006.

[5] R. Jani, "A Generator for Multimodal Test Functions," in *Proc. SEAL 2008: LNCS 5361*, X. Li, Ed. SPringer-Verlag, 2008, vol. 3, pp. 239–248.

[6] Y. Jin, "Constructing dynamic optimization test problems using the multi-objective optimization concept," in *Applications of Evolutionary Computing: LNCS 3005*, G. Raidl, Ed. Springer-Verlag, 2004, vol. LNCS 3005, pp. 525–536.

[7] Z. Michalewicz, K. Deb, and M. Schmidt, "Test-case generator for nonlinear continuous parameter optimization techniques," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 197–215, 2000.

[8] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 184–192.

[9] R. Chiong, *Nature-inspired algorithms for optimisation.* Springer-Verlag, 2009.

[10] Q. Zhang, "On the convergence of a class of estimation of distribution algorithms," *Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 127–136, Apr. 2004.

[11] J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 495–511, 2002.

[12] R. Barr, B. Golden, and J. Kelly, "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristics*, vol. 1, pp. 9–32, 1995.

[13] J. Hooker, "Testing heuristics: we have it all wrong," *Journal of Heuristics*, 1995.

[14] P. Merz, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *Evolutionary Computation, IEEE*, vol. 4, no. 4, pp. 337–352, 2000.

[15] J. Tavares, F. B. Pereira, and E. Costa, "Multidimensional knapsack problem: a fitness landscape analysis." *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, vol. 38, no. 3, pp. 604–16, Jun. 2008.

[16] G. Uludag and A. Sima Uyar, "Fitness landscape analysis of differential evolution algorithms," in *Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009.*, 2009, pp. 1–4.

[17] C. McGeoch, "Toward an experimental method for algorithm simulation," *INFORMS Journal on Computing*, vol. 8, no. 1, pp. 1–15, 1996.

[18] A. Eiben, "A critical note on experimental research methodology in EC," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 1. Ieee, 2002, pp. 582–587.

[19] A. Brabazon and M. O'Neill, *Biologically inspired algorithms for financial modelling*. Springer-Verlag, 2006.
[20] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, Jun. 2002.
[21] D. Pham, A. Ghanbarzadeh, and E. Koc, "The Bees Algorithm A Novel Tool for Complex Optimisation Problems," in *Intelligent Production Machines and Systems*, D. Pham, E. Eldukhri, and A. Soroka, Eds., 2006, pp. 454–459.
[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings. ...*, 1995, pp. 1942–1948.
[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Ed. Addison-Wesley, 1989.
[24] T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, Mar. 1993.
[25] Z. Geem and J. Kim, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
[26] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu, 2011. [Online]. Available: http://www.cleveralgorithms.com
[27] J. Horn and D. Goldberg, "Genetic algorithm difficulty and the modality of fitness landscapes," in *Foundations of Genetic Algorithms 3*, 1994.
[28] K. M. Malan and A. P. Engelbrecht, "Quantifying ruggedness of continuous landscapes using entropy," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, May 2009, pp. 1440–1447.
[29] T. Hendtlass, "Particle swarm optimisation and high dimensional problem spaces," in *2009 IEEE Congress on Evolutionary Computation, CEC'09.* IEEE, May 2009, pp. 1988–1994.
[30] S. Kukkonen and J. Lampinen, "GDE3: The third Evolution Step of Generalized Differential Evolution," *2005 IEEE Congress on Evolutionary Computation*, pp. 443–450, 2005.
[31] H.-g. Beyer and H.-p. Schwefel, "Evolution strategies," *Natural Computing*, vol. 1, pp. 3–52, 2002.
[32] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, Jan. 2005.

Quantifying the Impact of Parameter Tuning on Nature-Inspired Algorithms

Matthew Crossley, Andy Nisbet and Martyn Amos

School of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, Manchester M1 5GD, UK.
Email: m.crossley@mmu.ac.uk

Abstract

The problem of *parameterization* is often central to the effective deployment of nature-inspired algorithms. However, finding the optimal set of parameter values for a combination of problem instance and solution method is highly challenging, and few concrete guidelines exist on how and when such *tuning* may be performed. Previous work tends to either focus on a specific algorithm or use benchmark problems, and both of these restrictions limit the applicability of any findings. Here, we examine a *number* of different algorithms, and study them in a "problem agnostic" fashion (i.e., one that is not tied to specific instances) by considering their performance on *fitness landscapes* with varying characteristics. Using this approach, we make a number of observations on which algorithms may (or may not) benefit from tuning, and in which specific circumstances.

