Nature-inspired Optimisation: Improvements to the Particle Swarm Optimisation Algorithm and the Bees Algorithm

A thesis submitted to Cardiff University,

for the degree of

Doctor of Philosophy

By

Michael O. Sholedolu, ***B.Sc., M.Sc.***

Cardiff School of Engineering

Manufacturing Engineering Centre

Cardiff University

United Kingdom

2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed..... (Michael Sholedolu-Candidate)

Date....25/10/10.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed..... (Michael Sholedolu-Candidate)

Date.....25/10/10.....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed..... (Michael Sholedolu-Candidate)

Date....25/10/10.....

Dedicated to my loving parents

Abstract

This research focuses on nature-inspired optimisation algorithms, in particular, the Particle Swarm Optimisation (PSO) Algorithm and the Bees Algorithm. The PSO Algorithm is a population-based stochastic optimisation technique first invented in 1995. It was inspired by the social behaviour of birds flocking or a school of fish. The Bees Algorithm is a population-based search algorithm initially proposed in 2005. It mimics the food foraging behaviour of swarms of honey bees.

The thesis presents three algorithms. The first algorithm called the PSO-Bees Algorithm is a cross between the PSO Algorithm and the Bees Algorithm. The PSO-Bees Algorithm enhanced the PSO Algorithm with techniques derived from the Bees Algorithm. The second algorithm called the *improved* Bees Algorithm is a version of the Bees Algorithm that incorporates techniques derived from the PSO Algorithm. The third algorithm called the SNTO-Bees Algorithm enhanced the Bees Algorithm using techniques derived from the Sequential Number-Theoretic Optimisation (SNTO) Algorithm.

To demonstrate the capability of the proposed algorithms, they were applied to different optimisation problems. The PSO-Bees Algorithm is used to train neural networks for two problems, Control Chart Pattern Recognition and Wood Defect Classification. The results obtained and those from tests on well known benchmark functions provide an indication of the performance of the algorithm relative to that of other swarm-based stochastic optimisation algorithms.

The *improved* Bees Algorithm was applied to mechanical design optimisation problems (design of welded beams and coil springs) and the mathematical benchmark problems used previously to test the PSO-Bees Algorithm. The algorithm incorporates cooperation and communication between different neighbourhoods. The results obtained show that the proposed cooperation and communication strategies adopted enhanced the performance and convergence of the algorithm.

The SNTO-Bees Algorithm was applied to a set of mechanical design optimisation problems (design of welded beams, coil springs and pressure vessel) and mathematical benchmark functions used previously to test the PSO-Bees Algorithm and the *improved* Bees Algorithm. In addition, the algorithm was tested with a number of deceptive multi-modal benchmark functions. The results obtained help to validate the SNTO-Bees Algorithm as an effective global optimiser capable of handling problems that are deceptive in nature with high dimensions.

Acknowledgement

As a devoted Christian, I would like to thank Jehovah Lord God of Host for His Mercy, Grace, Protection, Guidance and Favour in abundance (despite all difficulties) from the very moment I came out of my mother's womb unto this present day and for listening & answering my prayers.

The Biography of my life will ***not*** be complete without Professor Duc Truong Pham. I am GREATLY indebted to my supervisor, Professor Duc Truong Pham for many things. I thank him for accepting me to be one of his students at the Manufacturing Engineering Centre (MEC) in Cardiff University; for his suggestion of this research area to work on (from Robotics to Optimisation), and for his numerous contributions and direct supervision. I would also like to thank him for his continual and invaluable suggestions, insight, motivation and support throughout this research. I owe special thanks to Prof. D. T. Pham for carefully reading and correcting the draft manuscripts of this thesis. ***Most important of all***, for being a father to me is PRICELESS and I really do not know what words to use to express my profound GRATITUDE and APPRECIATION. Special recognition to Mrs. Paulette Pham and Kim Pham for their kindheartedness.

I would like to thank Celia Rees (Mam) for her motherly love and support throughout my research work especially the many reminder emails. I also want to thank Dr. R.I. Grosvenor and Mr. Paul Prickett for giving me the references used in pursuit of this PhD research. Special thanks to Dr. Michael S. Packianather & Dr. Eldaw Eldukhri for their help and contributions at the CIRP ICME '06 conference in Ischia, Italy; the members

and staff of the MEC, the Cardiff Bay Bees research group, the MEC Robotics research group, my colleagues whom has contributed in one way or another. Special thanks again to the MEC for the high quality scientific ambiance and equipment they provided throughout my research work especially when I was building my team of flying robots for which I almost ended up flying myself as a replacement for the robots (laugh).

Furthermore, I would also like to thank Dr. Jan Beutler for his *assistance*; Ms. Pauline Richards for her love, sincerity and understanding; Mrs. Lynn Murrell & David Harrison of INSRV for their understanding when I urgently had to leave temporarily with short notice; Mrs. Rhian Williams for her assistance and the IT Administrators in the MEC for maintaining the computer system used throughout my research work.

I want to thank and acknowledge the contributions of the following people: Professor Andries P. Engelbrecht, the author of "Fundamentals of Computational Swarm Intelligence" and John Wiley & Sons, Ltd; Thomas Weise, the author of "Global Optimisation Algorithms – Theory and Applications"; Karin Zielinski and Rainer Laur, the authors of the Paper titled "Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimisation Algorithm" and finally; Yamille del Valle, Ganesh Kumar Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez and Ronald G. Harley, the authors of the Paper titled "Particle Swarm Optimisation: Basic Concepts, Variants and Applications in Power Systems" and the Institute of Electrical and Electronic Engineers, Inc (IEEE).

Last but not least, I would like to express deep gratitude and admiration to my parents Overseer & Deaconess H. A. Sholedolu for their prayers, love, support and care - I hold

Kindness is a language which the deaf can hear and the blind can read.
--Mark Twain

Without the kindness I received, this research work would not have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Courage is your greatest present need.
It's all in the mind, you know.

In a competitive world, only the best (fittest, safest, cheapest, fastest, etc) is good enough. This is why optimisation (local & global) is very frequent in applications. Optimisation is concerned with finding the *best* solution to a problem, where *best* refers to an acceptable (or satisfactory) solution, which can be the absolute best over a set of candidate solutions, or any other candidate solutions.

Optimisation techniques are employed in diverse fields such as engineering, manufacturing, finance, medicine, computing art and music, chemistry, physics and economics. The task of optimisation is that of determining the values of a set of parameters so that some measure of optimality is satisfied subject to certain constraints.

This research focuses on the Particle Swarm Optimisation Algorithm, an algorithm belonging to the population-based stochastic optimisation technique inspired by the social behaviour of birds flocking or a school of fish and the Bees Algorithm, a population-based search algorithm based on the food foraging behaviour of swarms of honey bees.

1.1 Motivation

The Particle Swarm Optimisation Algorithm is based on the *swarm intelligence* concept, which is the property of a system, whereby the collective behaviour of unsophisticated agents that are interacting locally with their environment to create coherent global functional patterns. In contrast to other global optimisers, the Particle Swarm

Optimisation Algorithm focuses on social interaction and the existence of cooperation amongst individuals purposely to exchange knowledge about the search space that makes it a robust, flexible and effective optimisation algorithm.

However, the Particle Swarm Optimisation Algorithm is known to suffer from the problem of premature convergence. This is well documented in the literature. The process of trying to find a solution to this problem lead to the development of the PSO-Bees Algorithm. The algorithm combines the fast convergence property of the Particle Swarm Optimisation Algorithm and the inherent ability of the original Bees Algorithm to avoid being trapped in local optima.

The Bees Algorithm is a nature-inspired population-based search algorithm that mimics the food foraging behaviour of swarms of honey bees. The algorithm performs a kind of neighbourhood search combined with global random search and can be used for both continuous and discrete optimisation problems.

Observations of the aerial view of the operation of the Bees Algorithm show a swarm of bees flying across the search space. However, on zooming in into the algorithm, it can be seen that there are independent patches of bees searching the problem space with no communication or cooperation amongst these patches to help and make the search process better as in the case of the Particle Swarm Optimisation Algorithm. The *improved* Bees Algorithm integrates cooperation and communication between different neighbourhoods in the original Bees Algorithm to find the global optimum. The proposed strategies enhanced the performance and convergence of the algorithm. These ensure the algorithm search only the promising areas of the search space and avoid the need for

'killing' bees as previously employed in other variants of the Bees Algorithm. This approach also reduces the number of function evaluations of the algorithm in finding the global optimum of functions.

The Sequential Number-Theoretic Optimisation (SNTO) Algorithm is a global optimisation technique where many points are generated in a *multi-dimensional domain*, the optimum point is selected and the domain is contracted around the neighbourhood of the optimum. This technique of generating points in all dimensions is incorporated into the Bees Algorithm to enhance its exploration capabilities from initialisation and to improve its ability to handle high dimensional problems.

The SNTO technique is attractive because of its impressive features, such as simplicity, ease of implementation, effective optimisation performance, ability to handle general optimisation problems and the fact that no calculation of the derivatives of the objective functions is required. Furthermore, the implementation of the SNTO technique in the Bees Algorithm resulted in

- a robust method (evenly distributed in all dimensions from initialisation);
- faster convergence to the global optimum of the objective functions;
- smaller number of function evaluations;
- eliminating the need for 'killing' bees as employed in some variants of the Bees Algorithm;
- avoidance of being trapped in local optima;
- a wide exploration across all dimensions and later an exploitative local search to improve the solution.

The SNTO-Bees Algorithm resolves the limitations of the Bees Algorithm when dealing with high dimension problems.

1.2 Aim and Objectives

The general aim of this research is to prove the hypothesis that improved nature-inspired optimisation algorithms will result from hybridisation. In particular, the ability of the Bees Algorithm to avoid being trapped in local optima will be exploited to solve the problem of premature convergence in the PSO Algorithm. Cooperation and communication between different neighbourhoods, which are features of the PSO Algorithm, will be introduced to enhance the performance and convergence of the Bees Algorithm. Finally, the SNTO technique of generating points in a multi-dimensional capacity will be incorporated to Bees Algorithm.

The main objectives of this research are as follows:

1. To perform a detailed analysis of existing global optimisation algorithms, especially swarm-based optimisation algorithms with a view to improving the PSO Algorithm and the Bees Algorithm.

2. To solve the problem of premature convergence in the PSO Algorithm.

3. To improve the ability of the PSO Algorithm to converge onto the global optima.

4. To develop a robust, flexible and effective PSO Algorithm able to train neural networks to recognise difficult patterns in control chart data and to be excellent in the classification of wood defects in a more effectual manner.

5. To develop and test the proposed algorithms on the well-known mathematical benchmark functions and obtain empirical results for comparison with other global optimisers including the deterministic simplex method (SIMPSA), the stochastic simulated annealing (NESIMPSA), the Genetic Algorithm (GA), the Ant Algorithm (ANT), the original Bees Algorithm and the original PSO Algorithm.

6. To develop and test the performance of the SNTO-Bees Algorithm on a number of deceptive multi-modal functions.

7. To improve the ability of the original Bees Algorithm to converge onto the global optima of functions with high dimensions.