Introduction and Background

There exist many algorithms that are inspired by nature, and each has associated with it a set of *parameters*. These define specific features or details of an algorithm that may be altered in order to change the behaviour or performance of the method (for example, in evolutionary algorithms, parameters may include mutation rate or crossover probability). The problem of finding the optimal settings for these parameters (often referred to as "tuning") is well-established (Lobo et al., 2007; Nannen et al., 2008; Akay and Karaboga, 2009; Birattari, 2009; Eiben and Smit, 2011), but little in-depth work has been performed on quantifying the *benefits* of tuning for a *range* of algorithms. We address this in the current paper, by investigating the precise benefits (or otherwise) of tuning for a number of different algorithms. Moreover, we do this in a way that is independent of any specific *problem*, by using an approach based on fitness landscape characteristics. The main contribution of the paper is therefore to establish a framework for deciding - prior to any problem-specific implementation - which algorithms may (or may *not*) benefit from tuning. Our aim is to offer advice to future practitioners on the relative merits of tuning, compared to the effort involved in finding the best set of parameter values. We achieve this by establishing, for each algorithm, the problem features that offer the most *potential* for performance improvements via tuning.

Previous work (Crossley et al., 2013) characterised a number of nature-inspired algorithms according to their performance on fitness landscapes with different features. However, the authors used the *default* parameter settings for each algorithm, which fails to reflect the fact that, in practice, methods are usually *tuned* prior to serious use (Leung et al., 2003; Adenso-Diaz and Laguna, 2006; Koster and Beney, 2007; Ridge and Kudenko, 2010). Here, we extend this work by quantifying the relative merits of tuning for a range of algorithms in a wide variety of fitness landscape scenarios. We achieve this by assessing both their tuned and untuned behaviour, using the methods described in Crossley et al. (2013).

In order to tune our selected algorithms, we use the notion of *racing*, which was first introduced in the field of machine learning (Maron and Moore, 1993, 1997). Specifically, we use the F-race algorithm (Birattari et al., 2002; Yuan and Gallagher, 2004; Smit and Eiben, 2009; Birattari et al., 2010), which has been extensively used to find the best possible set of parameter values for a given problem in a limited time.

The rest of the paper is organised as follows: we first describe our approach in the Methodology section, before presenting our experimental findings in the Results section. We conclude with a discussion of the implications of our results, and suggest further work.

Methodology

Our methodology may be summarised as follows: (1) select a number of nature-inspired algorithms, and obtain consistent source code for their implementation; (2) for each algorithm, find the best parameter settings (i.e., tune) over a number of different problems; (3) compare the performance of tuned and untuned algorithms.

Algorithm selection

We compare a number of nature-inspired algorithms, all of which are commonly applied to continuous function opti-

misation (we use the same set as in Crossley et al. (2013)). These may be classified (Brabazon and O'Neill, 2006) as either *social*, *evolutionary* or *physical*. The social algorithms we select are Bacterial Foraging Optimisation Algorithm (BFOA) (Passino, 2002), Bees Algorithm (BA) (Pham et al., 2006), and Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 1995). The evolutionary algorithms selected are Genetic Algorithms (GA) (Goldberg, 1989) and Evolution Strategies (ES) (Bäck and Schwefel, 1993), and physical algorithms are represented by Harmony Search (HS) (Geem and Kim, 2001). We also include stochastic hill climbing (SHC) as a "baseline" algorithm; in contrast to Crossley et al. (2013) we exclude random search, as it has no parameters to tune. As before, we heed the observation that "Ideally, competing algorithms would be coded by the same expert programmer and run on the same test problems on the same computer configuration" (Barr et al., 1995). With that in mind, we use only implementations provided by Brownlee (2011). Space prevents a complete description of specific implementation details for each algorithm, but full implementation details can be found in Brownlee (2011), which is freely available and contains complete source code.

Tuning

Our fundamental goal is to investigate the pre- and post-tuned performance of our selected algorithms on landscapes with different general features, and thus identify characteristics of landscapes for which tuning may yield significant differences in algorithm performance. As Morgan and Gallagher (2010) observe, "Different problem types have their own characteristics, however it is usually the case that complementary insights into algorithm behaviour result from conducting larger experimental studies using a variety of different problem *types*" (our emphasis). Rather than using arbitrary benchmark instances of problems in order to perform tuning, we use a landscape-based approach, as utilised in Crossley et al. (2013). As Morgan and Gallagher (2010) explain, this Max-Set of Gaussians (MSG) method (Gallagher and Yuan, 2006) is a "randomised landscape generator that specifies test problems as a weighted sum of Gaussian functions. By specifying the number of Gaussians and the mean and covariance parameters for each component, a variety of test landscape instances can be generated. The topological properties of the landscapes are intuitively related to (and vary smoothly with) the parameters of the generator." We now describe the characteristics under study:

Ruggedness of a landscape is often linked to its difficulty (Jones and Forrest, 1995), and factors affecting this include (1) the *number* of local optima (Horn and Goldberg, 1994), and (2) *ratio* of the fitness value of local optima to the global optimal value (Malan and Engelbrecht, 2009; Merz, 2000). Other significant factors concern (3) *dimensionality* (Hendtlass, 2009) (that is, the number of variables in the objective function), (4) *boundary constraints* (that is, the limits imposed on the value of a variable) (Kukkonen and Lampinen, 2005), and (5) *smoothness* of each Gaussian curve (effectively the gradient) used to generate the landscape (Beyer and Schwefel, 2002) - a smaller value indicates a smoother gradient. For each characteristic, we use the same ranges as in Crossley et al. (2013), summarised in Table 1.

Table 1: A summary of the ranges selected for the various characteristics in the landscape generation methodology.

Characteristic	Min	Max	Step	Default
No. of local optima	0	9	1	3
Ratio of local optima to global optimum	0.1	0.9	0.2	0.5
Dimensionality	1	10	1	2
Boundary constraints	10	100	10	30
Smoothness	10	100	10	15

To produce a test set of problems, we use the MSG landscape generator. For every value of every characteristic (in the range specified in Table 1) we generate a set of five landscapes, which makes up the initial problem set for each value. We then use the F-racing methodology (Birattari et al., 2002) to find optimised parameters for each algorithm, over *every* value of *every* landscape characteristic used. We ensure that termination criteria are standardised, in order to ensure reasonable comparisons, and therefore use the number of objective function evaluations to determine when to terminate an algorithm's run. We established, through initial experiments, that all selected algorithms generally converge within 20,000 objective function evaluations, so we use that as the specific value.

Comparison

We run each algorithm 100 times on each landscape in the set of landscapes generated for each particular characteristic value (when investigating smoothness, for example, we generate 1000 different landscapes (100 each for smoothness = $10 \ldots 100$), and run each algorithm 100 times on each landscape). This is done first for all algorithms with 'default' parameter configurations, and then again, this time using the parameter configurations obtained through the F-Racing process. We measure the performance of each algorithm in terms of the mean (μ) and standard deviation (σ) of the exact average error obtained, over all values for a particular characteristic. That is, we investigate the *robustness* of each algorithm to changes in the values for each characteristic, rather than their absolute performance on specific problem instances. This allows us to identify specific landscape features where tuning may make a significant difference, some difference, or no difference at all, for a particular algorithm.

Results

We find that the effect of tuning using F-Racing is varied across algorithms, and that they fit into three categories: Algorithms which *do not* benefit from F-Racing (ES), algorithms which only benefit significantly from F-Racing when a landscape is 'difficult' for the algorithm using default parameters (BA, HS, PSO), and algorithms which *always* benefit from F-Racing (BFOA, GA, SHC). Of course, we acknowledge the fact that F-Racing is just one of many possible meta-search techniques available for parameter tuning, and future work will involve a comparative study of alternative methods.

We summarise our results in Table 2; the full datasets are available online[1]; the repository contains all performance data across all runs, summary spreadsheets and details of all parameter settings. We now examine in detail the performance of each algorithm, using spider plots to graphically depict the results in Table 2. For each plot, the further a line is from the origin, the *smaller* the average error (that is, the "larger" an area, the larger the degree of robustness, which is considered "better").

Bacterial Foraging Optimisation Algorithm

There exists little discussion on the role of different parameters in the BFOA. While some elements of the search pattern are clearly altered by various parameters, it is very difficult to estimate values for these. In the original description of the BFOA (Passino, 2002), the parameter values were assigned based on observation of actual bacterial colonies. While this may be true to the nature-inspired concept, it is not necessarily the best way to obtain optimal performance from the algorithm. The combination of parameters offered by BFOA gives a highly configurable search environment. Parameters such as step size and population size directly affect the potential area the algorithm can explore in a given number of objective function calculations. Attraction and repulsion weights, and the "space" over which these attraction and repulsion effects spread, work to control local optima avoidance. Parameters controlling the number of chemotactic steps before a reproduction step, and the number of reproduction steps before an elimination-dispersal event, control the balance of *exploitation* versus *exploration*. Given that the search behaviour of the algorithm is highly configurable, it is unsurprising that BFOA is heavily reliant on tuning. Results for BFOA are shown in Figure 1. Across all characteristics, tuning offers a *significant* improvement on the average error and standard deviation of the performance - in many cases, improving the ranking of the algorithm from the largest average error to one of the smallest, and coping well with the changing characteristics. We see the most significant improvement where boundary constraint ranges change, a characteristic that is heavily reliant on parameters

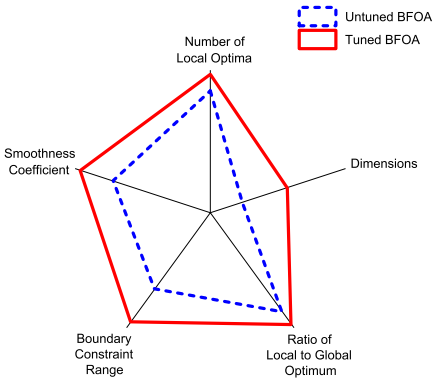


Figure 1: Summary of results for **Bacterial Foraging Optimisation Algorithm**.

which control the range at which new solutions are generated (in the case of BFOA, this is the *step size*). Improvements are also shown for dimensionality and smoothness coefficient, increasing the performance of BFOA where there is little gradient information in a large fitness landscape. Smaller improvements are demonstrated by the increasing number of local optima and the increasing attractiveness of these local optima, but tuning still benefits the algorithm considerably.

In terms of the configurations selected by F-Racing, there is little variation in parameter values as characteristics change. Across all characteristics, and all values for those characteristics, there are only eight different configurations selected by racing. This suggests that, while it is difficult to find a *good* configuration, once it has been found, it is likely to be good for all *similar* problems. Tuning is vital to the performance of the BFOA, but it is possible that by exploring problems using a similar methodology to that demonstrated here, we may create a 'bank' of promising configurations.

Bees Algorithm

The BA is considered to be an algorithm on which parameterisation has little effect (Pham et al., 2006). We observe that the BA is one of the best *untuned* performers in this study, offering weight to this argument for relative parameter insensitivity. In terms of adjusting the BA to cope with an increasing number of local optima, there are several parameters which have an effect. Parameters such as the *number of sites* under investigation, the *number of bees* attributed to those sites, and the *differentiation* between sites and 'elite' sites are all factors which affect the searching behaviour of the algorithm to allow for greater flexibility as the modality of the problem landscape increases. Results for BA are

[1] http://dx.doi.org/10.6084/m9.figshare.696908

Table 2: Mean (μ) and standard deviation (σ) of the exact average error of algorithm performance (both untuned (UT) and tuned (T)). Smaller values imply more robustness to changes in a specific characteristic.

		BFOA		Bees Algorithm		ES		GA		Harmony Search		PSO		SHC	
		UT	T	UT	T	UT	T	UT	T	UT	T	UT	T	UT	T
# of Local Optima	μ	0.118	0.003	0.001	8.8×10^{-6}	0.085	0.078	0.093	0.015	0.011	2.2×10^{-5}	0.025	0.014	0.266	0.072
	σ	0.011	0.001	2.1×10^{-4}	9.2×10^{-7}	0.028	0.026	0.033	0.008	0.005	2.9×10^{-5}	0.010	0.010	0.041	0.020
Dimensions	μ	0.754	0.417	0.216	0.073	0.542	0.544	0.420	0.529	0.364	0.263	0.420	0.157	0.577	0.589
	σ	0.388	0.360	0.202	0.069	0.345	0.346	0.233	0.401	0.271	0.204	0.307	0.145	0.261	0.371
Local Optima Ratio	μ	0.120	0.003	0.001	8.7×10^{-5}	0.084	0.082	0.079	0.007	0.007	0.002	0.025	0.016	0.284	0.088
	σ	0.021	0.002	2.3×10^{-4}	1.9×10^{-4}	0.012	0.012	0.006	0.006	0.003	0.004	0.004	0.011	0.045	0.011
Boundary Range	μ	0.317	0.022	0.001	0.001	0.097	0.093	0.125	0.021	0.048	0.001	0.076	0.022	0.446	0.305
	σ	0.213	0.033	1.3×10^{-4}	0.001	0.017	0.018	0.057	0.016	0.041	0.001	0.050	0.013	0.239	0.217
Smoothness	μ	0.260	0.010	0.004	0.001	0.110	0.102	0.154	0.021	0.018	0.001	0.043	0.014	0.349	0.112
	σ	0.089	0.005	0.002	4.3×10^{-4}	0.012	0.012	0.045	0.011	0.007	0.001	0.012	0.006	0.039	0.014

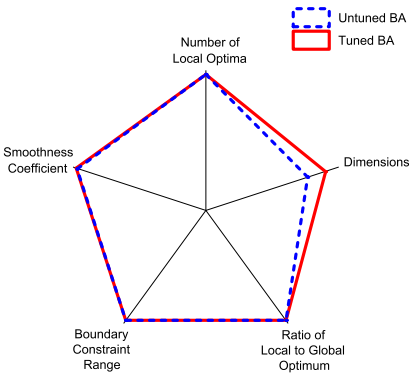


Figure 2: Summary of results for **Bees Algorithm**.