8. To develop and test the second and third proposed algorithms on certain mechanical design optimisation problems, namely the designs of welded beams (single-objective and multi-objective), coil springs. To obtain empirical results for comparison with other well-known global optimisers.

9. To develop and test the performance of SNTO-Bees Algorithm on mechanical design optimisation problem, the design of pressure vessel. To obtain empirical results for comparison with other well-known global optimisers including the APPROX method, the DAVID technique, the Geometric Programming (GP), the Genetic Algorithm (GA), the improved Genetic Algorithm, the SIMPLEX method and the RANDOM technique.

1.3 Methodology

- Review of previous work: an extensive survey was performed on the state of the art in intelligent optimisation techniques, focusing on nature-inspired algorithms, to identify research trends and potential solutions.

- Algorithm development and evaluation: The standard PSO Algorithm was extended by adding adaptive neighbourhood search and global random search. The PSO-Bees Algorithm combines the fast convergence property of the PSO Algorithm and the inherent ability of the Bees Algorithm to avoid been trapped in local optima. The performance of the new algorithm was evaluated by computer simulation to solve a number of benchmark problems. The results obtained were compared with those of other optimisation techniques including the deterministic simplex method (SIMPSA), the stochastic simulated annealing (NESIMPSA), the Genetic Algorithm (GA), the Ant Algorithm (ANT), the original Bees Algorithm and the original PSO Algorithm to assess the effectiveness of the proposed methods.

  The standard Bees Algorithm was extended by adding cooperation and communication between different neighbourhoods. The performance of the new version of the algorithm called the *improved* Bees Algorithm was evaluated by computer simulation to solve a number of benchmark problems. The results obtained were compared with those of other optimisation techniques including the deterministic simplex method (SIMPSA), the stochastic simulated annealing (NESIMPSA), the Genetic Algorithm (GA), the Ant Algorithm (ANT), the

original Bees Algorithm and the original PSO Algorithm to assess the effectiveness of the proposed methods.

The standard Bees Algorithm was extended by adding multi-dimensional point generation. The performance of the new version of the algorithm called the SNTO-Bees Algorithm was evaluated by computer simulation to solve a number of benchmark problems. The results obtained were compared with those of other optimisation techniques including the deterministic simplex method (SIMPSA), the stochastic simulated annealing (NESIMPSA), the Genetic Algorithm (GA), the Ant Algorithm (ANT), the original Bees Algorithm and the original PSO Algorithm to assess the success of the proposed methods.

- Each new algorithm was theoretically analysed using the results to show whether it converges on either a local or global minima, depending on the nature of the problem.

- Empirical result was obtained using many synthetic benchmark functions with well-known characteristics. These results are used as supporting evidence for the performance of the algorithms. It was possible to see whether the algorithm is still making progress towards its goal, or whether it has become trapped in local minima.

- The task of training both summation and product unit neural networks was selected as an example of real-life optimisation problem. On these problems, the results of the PSO-Bees Algorithm were compared to those of the Bees Algorithm, the PSO Algorithm and the well established back-propagation method.

- The task of solving mechanical design optimisation problems was selected as a real-life problem. On these problems, the results of the *improved* Bees Algorithm and the SNTO-Bees Algorithm were compared to those of the APPROX method, the Geometric Programming (GP), the Genetic Algorithm (GA), the Improved Genetic Algorithm and the SIMPLEX method.

1.4 Thesis Outline

In view of the fact that this research is about optimisation algorithms, Chapter 2 starts with a detailed introduction to the concept of optimisation. This is followed by a comprehensive assessment of the causes of problems in optimisation and optimality conditions. An in-depth evaluation of two nature-inspired optimisation algorithms is discussed: the novel Bees Algorithm and the Particle Swarm Optimisation Algorithm. A comparison between the Particle Swarm Optimisation Algorithm and Evolutionary Computation concludes the chapter.

Chapter 3 starts with an introduction to the PSO-Bees Algorithm. The parameters of the algorithm are explained. This is followed by a description of a number of stopping criterion that can be used on the PSO-Bees Algorithm. The performance measures used to compare the robustness, flexibility and effectiveness of the algorithm are also presented. The results obtained from training neural networks for control chart pattern recognition and wood defect classification problems are presented, inclusive of the results obtained by the algorithm on well-known mathematical benchmark test functions.

Chapter 4 describes the *improved* Bees Algorithm and its application to mechanical design optimisation problems, welded beams (single-objective and multi-objective) and coil springs with the results shown. The presentation of the results obtained from a number of mathematical benchmark problem concludes the chapter.

Chapter 5 presents the SNTO-Bees Algorithm. The algorithm is applied to mechanical design problems (design of welded beams, coil springs and pressure vessel), well-known mathematical benchmark functions and a number of deceptive multi-modal benchmark functions. The results obtained are presented.

Chapter 6 summarises the main contributions of this research and the conclusions reached. It also provides suggestions for future research.

Chapter 2: Background & Literature Review

Optimisation is one of the oldest of sciences, part of the art of successful living.

This chapter reviews the principle of Optimisation with attention focused on optimisation problem classification, optimality conditions and causes of problems affecting the performance of optimisation algorithms in general. The origin of the Bees Algorithm and the Particle Swarm Optimisation Algorithm is discussed. The chapter concludes with a comparison between the PSO Algorithm and Evolutionary Computation.

2.1 Optimisation

In a competitive world, only the best (fittest, safest, cheapest, fastest, etc) is good enough. This is why optimisation (local & global) is very frequent in applications. Optimisation is concerned with finding the *best* solution to a problem, where *best* refers to an acceptable (or satisfactory) solution, which can be the absolute best over a set of candidate solutions, or any other candidate solutions – this is explained in detail in the section on '*The Method of Inequalities* by (Weise 2008). Optimisation techniques are employed in diverse fields such as engineering, manufacturing, finance, medicine, computational art and music, chemistry, physics and economics. The task of optimisation is that of determining the values of a set of parameters so that some measure of optimality is satisfied subject to certain constraints.

The schematic of the optimisation process is shown in Figure 2.1 (Chinneck 2000).

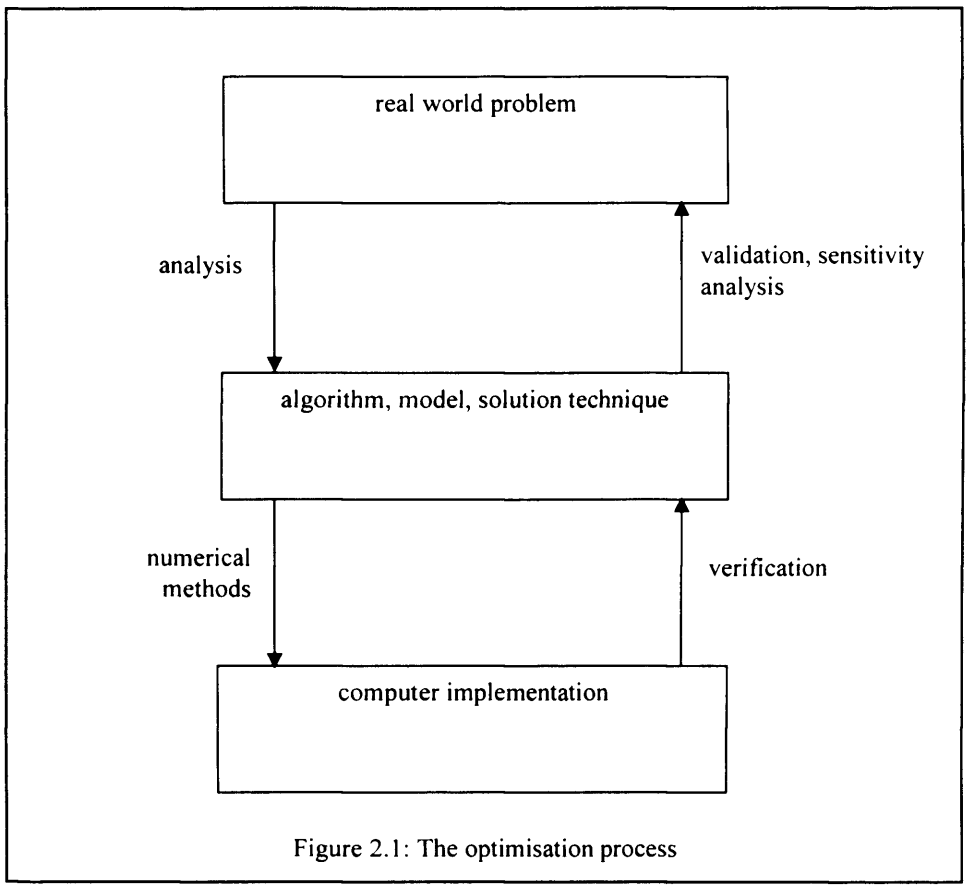


Figure 2.1: The optimisation process

From Figure 2.1, there is an unavoidable loss of realism as one moves down the diagram, from *real world problem* to *algorithm, model or solution technique* and finally to *computer implementation* while the arrows indicate the normal process of the optimisation cycle. Moving from the *real world problem* to the *algorithm, model or solution technique* is known as **analysis**. Here, the main work of abstracting away irrelevant details and focusing on important elements takes place.

Moving from the *algorithm, model, solution technique* to the *computer implementation* is called **numerical methods**. Moving back from *computer implementation* to the *algorithm, model, solution technique* is called **verification** and finally to *real world*

problem involves **validation and sensitivity analysis.** Here the obtained results are compared with the real world and an attempt is made to satisfy such queries as:

- Are the results appropriate?
- Do they make sense?
- Does the model need to be modified, or another solution technique need to be chosen?

Most of these problems involve linear models resulting in *linear optimisation* problems solved using linear programming (Greig 1980) while others are non-linear in nature that are difficult and tricky to solve. The term optimisation refers to problems in which one seeks to minimise or maximise a real function by systematically choosing values of real or integer variables from within an allowed set which is formally defined as:

Given:

a function $f: A \rightarrow \mathrm{R}$ from some set A to the real numbers

Sought:

an element x_0 in A such that

$f(x_0) \leq f(x)$ for all x in A ("minimisation")

$f(x_0) \geq f(x)$ for all x in A ("maximisation")

Typically, A is some subset of the Euclidean space R^n, often specified by a set of:

- *constraints*

- *equalities* or
- *inequalities*

that the members of A have to satisfy. The elements of A are called *feasible solutions*. The function f is called an *objective function*. A *feasible solution* that minimises (or maximises) the objective function is called an *optimal solution*. The domain A of f is called the *search space*, while the elements of A are called *candidate solutions* or *feasible solutions*.

Generally, when the feasible region or the objective function of the problem does not present convexity, there may be several local minima and maxima, where a *local minimum* x^* is defined as a point for which there exists some $\delta > 0$ such that for all x

$$\| x - x^* \| \leq \delta \tag{2.1}$$

and

$$f(x^*) \leq f(x) \tag{2.2}$$

holds. This means that in some region around x^*, all of the function values are greater than or equal to the value at that point. A local maxima is defined similarly.