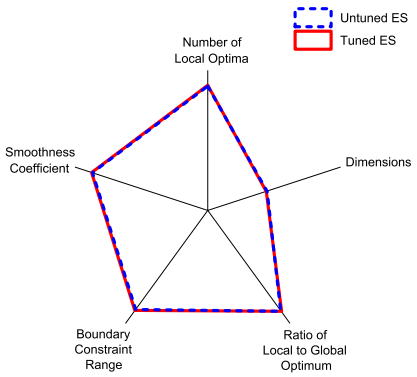


Figure 3: Summary of results for **Evolution Strategies**.

shown in Figure 2. Post-tuning, we find that the BA selects the *same* parameter configuration, regardless of the number of local optima present in the landscape. We then see that tuning has no effect on the ability of the algorithm to cope with increasing numbers of local optima. As long as the number of sites under investigation is greater than the number of optima, the algorithm is capable of dealing with modality. Coupled with the abandonment of 'unpromising' sites, this means that 'too many' sites are not detrimental to the exploration pattern of the algorithm.

We see the same pattern when increasing the ratio of local optima to the global optimum. As long as the number of sites under investigation covers the modality of the landscape, the BA is not hampered by increasing levels of attractiveness, regardless of parameter settings. The *patch size* parameter of the BA controls the distance from a site bees are allowed to explore. This is the parameter which affects the search behaviour of the algorithm as boundary constraint size increases. The BA allows for full coverage of any sized search space, using *scout bees* to investigate new random sites to give 'teleportation' across the landscape. As with the number of local optima, we find the F-Races for the BA select the same parameter set for most boundary constraint sizes. We find that, post-tuning, the performance of the BA actually *decreases* slightly, suggesting the algorithm can cope less well with changes in boundary constraint size. We believe that the configurations may have become over-fitted to the landscapes used for tuning, and, while performance on the landscapes used for racing may have increased, the ability to search generalised landscapes has decreased. Dimensionality provides the most significant result in terms of pre-tuning and post-tuning performance of the BA. We observe little change in performance at one to three dimensions - the point where the untuned algorithm is already performing well. As dimensionality increases beyond this, the effect of tuning becomes *increasingly* beneficial. We suggest that there is no increase in performance in other characteristics because these landscapes are *not challenging enough* to the BA to require adjusting the parameters. For the ranges of landscape characteristics on which we have tested the BA, it is clear that tuning generally makes little difference to the performance, as suggested by its original developer.

Evolution Strategies

ES has the smallest number of parameters of all the algorithms studied here (excepting the baseline algorithm, stochastic hill climbing). The two parameters this form of ES offers are (1) *population size* and (2) *number of children*. It is suggested (Cant-Paz, 2001) that altering these parameters adjusts *selection pressure* (that is to say, the *greediness* of the algorithm changes). The parameter configurations obtained through F-Racing are varied, implying that there do exist some configurations that are more successful than others. A range of configurations are selected across each characteristic - both in terms of different values for the two parameters, and different selection pressures when the two parameters are combined. Results for ES are shown in Figure 3. It is perhaps surprising to observe that the results of using the tuned parameters show little or no change in performance across all characteristics. There is a small decrease in average error as the number of local optima changes, but the standard deviation is similar for both untuned and tuned, suggesting that while the average error has decreased very slightly, the ability of the algorithm to cope with increasing numbers of local optima is unchanged. For all other characteristics post-tuning, there is little change in both average error and standard deviation across characteristics values (that is to say, the algorithm is no more capable of dealing with changes in these characteristics). This is perhaps consistent with the definition of the two parameters the algorithm offers - selection pressure can only affect the way in which ES explores local optima, and there is no control over the area that is explored around each point of interest, or any way to encourage the algorithm to rapidly explore an increasingly large search space.

We use a simple variant of ES, here, and there exist many other versions of the ES algorithm that offer a greater range of parameters (such as CMA-ES (Hansen and Kern, 2004)). ES clearly yields its best performance with an "out-of-the-box" parameter configuration, which means that it is quick to implement. However, our results suggest that there is little that can be done to improve the performance of this particular variant.