The following section highlights the classification of optimisation problems (Engelbrecht 2005) based on a number of characteristics: the number of variables, type of variables, the degree of nonlinearity of the objective function, constraints used, number of optima and the number of optimisation criteria.

2.1.1 Optimisation Problem Classification (OPC)

This section identifies the following characteristics used to classify optimisation problems (Engelbrecht 2005):

- The **number of variables** that influences the objective function: A problem having a single variable to be optimised is referred to as a *univariate* problem. If more than one variable is considered, the problem is called a *multivariate* problem.
- The **type of variables**: By default, a continuous problem has continuous-valued variables, i.e. $x_j \in \mathrm{R}$, for all $j = 1,\ldots,n_x$. If $x_i \in \mathrm{Z}$, the problem is referred to as an integer or discrete optimisation problem. A mixed integer problem has both continuous-valued and integer-valued variables. Problems where solutions are permutations of integer-valued variables are classified as *combinatorial* optimisation problems.
- The **degree of nonlinearity of the objective function:** Linear problems have objective functions with linear variables. Quadratic problems use quadratic functions and when other non-linear objective functions are used, the problem is classified as a nonlinear problem.
- The **constraints used:** A problem using just boundary constraints is categorised as an unconstrained problem while constrained problems have additional equality and / or inequality constraints.
- The **number of optima:** If there is only one clear solution, the problem is *unimodal*. On the other hand, when more than one optimum exists, the problem is *multi-modal*. Some other problems may have false optima in which case the problem is classified as being *deceptive*.

- The **number of optimisation criteria:** A problem is categorised as a *uni-objective* (single objective) when the quantity to be optimised is expressed using only one objective function. A multi-objective problem has more than one sub-objective that must be optimised simultaneously.

The optimisation techniques used to solve the optimisation problem classifications defined above can be placed into two categories: Local and Global optimisation algorithms.

2.1.2 Optimality Conditions (OC)

The solutions found by optimisation algorithms are typically categorised by the quality of the solution. The main types of solutions are referred to as local optima or global optima (Bergh 2001; Engelbrecht 2005).

2.1.2.1 Local Optimisation (LO)

A local minimiser, x^*_B, of the region B, is defined as:

$$f(x^*_B) \leq f(x), \ \forall x \in B \tag{2.3}$$

where $B \subset S \subseteq \mathrm{R}^n$, and S denotes the search space when dealing with unconstrained problems $S = \mathrm{R}^n$. Note that B is a proper subset of S. A search space S can contain multiple regions B_i such that $B_i \cap B_j = 0$ when $i \neq j$. It then follows that $x^*_{Bi} \neq x^*_{Bj}$, so that the minimiser of each region B_i is unique. Any of the x^*_{Bi} can be considered a minimiser of B, though they are local minimisers. There is no restriction on the value that the

function can assume in the minimiser, so that $f(x^*_{Bi}) = f(x^*_{Bj})$ is allowed. The value $f(x^*_{Bi})$ is called the local minimum.

While most optimisation algorithms require a starting point $z_0 \in S$, a local optimisation algorithm needs to guarantee that it will be able to find the minimiser x^*_B of the set B if $z_0 \in B$. Some selected algorithms satisfy a slightly weaker constraint in that they guarantee to find a minimiser x^*_{Bj} of some set B_i, not necessarily the one closest to z_0 as shown in Figure 2.2 (Engelbrecht 2005).

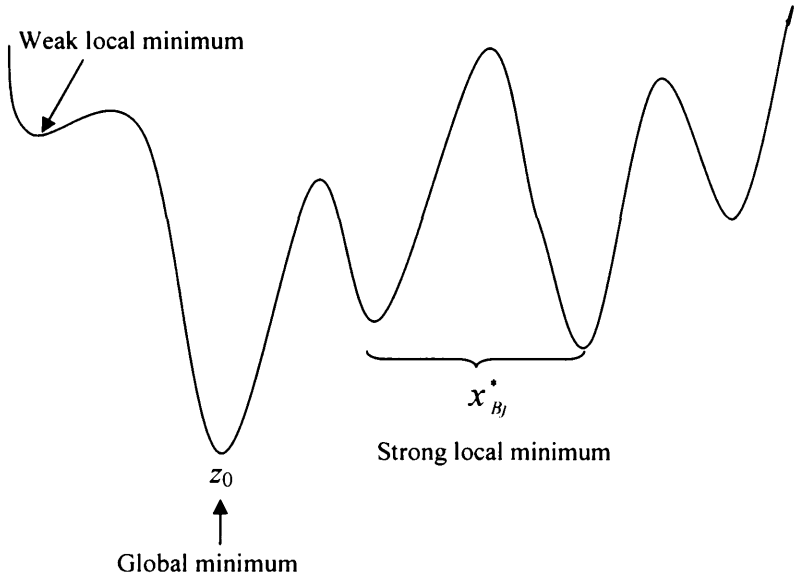


Figure 2.2: Types of Optima for Unconstrained Problems

Weak local minimum: The solution $x_B^* \in B \subseteq S$, is a weak local minimum of f if

$$f(x_B^*) \leq f(x), \ \forall x \in B \tag{2.4}$$

Where $B \subseteq S$ is a set of feasible points in the neighbourhood of x_B^*.

2.1.2.2 Global Optimisation (GO)

The solution $x^* \in S$, is a global optimum of the objective function, f, *if*

$$f(x^*) < f(x), \ \forall x \in S \tag{2.5}$$

where $B \subseteq S$.

The global optimum is the best of a set of candidate solutions as shown in Figure 2.2 for a minimisation problem. This global algorithm starts by choosing an initial starting position $z_0 \in S$.

The copious factors identified by (Weise 2008, 2009) that impinge negatively on the performance of optimisation algorithms are discussed next - Problems in Optimisation.

2.1.3 Problems in Optimisation

In section 2.1.1, the classifications of optimisation problems are highlighted. It is therefore worth mentioning the reasons for these varied classifications. A probable cause can be attributed to numerous kinds of optimisation tasks. These tasks present varied impediments in the paths of the optimisers and also each task has its own characteristic complexity and difficulties. This mostly concerns global optimisation in general (e.g.

multi-modality, overfitting); others apply especially to nature-inspired approaches like genetic algorithms (e.g. epistasis, neutrality). As a result, neglecting even a single issue in *sections 2.1.3.1* through *section 2.1.3.9* during the design / process of optimisation can render the whole effort invested futile, even if the most efficient optimisation techniques are applied. These include (Weise 2008, 2009):

2.1.3.1 Premature Convergence (PC)

Convergence: An optimisation algorithm has converged if it keeps on producing solutions from a "small" subset of the problem space or if it cannot reach new solution candidates anymore (Schaffer *et al.* 1990).

As a standard, global optimisation algorithms need to converge at a moment in time. However, one of the most important problems in global optimisation is we generally cannot determine whether the best solution currently known is a local or a global optimum and there is also the dilemma whether its convergence is acceptable or not. In other words, we are not able to say whether we can stop the optimisation process, or we should concentrate on refining our current optimum, or whether we should examine and explore other areas of the search space. Furthermore, premature convergence can also occur when there are multiple (local) optima in which case it is a multi-modal problem.

Multimodality: A set of objective functions f is multi-modal, if it has multiple local or global optima (Shekel 1971).

Premature Convergence: An optimisation process has prematurely converged to a local optimum if it is no longer able to explore other parts of the search space than the currently examined area and there exists another region that contains a solution superior to the currently exploited one (Schaffer *et al.* 1990).

Premature convergence can occur when an optimisation algorithm passes by several local optima in the objective space before reaching a good solution. As a result, it is most likely to get stuck on such an intermediate solution and would not be able to proceed to search other areas in the solution space. Each optimisation algorithm has features and parameter settings that help to influence its convergence behavior (Rudolph 1997). Some algorithms are capable of self-adaptation, allowing them to change their strategies or to modify their parameters depending on its current state and environment. Such behavior is often implemented in order to speed up the optimisation process, but may lead to premature convergence onto local optima (Rudolph 1999, 2001). A possible resolution would be to randomly restart the optimisation process at some chosen points in time. Although crude, it is sometimes an effective measure against premature convergence. Also worth mentioning is domino convergence.

Domino Convergence (DC): Domino convergence occurs when the solution candidates have features which contribute to significantly different degrees of the total fitness. When these features are encoded in separate genes (or building blocks) in the genotypes, there is likelihood that they will be treated with different priorities in randomised or heuristic optimisation methods. The building blocks having a very strong positive influence and stimulus on the objective values will most likely be adopted first by the optimisation

process ("converge") while at the same time, the alleles of genes, having smaller contributions, play no role. This is because the alleles of genes, having smaller contributions, do not come into play until the more "important" blocks have been accumulated. Rudnick (Rudnick 1992) called this sequential convergence phenomenon "domino convergence" due to its resemblance with a row of falling domino stones (Thierens *et al.* 1998). Also worth mentioning is that the relationship between exploration and exploitation influences convergence.

Exploration vs. Exploitation: From (Eshelman and Schaffer 1991; Smith 2004), the procedure that creates new solutions from existing ones has a very large impact on the balance between exploration and exploitation. For instance, the "step size" setting influences how an optimisation algorithm solves the balancing problem between exploration and exploitation.

(Eiben and Schippers 1998; Muttil and Liong 2004) researched the trade-off between exploration and exploitation that optimisation algorithms have to face.

Exploration: in terms of optimisation it means finding new points in areas of the search space which have not yet been investigated.

Exploration is the only means to finding a new or an even better solution. Until the algorithm finds a better solution – which is not guaranteed – the performance of the optimisation process degenerates because we are evaluating solution candidates inferior to the ones we already know.

Exploitation: in terms of optimisation it means trying to improve the currently known solution(s) by performing small changes which lead to new individuals which are very close to them.

The process of exploitation often results in performance improvements since the chance of finding better solutions which are similar to the already known individuals is often good. Conversely, better solutions located in distant areas of the solution space, would not be discovered by minor refinements. Occasionally, some parts of optimisation strategies can be used either for increasing exploitation or in favour of exploration. For instance, unary search operations can be built to improve an existing solution in small steps, hence being exploitation operators. On the other hand, it can also be implemented in a way that introduces much randomness into the individuals, thus performing exploration operations.

Generally, the algorithms that favour exploitation have high convergence speed but run the risk of not finding the optimal solution and can get stuck at a local optimum. On the other hand, algorithms that perform excessive exploration may find the global optimum but it will take them a very long time to do so.

Diversity: Exploitation and exploration are directly linked with diversity: exploration increases the diversity whereas exploitation works against it. As a result, diversity preservation is a major concern in optimisation. The loss of diversity can lead to premature convergence onto a local optimum. Because of its effect and consequence, this has been widely studied by researchers; in Genetic Algorithms (Ronald 1996), in

Evolutionary Algorithms (Singh and Deb 2006), in Genetic Programming (Burke *et al.* 2002b) and in Particle Swarm Optimisation (Wilke *et al.* 2007).