Genetic Algorithm

The performance of the GA increases post-tuning, coping significantly better with increasing numbers of local optima, increasing boundary constraint range and an increasing smoothness coefficient. Results for the GA are shown in Figure 4. The parameters of the GA are not as intuitively linked to the exploration pattern as many of the other algorithms in the study. This particular GA offers four configurable parameters: (1) *population size*, (2) *'bits' per parameter* in the representation, (3) *crossover rate* and (4) *mutation rate*. In experiments with a fixed number of objective function calculations, population size affects the number of generations the algorithm evaluates before terminating. A

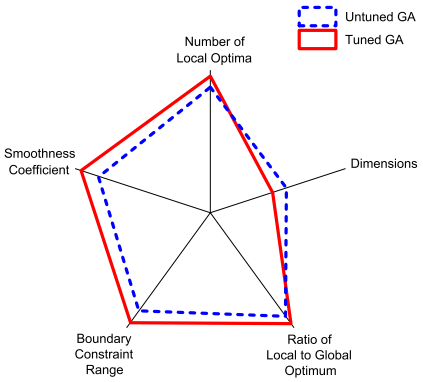


Figure 4: Summary of results for **Genetic Algorithm**.

larger number of bits in a bit string representation allows more 'precise' solutions to be generated at the expense of having a representation which is less affected by mutation. Similarly to BFOA, there are a few configurations which reoccur across different characteristics and different characteristic values. It is probable that once a 'good' configuration has been found for a GA, it is applicable to 'similar' landscapes, which is consistent with the suggestion Goldberg (1989) that GAs are robust problem solvers, exhibiting approximately the same performance across a wide range of problems.

With increasing dimensionality, the GA initially shows promising results in terms of tuned performance, with a marked performance increase up to four dimensions. The benefit from tuning rapidly declines, however, until the tuned performance is *worse* than that of the tuned version. There are two possible explanations for this: the first is that the restriction on the number of objective calculations did not allow the F-Race algorithm to gather any meaningful performance data from the configurations. The second explanation is that we did not test a wide enough range of configurations - although two of the four parameters have definite ranges (mutation and crossover rates are percentages, thus generation was bounded between zero and one), so this is unlikely.

Harmony Search

The four parameters of HS all control different aspects of the search strategy. *Memory size* dictates how many promising solutions can be stored - effectively, how many potential sites of interest are retained by the algorithm. *Consideration rate* and *adjustment rate* control how new solutions are generated. The consideration rate is the percentage chance that a solution based on one in memory will be generated (conversely, *1-consideration rate* is the chance a random so-

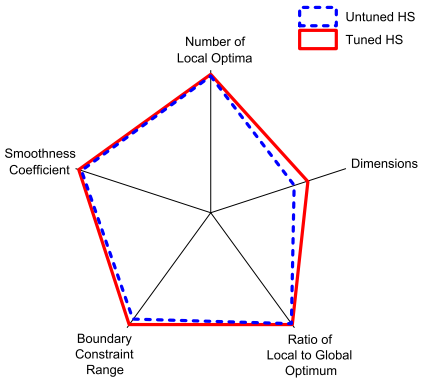


Figure 5: Summary of results for **Harmony Search**.

lution is generated instead). The adjustment rate is then the percentage chance that the randomly chosen solution from memory will be adjusted. If so, the fourth parameter, which controls the *maximum range* at which solutions can be adjusted, is used. If the adjustment does not occur, the considered solution potentially occupies an additional slot in the memory - thus increasing the chance that this solution may be chosen for consideration again. The interplay between these parameters is crucial, and it is somewhat hard to see how consideration rate and adjustment rate can directly affect the search strategy - unlike memory size and range, which are more obvious. The results for HS are shown in Figure 5. HS, like the BA, offers some of the lowest 'out of the box' average error rates in this study. For most characteristics, there is little room for a performance increase post-tuning. Boundary constraint range proves to be the second-most challenging characteristic to HS pre-tuning, but post-tuning shows improved performance. The range values in all the configurations selected by F-Racing are much smaller than those in the 'out of the box' values, and this contributes significantly to the performance improvement when boundary constraint ranges are increasing. The consideration rate also decreases almost linearly as size increases - effectively, more random solutions are used instead of relying on the 'memory'. These random solutions allow the solution pool to jump from one position in the search space to another, encouraging a wider search space, and explaining the significant improvement as boundary constraint range increases. Dimensionality also yields an improvement in the tuned parameter performance of HS, in terms of both average error and ability to cope, as it rises. High dimension problems (seven and above) have a much higher consideration rate than the successful configurations for lower dimensionality, suggesting that a focus on *exploitation* rather than *exploration* is beneficial to the HS when dimensionality is high. This is the opposite case of what happens with boundary constraint range, as discussed above.