2.1.3.2 Ruggedness and Weak Causality

Ruggedness: Most optimisation algorithms depend on some form of gradient in the objective or fitness space. Occasionally, the objective function is continuous and exhibits low total variation to enable the optimiser to descend the gradient easily. On the other hand, if the objective function fluctuates up and down, it becomes more difficult for the optimiser to find the right direction to proceed in. In short, the more rugged a function gets, the harder it is to optimise (ruggedness is multi-modality plus steep ascents and descents in the fitness landscape).

Strong Causality: The principle of strong causality (locality) proposed by Rechenberg (Rechenberg 1994) states that small changes in an object lead to small changes in its behaviour.

During an optimisation process, new points in the search space are found by the search operations. Generally it can be assumed that the genotypes are the input of the search operations corresponding to the phenotypes which have previously been selected. The chance of being selected for further investigation is usually the higher the better or the more promising an individual is. This statement implies that individuals which are passed to the search operations are likely to have a good fitness. As the fitness of a solution candidate depends on its properties, it is assumed that their properties were not so bad

either. It is thus possible for the optimiser to perform slight changes to these properties in order to find out whether they can be improved further.

On the other hand, if we consider fitness landscapes with weak (low) causality, small changes in the solution candidates often lead to large changes in the objective values, i.e. ruggedness. This makes it very difficult to come to a decision as to what area of the solution space to explore, thereby making it impossible for the optimiser to consistently find any reliable gradient information to follow. Consequently, small modifications of a very bad solution candidate will most likely lead to a new local optimum and the best solution candidate currently known may be surrounded by points that are inferior to all other tested individuals.

2.1.3.3 Deceptiveness

Deceptiveness is one of the upsetting features of the fitness landscapes. As the name implies, the gradient of the deceptive objective function leads the optimiser away from the global optima.

The deceptiveness idiom is employed frequently in the Genetic Algorithm community in the context of the Schema Theorem where schemas describe particular areas (hyper planes) in the search space. When an optimisation algorithm has discovered an area with a superior average fitness in contrast to other regions, logically it focuses on exploring this area with certainty to converge on the true optimum. Dissimilar objective functions are said to be deceptive (Liepins and Vose 1991).

2.1.3.4 Neutrality and Redundancy

Neutrality: The outcome of a search operation to a solution candidate is neutral if it yields no change in the objective values (Barnett 1998).

For all optimisation algorithms, it is problematic when the best solution candidate currently found is located on a plane of the fitness landscape. This implies that all other adjacent solution candidates have the same objective values. Thus, there is neither gradient information nor direction into which the optimisation algorithm can proceed in a systematic manner. As a result, each search operation will yield identical individuals. The possible solution to this is for optimisation algorithms to maintain a list of the best candidates found, which will sooner or later overflow and require pruning.

Evolvability: is a metaphor in global optimisation taken from biological systems (Dawkins 1987). According to Wagner (Wagner 2005), this word has two uses in biology. A biological system is evolvable if it is able to generate heritable, selectable phenotypic variations (Kirschner and Gerhart 1998). Such properties can then evolve and change through natural selection. In the second meaning, a system is evolvable if it can acquire new characteristics via genetic change that help the organism(s) to survive and to reproduce. (Weise 2008) adopted the idea of evolvability for global optimisation as follows:

> *The evolvability of an optimisation process in its current state defines how likely the search operations will lead to solution candidates with new (and eventually, better) objectives values.*

The direct probability of success (Beyer 1994) - the chance that search operators produce offspring that are fitter than their parents, is also sometimes referred to as evolvability in the context of evolutionary algorithms (Altenberg 1994).

Many researchers disagree and argue the converse concerning this suggested link between evolvability and neutrality, maintaining that the evolvability of neutral parts of a fitness landscape is dependent on the optimisation algorithm used. For hill climbing and similar approaches, this dependence is low because the search operations cannot provide improvements (or even changes). The optimisation process is then reduced to a random walk.

Neutral Networks

The concept of neutral networks was derived from the idea of neutral bridges between different parts of the search space as sketched by (Smith *et al.* 2002).

By definition, neutral networks are equivalence classes K of elements of the search space G which map to elements of the problem space X with the same objective values and are connected by chains of applications of the search operators (Barnett 1998). According to Barnett (Barnett 1998), a neutral network has the property of constant innovation if:

- the rate of discovery of innovations keeps constant for a reasonably large amount of applications of the search operations (Huynen 1996).
- if this rate is comparable with that of an unconstrained random walk.

Stewart (Stewart 2001) utilised neutral networks and the idea of punctuated equilibria in his extrema selection, where a Genetic Algorithm variant focusing on exploring individuals has good objective values that are located further away from the centroid of the set of the currently investigated solution candidates.

Bornberg-Bauer and Chan (Bornberg-Bauer and Chan 1999), van Nimwegen (Nimwegen *et al.* 1999), and Wilke (Wilke 2001) studied the convergence of neutral networks. The outcome of their results illustrate that the topology of neutral networks strongly determines the distribution of genotypes, while from (Beaudoin *et al.* 2006) the genotypes are "drawn" to the solutions with the highest degree of neutrality on the neutral network.

Redundancy: is defined in the context of global optimisation as a feature of the genotype-phenotype mapping. It means that multiple genotypes map to the same phenotype (the genotype-phenotype mapping is not injective, which means a one-to-one function). Mathematically, this is written as:

$$\exists\, g_1, g_2 : g_1 \neq g_2 \;\wedge\; gpm(g_1) = gpm(g_2) \tag{2.6}$$

Where g1, g2 are the genotype (elements of the search space) and 'gpm' is the genotype-phenotype mapping.

The role of redundancy in the genome is as controversial as that of neutrality. There are numerous accounts of its positive influence on the optimisation process. Shipman (Shipman *et al.* 2000) and Huynen (Huynen *et al.* 1996) developed redundant genotype-phenotype mapping using:

- voting
- turing-machine like binary instructions

- cellular automata
- random Boolean networks (Kauffman 1993)

All four mappings produced neutral networks which proved beneficial for exploring the problem space. One of the possibly converse effects is *epistasis*.

Redundancy has significant impact on the explorability of the search space. In a one-to-one mapping, the translation of a slightly modified genotype often results in a different phenotype. Conversely, if there exists a many-to-one mapping between genotypes and phenotypes, the search operations can create offspring genotypes different from the parent, which still translate to the same phenotype. The effect will cause the optimiser to stride along a path through this neutral network. In this case, when many genotypes along this path are modified to different offspring, it often results in many new solution candidates being reached (Shipman *et al.* 2000).

2.1.3.5 Epistasis

From biology, *epistasis* is described as a form of relations or interactions between different genes (Phillips 1998). The term was originally invented by Bateson (Bateson 1909), meaning that one gene suppresses the phenotypical expression of another gene. Fisher (Fisher 1918) called *epistasis* "epistacy" in the context of statistical genetics. From (Lush 1935), the interaction between genes is epistatic if the effect on the fitness from altering one gene depends on the allelic state of other genes. The knowledge and perception of epistasis comes very close to another biological expression: pleiotropy, which means a single gene influences multiple phenotypic traits (Williams 1957). In global optimisation, there is no such fine-grained distinction.

Epistasis (Davidor 1990; Naudts and Verschoren 1996) in global optimisation means that a change in one gene of a genotype, introduced by a search operation for instance, leads to changes in multiple, otherwise unrelated, phenotypical properties. A minimal epistasis occurs when every gene is independent of every other gene. A maximal epistasis arises when no proper subset of genes is independent of any other gene (Naudts *et al.* 2000). For a genome with high epistasis, a modification in a genotype will alter multiple properties of the corresponding phenotype. Naudts and Verschoren (Naudts and Verschoren 1999) showed that deceptiveness does not occur in situations with low epistasis and also that the objective functions with high epistasis are not necessarily deceptive on the example of length-two binary string genomes.

2.1.3.6 Overfitting and Oversimplification

In circumstances where optimisers appraise some of the objective values of the solution candidates by using training data, two phenomena with negative influence have been detected: *Overfitting* and *Oversimplification*.

Overfitting is defined as the emergence of an overly-complicated model (solution candidate) in an optimisation process resulting from the effort to provide the best possible results for as much of the available training data as possible (Dietterich 1995; Sarle 1997).

A model (solution candidate) $m \in X$ that is produced with a finite set of training data is considered to be overfitted if a less complicated, alternative model $m' \in X$, $m' \neq m$ exists which has a smaller error for the set of all possible producible data samples. The

model m′ may have a larger error in the training data. Yet again, the phenomenon of Overfitting is encountered in the field of artificial neural networks (ANN) or in curve fitting (Lawrence and Giles 2000; Ling 1995; Sarle 1995; Tetko *et al.* 1995). The latter imply that we have a set A of n training data samples (x_i, y_i) and we need to find a function f that represents these samples as well as possible, that is:

$$f(x_i) = y_i \ \forall (x_i, y_i) \in A \tag{2.7}$$

To be precise, there is one polynomial of degree $n - 1$ that fits to such training data and goes through all its points. When it is restricted to polynomial regression, there is one global optimum, *single perfect fit*. On other occasion there is the likelihood of having an infinite number of polynomials with a higher degree than $n - 1$ that also matches the sample data perfectly – this is considered as *overfitted.* A very common cause for Overfitting is noise present in the sample data for which there is no measurement device for physical processes that delivers perfect results without error. Additionally, in opinion surveys of people working in various fields or with randomised simulations reveal variations from the true interdependencies of the observed entities. Hence, the data samples A based on measurements will always contain some noise.

The major problem resulting from overfitted solutions is the *loss of generality.*

Generality: by definition, the solution of an optimisation process is "general" if it is not only valid for the sample inputs $x_1, x_2, \ldots, x_n$ which were used for training during the optimisation process but also for different inputs $x \neq x_i \ \forall_i : 0 < i \leq n$ if such inputs x exist. Hence, a solution is also general if it is valid for all possible inputs.

Overfitting Prevention

There are multiple techniques used to prevent overfitting to a certain degree. It is most effective to apply multiples of such techniques together in order to achieve the best results. The following as identified by (Weise 2008) are known to be helpful in preventing overfitting:

1. ***Restriction of the Problem Space***: restricting the problem space X in a way that only solutions up to a given maximum complexity can be found.

2. ***Additional Optimisation Criteria***: the functional objective function that solely concentrates on the error of the solution candidates needs to be augmented by penalty terms and the secondary objective functions need to put pressure into the direction of small and simple models (Dietterich 1995).

3. ***Training Large Data Sets*:** although this slows down the optimisation process, at times it may improve the generalisation capabilities of the solutions derived.

4. ***Changing Training Data / Simulation Scenarios***: there are two approaches to prevent overfitting provided arbitrary training datasets or training scenarios can be generated:
   - Use a new set of (randomised) scenarios for each evaluation of a solution candidate. Here, the resulting objective values may differ largely even if the same individual is evaluated twice in a row with the introduction of ruggedness and incoherence into the fitness landscape.

- At the beginning of each iteration of the optimiser, generate a new set of (randomised) scenarios which is used for all individual evaluations during that iteration.

In both cases, it is important to use more than one training sample or scenario per evaluation and set the resulting objective value to the average (or better median) of the outcomes. Otherwise, the fluctuations of the objective values between iterations will be very large, making it hard for the optimisers to follow a stable gradient for multiple steps.

5. ***Early Stopping***: A very simple method to prevent overfitting is to limit the runtime of the optimisers (Sarle 1997). It is commonly assumed that learning processes normally first find relatively general solutions which subsequently begin to overfit due to the presence of noise.

6. ***Decay of Influence*:** Some algorithms allow the decreasing of the rate at which the solution candidates are modified depending on time. Such a decay of the learning rate makes overfitting less likely.

7. **Dividing Data into Training and Test Sets**: When only a finite set of data samples D is available for training / optimisation, it is a regular practice to separate the data into a set of training data D_t and a set of test cases D_c. During the optimisation process, only the training data D_t is used. The resulting solutions are then tested with the test cases D_c. If their behavior is significantly worse when applied to D_c than when applied to D_t, they are probably overfitted.

A similar methodology is used to detect when to stop the optimisation process. The best known solution candidates can be checked with the test cases in each iteration, without influencing their objective values that exclusively depend on the training data. Besides, there is the need to stop if the performance on the test cases begins to diminish.

Oversimplification

Oversimplification (at times called overgeneralisation) is the opposite of overfitting. Despite the fact that overfitting symbolises the emergence of overly-complicated solution candidates, oversimplified solutions are not complicated enough. While it properly represents the training samples used during the optimisation process, oversimplification are rough overgeneralisations which fail to provide good results for scenarios not part of the training.

Among the general causes for oversimplification are often training data sets that only represent a fraction of the set of possible inputs. Such an incomplete coverage may possibly fail to represent some of the dependencies and characteristics of the data which leads to oversimplified solutions.

2.1.3.7 Robustness and Noise

Robustness

In global optimisation, we always seek the global optima of the objective functions from the theoretical point of view but not from the practical point of view. One reason for this is that the solutions of practical problems often rely on parameters which can only be identified if they allow a certain degree of imprecision (there is no process in the world

that is 100% accurate). Local optima in regions of the search space with strong causality are sometimes better than global optima with weak causality while the level of acceptability is application-dependent.

For the special case where the problem space corresponds to the real vectors ($X \subseteq R^n$), several approaches for dealing with the problem of robustness have been developed inspired by Taguchi methods (Taguchi 1986).

Noise

There are two types of noise in optimisation:

1. There is noise in the training data that is used as the basis for learning which cause overfitting. This noise results because no measurement is 100% accurate and noise always exists when trying to fit a model to measured data.
2. The second form of noise subsumes the perturbations that are expected to occur in the subsequent process – the reason why the best robust solutions and not just the globally optimal ones are needed. This category is subdivided into perturbations that may arise from inaccuracies in:
    - the process of realising solutions
    - environmentally induced perturbations

2.1.3.8 Dynamically Changing Fitness Landscape (DCFL)

There exist some optimisation problems having dynamically changing fitness landscapes (Branke 1999; Branke *et al.* 2005; Richter 2004). The goal and purpose of an optimisation algorithm applied to a dynamically changing fitness landscape is to produce

solution candidates with momentarily optimal objective values for each point in time. An optimum in iteration t will no longer be an optimum in iteration $t + 1$. In the literature (Weise 2008), these problems have been solved using evolutionary algorithms (Aragon and Esquivel 2004; Branke 2001; Morrison 2004), genetic algorithms (Gobb and Grefenstette 1993; Mori *et al.* 1997), Particle Swarm Optimisation (Blackwell 2007; Carlisle and Dozier 2002), Differential Evolution (Mendes and Mohais 2005) and Ant Colony Optimisation (Guntsch and Middendorf 2001).

Branke (Branke 1999) and Morrison and DeJong (Morrison and DeJong 1999) are typically good examples of dynamically changing fitness landscapes.

2.1.3.9 No Free Lunch Theorem (NFL)

The No Free Lunch Theorem means that there is no optimisation algorithm that can outperform all others on all problems (Igel and Toussaint 2003; Köppen *et al.* 2001; Wolpert and Macready 1997). There is a variety of optimisation methods specialising on solving different types of problems. Also, there are algorithms that deliver good results for many different problem categories, but are outperformed by highly specialised methods in each of them. These facts were stated by Wolpert and Macready (Wolpert and Macready 1997) in their No Free Lunch Theorems (NFL) for search and optimisation algorithms.

2.2 The Bees Algorithm

Researchers at the Manufacturing Engineering Centre (MEC) in Cardiff University, myself included, under the supervision of Prof. D.T. Pham (Pham *et al.* 2005, 2006a)

developed the Bees Algorithm after observing the "waggle dance" of bees foraging for nectar. The application of this ingenious new mathematical procedure based on the behaviour of honey bees has delivered excellent results for the industry by enabling companies to maximise results by changing basic elements of their processes and also to establish the most effective way to set up their machines. This has saved money through running their processes as efficiently as possible.

When a bee finds a source of nectar, it returns to the hive and performs a special dance (called waggle dance) to show other bees the direction and distance of the flower patch and how plentiful it is (see Figure 2.3a from (Felix *et al.* 2007)). The other workers then decide how many of them will fly off to find the new source, depending on its distance and quality (see Figure 2.3b, c and d from (Ratnieks 2008; Seeley 1996; Seeley *et al.* 2006)).

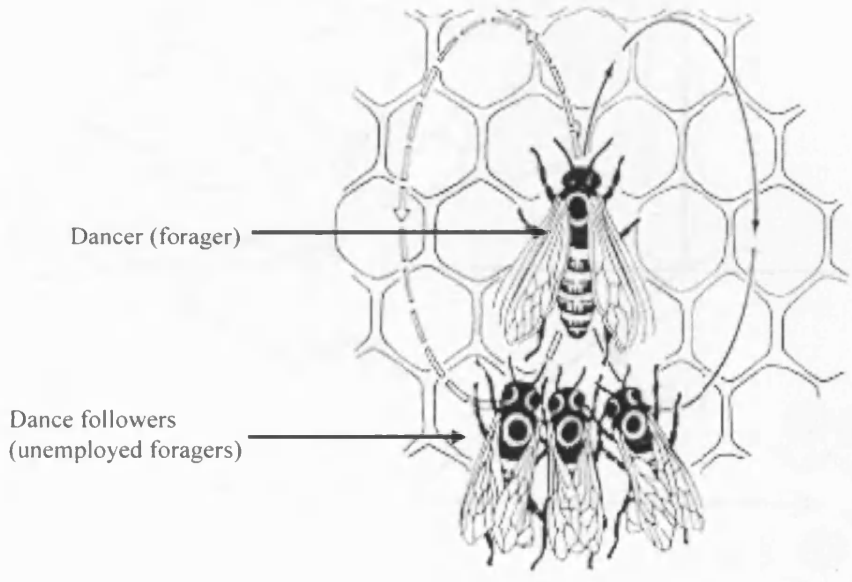


Figure 2.3a: Waggle dance of honey bees

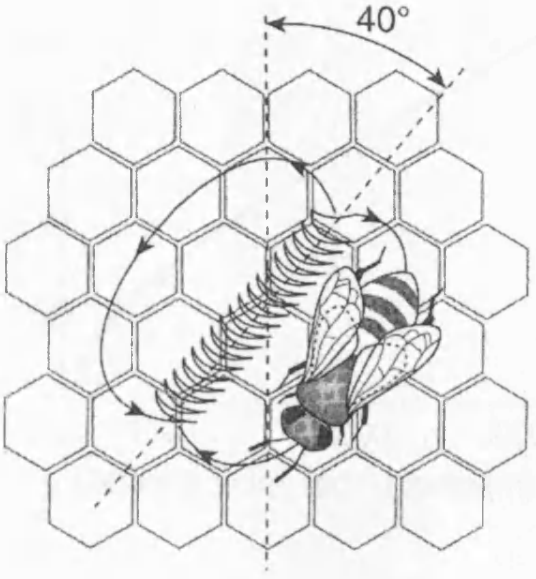


Figure 2.3b: Waggle dance - angle of dancing bee to vertical

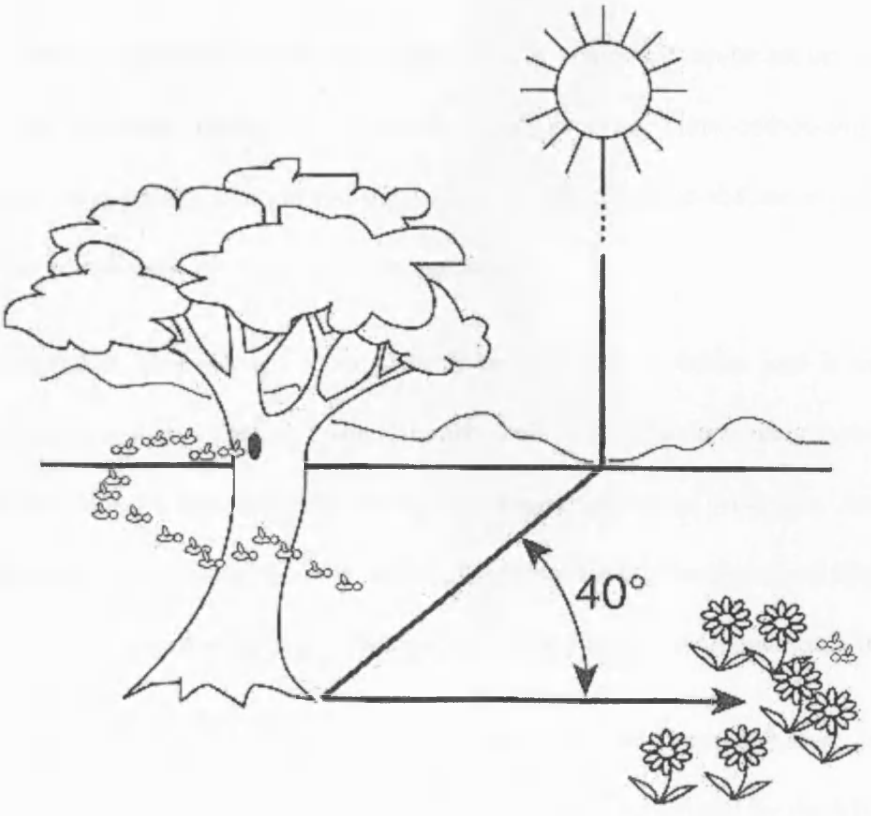


Figure 2.3c: Waggle dance – angle of flowers to sun

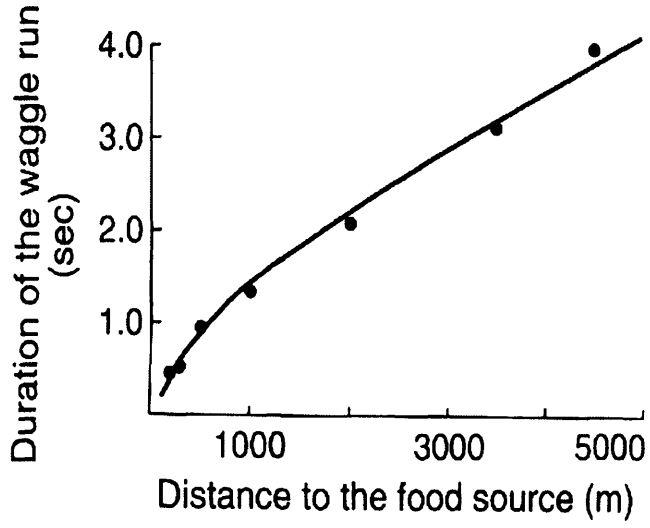


Figure 2.3d: Waggle dance duration encodes distance

The MEC's Bees Algorithm mimics this behaviour. A computer can be set up to calculate the results of different settings on a manufacturing process. More computing power is then devoted to searching around the most successful settings, in the same way as more bees are sent to the most promising flower patches.