Particle Swarm Optimisation

PSO in this form has four parameters; these control the *population size*, the *maximum velocity* of a particle, the bias towards the *particle best solution* and the bias towards the *global best solution*. With these parameters, it is possible to control the coverage of a search space (the number of particles), enforce a large search area of a small search area for each particle (the maximum velocity), and, through manipulation of the local and global best solution bias, control the capability of the algorithm to converge on a single solution or explore several areas of interest (optima avoidance). Results for PSO are shown in Figure 6. The parameters used cover a broad range of search behaviours, and, as such, we would expect to see a large improvement in particle swarm performance post-tuning. This holds true for most of our characteristics. Results for the number of local optima, for example, show a reasonable decrease in average error as the number of local optima increases, yet the standard deviation demonstrates no change, indicating that the algorithm is no more capable of dealing with increasing numbers of local optima post-tuning. Performance of PSO greatly improves on dimensionality post-tuning, in terms of both average error and ability to cope as it grows. The F-Race algorithm for PSO selects the same configuration for all values of dimensionality (except for 2 dimensions), implying that there is no specific parameter that requires adjustment to cope with the increase in dimensionality, but selecting a configuration which provides good *exploration* allows PSO to perform well as the size of the search space increases exponentially. This trend continues across all characteristics, with F-Races often selecting the same configurations, regardless of characteristic values. As with the other swarming algorithms, we suggest that once a good configuration has been found, it is able to deal with a wide range of problems of a similar nature, regardless of the specific characteristics. The configurations selected are all varied in their parameters, and it is unexpected to see that there is no pattern to maximum velocity as boundary constraint range increases. This is possibly because maximum velocity is an upper bound, and there are particles with randomly generated velocities below the maximum, so this parameter is less significant than it may initially appear. It would perhaps be interesting to consider the effect of having a *minimum* velocity on the increase in boundary constraint range, although this would also severely hamper exploitation.

Stochastic Hill-Climbing

With only a single parameter - the *range* at which new solutions are generated - the SHC algorithm does not offer a large amount of customisation. This single parameter is directly linked to the search pattern and nothing else, and as

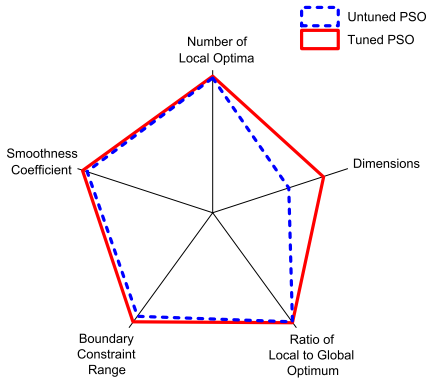


Figure 6: Summary of results for **Particle Swarm Optimisation**.

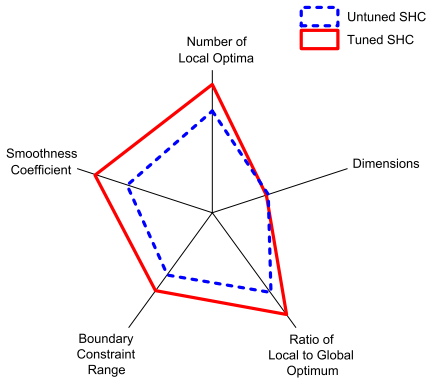


Figure 7: Summary of results for **Stochastic Hill-Climbing Algorithm**.

there are no other parameters there is no interplay between parameters to consider. Arguably, therefore, SHC should prove the easiest algorithm to tune. Results for SHC are shown in Figure 7. All characteristics, barring dimensionality, show an improvement post-tuning. As the neighbourhood size is the range at which new solutions are generated, it is unsurprising that tuning improves algorithm performance as boundary constraint ranges change. As the number of objective function calculations is limited, despite having a larger neighbourhood size, the ability of the algorithm to effectively explore larger environments is still restricted, therefore the average error does not decrease by as much as may be expected, and the ability of the algorithm to deal with increasing search space sizes improves only slightly. SHC demonstrates a large increase in performance and a greater ability to cope with more optima (a reduced standard deviation) post-tuning. The parameter configurations selected for the number of local optima, the ratio of local optima *and* the smoothness all have a neighbourhood size of around 50% of the search space size. We suggest that the performance improvement for all of these characteristics is actually derived from the algorithm having configured itself properly for the search space size used as a default for all other characteristics, rather than tuning itself to best perform on any specific characteristic.

Conclusions and Future Work

In this paper we have built on previous studies of the performance of nature-inspired algorithms on fitness landscapes with different characteristics. Earlier work explored 'out of the box' parameter configurations, and we futher develop this by using an automated parameter configuration methodology. This allows us to study the effect of tuning on different algorithms, contributing significantly to the debate on when and how it is beneficial to tune specific algorithms.