The algorithm has been shown to cope with up to 3,000 variables and is faster than existing calculations. By entering basic data about all or part of a company's processes, it is easy to calculate the best outcome for a wide range of business processes. An example is its application to determine the most efficient settings on the design of welding systems and for the design of coiled springs (Pham *et al.* 2008; Pham and Ghanbarzadeh 2007).

The algorithm was unveiled by the MEC team at the internet-based Innovative Production and Machines and Systems (IPROMS) Conference hosted by the MEC as part of its work with the EU-funded Network of Excellence in this field (Pham *et al.* 2006a).

Bees in Nature

A colony of honey bees can extend itself over long distances (more than 14 km) and in multiple directions simultaneously to exploit a large number of food sources (Frisch 1976). A colony prospers by deploying its foragers to good fields. In principle, flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more bees, whereas patches with less nectar or pollen should receive fewer bees.

The foraging process begins in a colony by scout bees being sent to search for promising flower patches. Scout bees move randomly from one patch to another. During the harvesting season, a colony continues its exploration, keeping a percentage of the population as scout bees.

When they return to the hive, the scout bees that have found a patch which is rated above a certain quality threshold (measured as a combination of some constituents, such as sugar content) deposit their nectar or pollen and go to the "dance floor" to perform a dance known as the "waggle dance".

This mysterious dance is essential for colony communication, and contains three pieces of information regarding a flower patch:

- the direction in which it will be found
- its distance from the hive and
- its quality rating (or fitness)

This information helps the colony to send its bees to flower patches precisely, without using guides or maps. Each individual's knowledge of the outside environment is gleaned solely from the waggle dance. This dance enables the colony to evaluate the relative merit of different patches according to both the quality of the food they provide and the amount of energy needed to harvest it. After waggle dancing on the dance floor, the dancer (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive (see Figure 2.3b, c and d from (Ratnieks 2008; Seeley 1996; Seeley *et al.* 2006)). More follower bees are sent to more promising patches. This allows the colony to gather food quickly and efficiently.

While harvesting from a patch, the bees monitor its food level. This is necessary to decide upon the next waggle dance when they return to the hive. If the patch is still good enough as a food source, then it will be advertised in the waggle dance and more bees will be recruited to that source.

The Pseudo code for the Bees Algorithm

The Bees Algorithm is an optimisation algorithm inspired by the natural food foraging behaviour of honey bees to find the optimal solution. Figure 2.4 shows the pseudo code for the algorithm in its simplest form. The algorithm requires a number of parameters to be set, namely:

- number of scout bees (n)
- number of sites selected out of n visited sites (m)
- number of best sites out of m selected sites (e)

- number of bees recruited for best e sites (*nep*)
- number of bees recruited for the other (m-e) selected sites (*nsp*)
- initial size of patches (*ngh*), which includes the site and its neighbourhood and a stopping criterion.

```
1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met).
   //Forming new population.
4. Select patches for neighbourhood search.
5. Recruit bees for selected patches (more bees for best patches)
   and evaluate their fitness.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their
   fitness.
8. End While
```

Figure 2.4: Pseudo code of the basic Bees Algorithm

The algorithm starts with the *n* scout bees being placed randomly in the search space. The fitnesses of the sites visited by the scout bees are evaluated in step 2.

In step 4, the bees that have the highest fitness are chosen as "selected bees" and the sites visited by them are chosen for neighbourhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected sites, assigning more bees to search near to the best '*e*' sites. The bees can be chosen directly according to the

fitnesses associated with the sites they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighbourhood of the best '*e*' sites which represent more promising solutions are made more detailed by recruiting more bees to follow them than the other selected bees. Together with scouting, this differential recruitment is a key operation of the Bees Algorithm.

However, in step 6, for each patch only the bee with the highest fitness will be selected to form the next bee population. In nature, there is no such restriction. This restriction is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to its new population - representatives from each selected patch and other scout bees assigned to conduct random searches.

A review of the Bees Algorithm is presented in Appendix H.

2.3 Particle Swarm Optimisation (PSO)

The Particle Swarm Optimisation (PSO) Algorithm is a population-based stochastic optimisation technique developed by Eberhart and Kennedy (Eberhart and Kennedy 1995; Kennedy and Eberhart 1995a) and inspired by the social behaviour of birds flocking or fish schooling (Bonabeau *et al.* 1999). PSO has its roots in artificial life and in social psychology, as well as in engineering and computer science. It utilises a "population" of particles that fly through the problem hyperspace with given velocities.

In each iteration, the velocities of the individual particles are stochastically adjusted according to the historical best position for the particle itself and the neighborhood best position. In other words, a swarm consists of individuals, or particles, which change their positions over time. Each particle represents a potential solution to the given optimisation problem. These particles "fly" freely in the multi-dimensional search space and during its flight each particle adjusts its position according to its own experience and that of neighbouring particles, based on the best positions encountered by itself and its neighbours. The effect is that particles move towards good solution areas, while still having the ability to search around those areas. The performance of each particle is measured according to a pre-defined fitness function related to the given problem (Eberhart and Kennedy 1995).

PSO has some advantages over other optimisation techniques such as the GA, namely:

- PSO is easier to implement and there are fewer parameters to adjust.
- In PSO, every particle remembers its own previous best value as well as the neighborhood best; therefore, it has a more effective memory capability than the GA.
- PSO is more efficient in maintaining the diversity of the swarm - more similar to the ideal social interaction in a community (Engelbrecht 2006), since all the particles use the information related to the most successful particle in order to improve themselves, whereas in GA, the worst solutions are discarded and only the good ones are saved. Therefore, in GA the population evolves around a subset of the best individuals.

The Particle Swarm Optimisation Algorithm makes use of two fundamental branches of learning or fields: *social science* and *computer science*. Furthermore, the PSO Algorithm uses the *Swarm Intelligence* theory - a system exhibiting the cooperative and communal actions / conduct of unsophisticated agents that interact locally with their environment, creating coherent global functional patterns. The fundamentals of the PSO can be described as follows:

1) ***Social Concepts*** *(Eberhart et al. 2001):* As a matter of fact, "*human intelligence results from social interaction*". Evaluation, comparison, and imitation of others, as well as learning from experience, allow the human race to acclimatise to the environment and establish the most favourable patterns of behaviour and attitudes. Moreover, a second fundamental social concept states that "*culture and cognition are inseparable consequences of human sociality.*" This is because culture is generated when individuals become more similar due to mutual social learning. The sweep of culture allows individuals to move towards more adaptive patterns of behaviour.

2) ***Swarm Intelligence Principles*** *(Eberhart et al. 2001; Eberhart and Kennedy 1995; Kennedy and Eberhart 1995a)*: Swarm Intelligence is based on five fundamental principles:

a) *Proximity Principle*: ability to perform simple space and time computations by the population.

b) *Quality Principle*: ability to respond to quality factors in the environment by the population.

c) *Diverse Response Principle*: the population should not confine its activity along excessively narrow channels.

d) *Stability Principle*: the population should not alter its mode of behaviour every time the environment changes.

e) *Adaptability Principle*: the population should be able to adjust its behaviour mode when it is worth the computational price.

The PSO Algorithm has been successfully applied to a number of optimisation problems (Carlisle and Dozier 2002; Eberhart *et al.* 2001; Xu and Eberhart 2002a, b). Often, PSO can produce better results faster, more simply and more robustly in comparison with other methods. As mentioned above, this is because the PSO Algorithm has relatively few parameters to adjust and is not overly sensitive to the choice of parameter values.

Figure 2.5 shows the pseudo code for the PSO Algorithm in its simplest form.

```
For each particle
    Initialise velocity V0 and position P0
End
Do For each particle
    Calculate fitness value
    If fitness better than Pbest
        Update Pbest
End
    Determine Gbest among all particles
    For each particle
        Update velocity
        Update position
End
While maximum iterations are not exceeded or
      minimum error is not attained
```

Figure 2.5: Pseudo code of PSO Algorithm

In every iteration, the position and velocity of each particle are updated according to the following two "best" quantities:

- *Pbest*, the best position the particle has visited so far. This represents a local optimum.
- *Gbest*, the best position visited so far by any particle in the population. This represents the current global best solution.

Velocity and position updating is carried out using the following equations:

$$V_{n+1} = wV_n + c1 * rand_1*(Pbest_n - P_n) + c2 * rand_2 * (Gbest_n - P_n) \quad (2.9)$$

$$P_{n+1} = P_n + k \,.\, V_{n+1} \quad (2.10)$$

where:

V_n, (V_{n+1}) is the particle velocity in iteration n, $(n+1)$

P_n, (P_{n+1}) is the particle position (solution) in iteration n, $(n+1)$

$Pbest_n$ and $Gbest_n$ are the "personal" best and "global" best positions in iteration n

$rand_1$ and $rand_2$ are random numbers between 0 and 1

$c1$, $c2$ are weighting factors (each usually a number in the range 0 to 4)

w is the 'inertia' weight. A large value of w facilitates global searching while a small value encourages local searching. Usually, w is allowed to decrease as the optimisation progresses.

The number of particles in a PSO population is usually in the range of 20 to 40 and the weighting factors $c1$and $c2$ typically have the value 2. These factors determine the

maximum jump that a particle can make in one step. Too large a jump can result in oscillation, while too small a value can cause slow convergence making the particle to become trapped in local minima.

Figure 2.6 illustrates the operation of the PSO Algorithm for a simple one-dimensional optimisation problem.

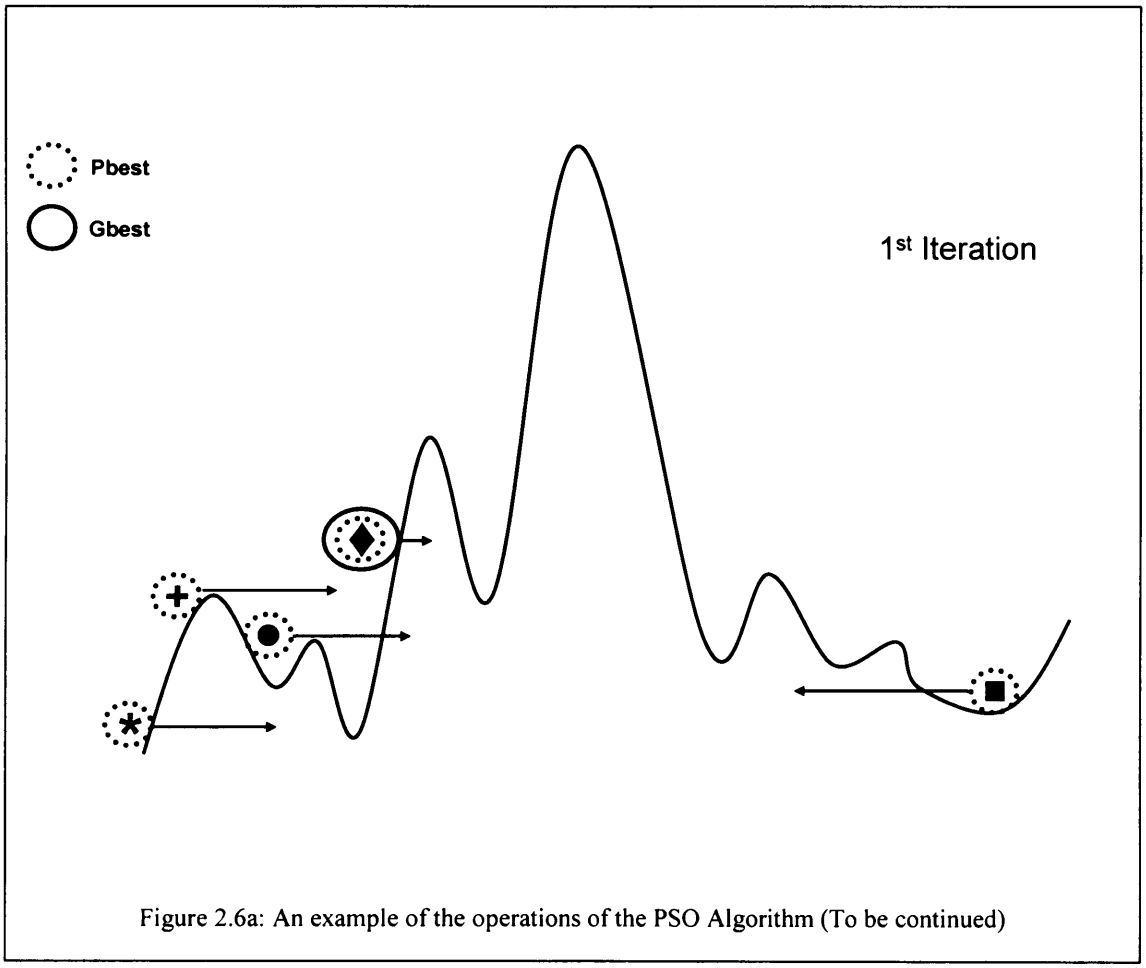


Figure 2.6a: An example of the operations of the PSO Algorithm (To be continued)

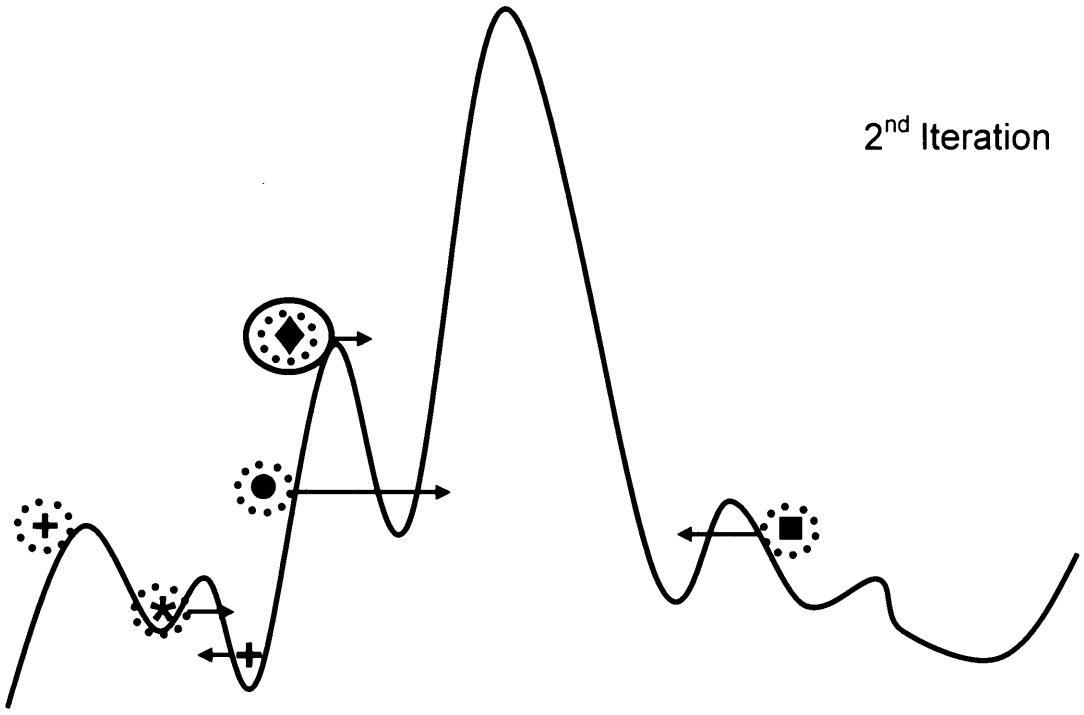


Figure 2.6b: An example of the operations of the PSO Algorithm

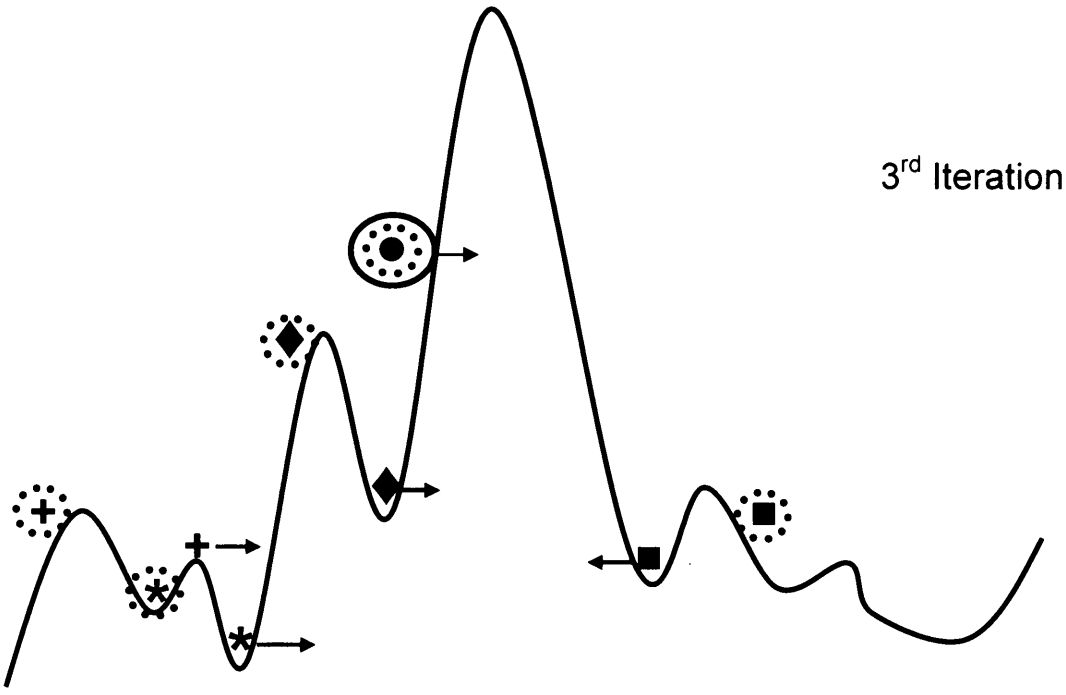


Figure 2.6c: An example of the operations of the PSO Algorithm (To be continued)

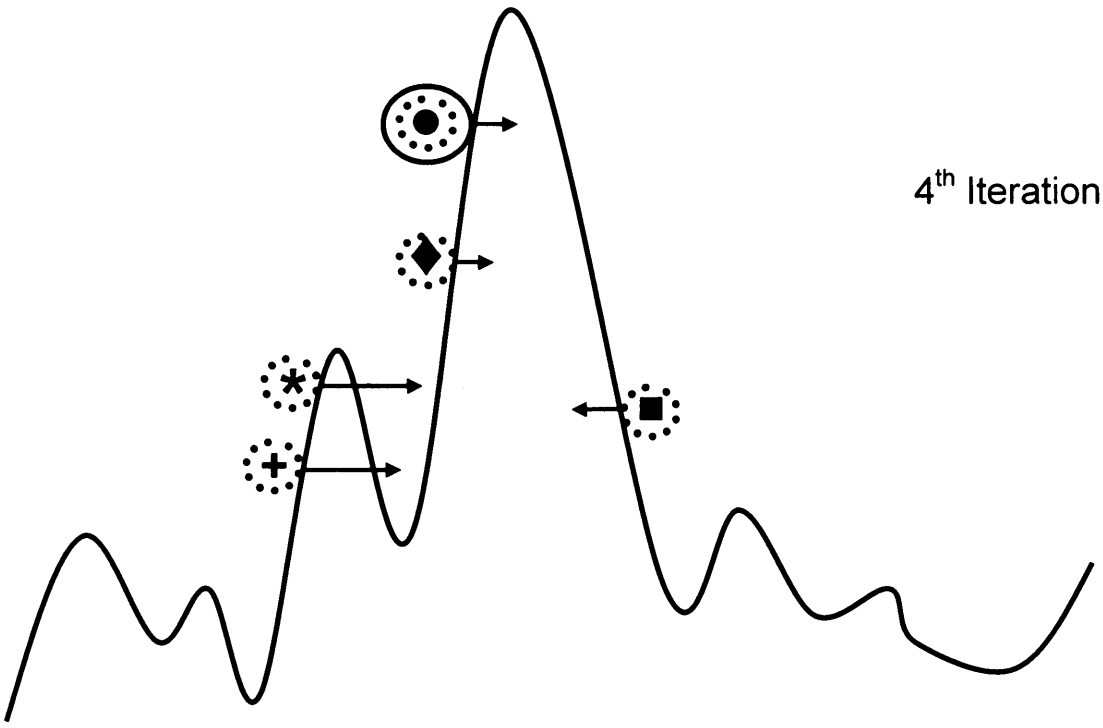


Figure 2.6d: An example of the operations of the PSO Algorithm (Cont'd)

2.3.1 Particle Swarm Optimisation (PSO) vs. Evolutionary Algorithm (EA)

As previously mentioned, Particle Swarm Optimisation Algorithm has its roots in other branches of learning or fields, including artificial life, Evolutionary Algorithm (EA) and the Swarm theory. Though there are similarities between the PSO Algorithm and Evolutionary Algorithms, the differences validate the separation of the PSO and EA. In the following subsections, the similarities and differences between the PSO and EA are presented with reference to the search process, representation of individuals, the fitness function, recombination, mutation, selection and elitism (Engelbrecht 2005).