We observe that algorithms broadly fall into three categories: algorithms which *do not/sometimes/always* benefit from tuning by F-Racing. Dimensionality often offers the most significant improvement post-tuning in algorithms, particularly those with parameters that increase the *breadth* of search space (swarming algorithms are significantly better here than evolutionary algorithms). The methodology presented here is easy to implement, is computationally inexpensive, and offers considerably more information on the performance of an algorithm than using a standard set of benchmark problems. We hope that it will offer a framework for the experimental comparison of nature-inspired algorithms, as well as a useful set of heuristics for practitioners to use in order to decide when and how to tune their methods. Future work will focus on a comparative study of tuning techniques (i.e., in addition to F-Racing), and the application of our insights to the predictive performance ranking of methods on given problems.

References

Adenso-Diaz, B. and Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114.

Akay, B. and Karaboga, D. (2009). Parameter tuning for the artificial bee colony algorithm. In Nguyen, N., Kowalczyk, R., and Chen, S.-M., editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, volume 5796 of *Lecture Notes in Computer Science*, pages 608–619. Springer Berlin Heidelberg.

Bäck, T. and Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23.

Barr, R., Golden, B., and Kelly, J. (1995). Designing and reporting

on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32.

Beyer, H.-g. and Schwefel, H.-p. (2002). Evolution strategies. *Natural Computing*, 1:3–52.

Birattari, M. (2009). *Tuning metaheuristics: a machine learning perspective*. Springer.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 11–18, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-race and iterated f-race: An overview. *Experimental methods for the analysis of optimization algorithms*, pages 311–336.

Brabazon, A. and O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Springer-Verlag.

Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.

Cant-Paz, E. (2001). Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7(4):311–334.

Crossley, M., Nisbet, A., and Amos, M. (2013). Fitness landscape-based characterisation of nature-inspired algorithms. In Tomassini, M., Antonioni, A., Daolio, F., and Buesser, P., editors, *Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA'13), Lausanne, Switzerland, April 4-6, 2013. Lecture Notes in Computer Science, Vol. 7824*, pages 110–119. Springer.

Eiben, A. E. and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31.

Gallagher, M. and Yuan, B. (2006). A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603.

Geem, Z. and Kim, J. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Hansen, N. and Kern, S. (2004). Evaluating the cma evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer.

Hendtlass, T. (2009). Particle swarm optimisation and high dimensional problem spaces. In *2009 IEEE Congress on Evolutionary Computation, CEC'09.*, pages 1988–1994. IEEE.

Horn, J. and Goldberg, D. (1994). Genetic algorithm difficulty and the modality of fitness landscapes. In *Foundations of Genetic Algorithms 3*.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings. ...*, pages 1942–1948.

Koster, C. and Beney, J. (2007). On the importance of parameter tuning in text categorization. *Perspectives of Systems Informatics*, pages 270–283.

Kukkonen, S. and Lampinen, J. (2005). GDE3: The third Evolution Step of Generalized Differential Evolution. *2005 IEEE Congress on Evolutionary Computation*, pages 443–450.

Leung, F. H., Lam, H., Ling, S., and Tam, P. K. (2003). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *Neural Networks, IEEE Transactions on*, 14(1):79–88.

Lobo, F. G., Lima, C. F., and Michalewicz, Z. (2007). *Parameter setting in evolutionary algorithms*. Springer Verlag.

Malan, K. M. and Engelbrecht, A. P. (2009). Quantifying ruggedness of continuous landscapes using entropy. In *2009 IEEE Congress on Evolutionary Computation*, pages 1440–1447. IEEE.

Maron, O. and Moore, A. (1993). Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263.

Maron, O. and Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11:193–225.

Merz, P. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *Evolutionary Computation, IEEE*, 4(4):337–352.

Morgan, R. and Gallagher, M. (2010). When does dependency modelling help? Using a randomized landscape generator to compare algorithms in terms of problem structure. In et al Schaefer, R., editor, *PPSN XI*, pages 94–103. Springer-Verlag.

Nannen, V., Smit, S. K., and Eiben, A. E. (2008). Costs and benefits of tuning parameters of evolutionary algorithms. In *Parallel Problem Solving from Nature–PPSN X*, pages 528–538. Springer.

Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67.

Pham, D., Ghanbarzadeh, A., and Koc, E. (2006). The Bees Algorithm A Novel Tool for Complex Optimisation Problems. In Pham, D., Eldukhri, E., and Soroka, A., editors, *Intelligent Production Machines and Systems*, pages 454–459.

Ridge, E. and Kudenko, D. (2010). Tuning an algorithm using design of experiments. In *Experimental methods for the analysis of optimization algorithms*, pages 265–286. Springer.

Smit, S. K. and Eiben, A. E. (2009). Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 399–406. IEEE.

Yuan, B. and Gallagher, M. (2004). Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 172–181. Springer.