2.3.1.1 Search Process

The PSO and EA are examples of stochastic, population-based optimisation algorithms and both maintain a population of individuals or candidate solutions. The solutions to optimisation problems are achieved by transforming the current population using a variety of operators. Transformation in the PSO Algorithm is inspired by simplified models of social behaviour of biological organisms while with EA, the transformation is inspired by the neo-Darwinian view of evolution. Both PSO and the EA are inspired by natural occurrence.

The driving force in the PSO Algorithm is social interaction amongst individuals purposely to exchange knowledge about the search space. Individuals, also referred to as particles, apply a direct influence on each other that results in solutions obtained by the search space exploration of all the individuals in the population and not just that of a single individual. The driving force within EA is survival of the fittest, where individuals vie for survival and the production of offspring – generally, individuals perform an independent exploration of the search space. Certain EA have a component where individuals are influenced in some form by other individuals such as in Cultural Algorithms (CA) by (Reynolds 1999; Rychtyckyj and Reynolds 1999) maintaining two spaces – a search space and a belief space. Here in CA, individuals evolve independently from one another in the population space using any EA while selected individuals are allowed to update a belief space. Operators in the population space make use of the knowledge in the belief space to adjust individuals via mutation and cross-over. Whereas, the PSO features direct influence between individuals, the influence within CA is indirect

– all individuals in a CA experience the same influence but the opposite applies in the PSO, where an individual is influenced by different individuals.

In PSO, transformation involves the movement of particles in continuous trajectories (due to velocity and position updates). In EA, transformation can be viewed as discrete changes.

PSO uses the memory of previous good positions and a flight direction to influence new positions. Each particle maintains a memory of its best obtained position, acquired through its own search experience while the previous flight direction is remembered via the momentum or inertia quantity. Conversely, EA that use elitism or "hall of fame" may be thought of as having a memory (though limited to a subset of individuals). The belief space of CA can be considered as a memory, but it serves as a global memory (contributed by only a subset of individuals) for all individuals. There is no individual memory.

Finally, search by the PSO is not driven by a fitness function as is the case for EA; rather search is driven by social interaction among individuals.

2.3.1.2 Representation

In the literature, Evolutionary Algorithms (EA) have been used to solve problems in different domains (continuous-valued and discrete-valued spaces) where individuals are characterised as bit strings, floating-point vectors, tree structures or graphs. The PSO Algorithm on the other hand was developed specifically for continuous-valued spaces; the discrete versions were developed later. As a result of velocity updates that are created

for continuous spaces, the representation scheme is restricted to floating-point vectors using special operators to convert the velocity / position vectors into discrete representations.

The EA have been applied to individuals of variable length in the same population; the PSO can only operate on particles of the same length due to the vector arithmetic operators used to update velocities and positions.

2.3.1.3 Fitness Function

PSO and EA use a fitness function to quantify the optimality of a candidate solution represented by an individual. The search process of EA is driven by the fitness function and information exchange is done from fitness-dependent selected parents to offspring while in PSO the search process is driven by the experience of the individual and that of its neighbours, where the fitness function is used only to quantify the optimality of a solution and also to select the personal best and global best (local best) solutions.

2.3.1.4 Recombination

The concept of survival of the fittest is not implemented in PSO – here, individuals do not compete for survival. The particles persist throughout the search process, and do not die nor do they create offspring. Though new individuals are not created, the PSO does have an implicit form of recombination through the stochastic combination of the cognitive and social components in the velocity update equation – each particle is accelerated stochastically towards a weighted average of its personal best and global best (or local best) position by the use of a cross-over operator.

2.3.1.5 Mutation

The reason or rationale for mutation in EA is to introduce new genetic materials into the population in order to increase diversity. Mutation facilitates balance between exploration and exploitation, achieved by adjusting the variance of the noise injected into an individual. The directional PSO position updates replicate mutation in EA with a kind of built-in memory. Stochastic variation in PSO is accomplished via the random vectors *r1* and *r2*, and the magnitude of the 'mutation' is determined by the past velocity, the cognitive and social components. Velocity clamping, a constriction factor and inertia weight control and balance exploration and exploitation.

2.3.1.6 Selection

The process of selection is not explicitly present in the PSO, though the attraction of particles toward the global best position bears a resemblance to the effect of selection to some extent (PSO has an implicit, weak selection mechanism). In EA, the purpose of selection is to force unfit individuals to die, while ensuring that fit individuals survive and reproduce (offspring replacing parents). Through the use of an elitism operator, some parents survive to the next generation.

In PSO all the individuals survive, unfit individuals do not die and there is no competition for survival (individuals cooperate to achieve the common goal to find an optimum solution). Though the unfit individuals do not die, they are seen to 'surrender' to the more fit individuals. The PSO moves towards the fit individuals.

Again in EAs, selection / elitism / hall-of-fame are mainly used to select parents for recombination to produce offspring. Elitism is naturally used in PSO through the

selection of the global best (or local best) position and elitism purposely ensures that only the best individual influences the direction of the search, with particles moving towards the global best (local best) position. Furthermore, the cognitive component of the velocity update equation of the PSO looks a lot like the hall-of-fame selection where the best position of the trajectory of each particle is remembered and later used to influence the new search direction.

Finally, PSO controls the rate of convergence through the use of acceleration coefficients and inertia weights. In EA, the rate of convergence is controlled through the use of selection pressure.

Some of the modifications to the PSO Algorithm since its development in 1995 are described in Appendix G. These modifications resulted in variants of the algorithm that were proposed to incorporate the aptitude and capabilities of other evolutionary computation methods, such as hybrid versions of the PSO or the adaptation of the PSO parameters for a better performance (adaptive PSO). In other variations of the PSO Algorithm, the nature of the problem to be solved necessitate the PSO to work under complex environments as in the case of the multi-objective, constrained optimisation problems and tracking dynamic systems. Also included are other variants to the original formulation incorporated to improve the performance of the algorithm, such as the stretching and passive congregation techniques to prevent the particles from being trapped in local minima.

Other notable optimisation algorithms excluded from the thesis include the Genetic Algorithm 'GA' (Luger 2002; Oei *et al.* 1991; Ronald 1995, 1996), the Learning System Classifier 'LCS' (Davis and King 1977; Holland and Burks 1985; Holland and Reitman 1977; Moriarty *et al.* 1999; Smith 1992), Hill Climbing 'HC' (Russell and Norvig 2002), Random Optimisation 'RO' (Gurin and Rastrigin 1965; Matyas 1965; Rastrigin 1963; Schumer 1965; Schumer and Steiglitz 1968), Simulated Annealing 'SA' (Černý 1985; Kirkpatrick *et al.* 1983; Metropolis *et al.* 1953), Downhill Simplex 'DS' (Lagraias *et al.* 1998; Lewis *et al.* 2000; McKinnon 1999; Nelder and Mead 1965; Olsson and Nelson 1975), Tabu Search 'TS' (Glover 1986, 1989, 1990; Hansen 1986; Werra and Hertz 1989), Memetic Algorithm 'MA' (Digalakis and Margaritis 2004; Krasnogor and Smith 2005; Moscato 1989), Differential Evolution 'DE' (Besson *et al.* 2006; Mendes and Mohais 2005; Mezura-Montes *et al.* 2006; Storn and Price 1995) and the Ant Colony Optimisation Algorithm 'ACO' (Deneubourg and Goss 1989; Deneubourg *et al.* 1983; Dorigo and Blum 2005; Dorigo *et al.* 1998; Dorigo *et al.* 1996; Goss *et al.* 1990; Grassé 1959; Korošec and Šilc 2006; Manderick and Moyson 1988; Schoonderwoerd *et al.* 1996; Stickland *et al.* 1992; Théraulaz and Bonabeau 1995; Werfel and Nagpal 2006).

2.4 Summary

This chapter has given detailed background information on Optimisation with attention focused on optimisation problem classification, optimality conditions and causes of problems affecting the performance of optimisation algorithms in general. The origin of the Bees Algorithm and the PSO Algorithm are discussed. Finally, a comparison between the PSO Algorithm and Evolutionary Computation in relation to the search process,

fitness function, representation of individuals, recombination, mutation, selection and elitism concludes the chapter.

Chapter 3: Particle Swarm Optimisation – Bees Algorithm

For courage mounteth with occasion.
--William Shakespeare, "King John"

This chapter presents the Particle Swarm Optimisation - Bees Algorithm (PSO-Bees Algorithm), a modification to the Particle Swarm Optimisation Algorithm that incorporates adaptive neighbourhood and global random search around the global best particle with two main advantages. Firstly, the PSO-Bees Algorithm is more robust and exhaustively searches the problem space converging onto the global optimum. Secondly, it solves the problem of premature convergence of the PSO Algorithm that limits its ability to find the global optimum of objective functions. Thus, the PSO-Bees Algorithm combines the fast convergence property of the PSO Algorithm and the inherent ability of the Bees Algorithm to avoid been trapped in local optima.

3.1 PSO-Bees Algorithm

The Particle Swarm Optimisation (PSO) Algorithm is an optimisation algorithm that shows promise. However its performance on complex problems with multiple minima falls short when compared with other optimisation algorithms. The basic PSO Algorithm has been applied successfully to a number of problems including standard function optimisation problems (Angeline 1998; Kennedy and Eberhart 1997; Kennedy and Spears 1998), solving permutation problems (Salerno 1997) and training multi-layer neural networks (Eberhart and Kennedy 1995; Kennedy and Eberhart 1995b, 1997). Though the empirical results presented illustrated the ability of the PSO Algorithm to solve optimisation problems, the results also confirmed that the basic PSO Algorithm has

problem of premature convergence that prevents the algorithm from consistently converging to globally optimal solutions. As a result, a number of modifications (see Appendix G) to the basic PSO Algorithm are introduced to improve the speed of convergence and the quality of the solutions found.

The PSO-Bees Algorithm was developed to solve the problem of premature convergence known to exist in the different versions of the PSO algorithm, including the inertia weight and constriction version.

Focusing on the issue of premature convergence, the different versions of the PSO models highlighted in the previous chapter all have a dangerous property: when P_n = $Pbest_n$ = $Gbest$, the velocity update equation will depend only on the value of wV_n i.e. momentum. This implies that if the current position of a particle coincides with the global best position / particle, the particle will only move away from this point if its previous velocity V_n and inertia weight 'w' are non-zero. On the other hand, if the previous velocities are close to zero, all the particles will stop moving once they catch up with the global best particle, which leads to premature convergence of the algorithm. However, this does not necessarily mean that the algorithm has converged unto the local minimum or even a global optimum but instead it implies that all the particles have converged on the best position discovered so far by the swarm.

In the PSO Algorithm, a trajectory that converges is seen as a form of termination criterion, but this does not help to determine whether the algorithm has converged onto a global or local optimum or minimum. The PSO-Bees Algorithm alleviates the problem of

premature convergence by incorporating features of adaptive neighbourhood and global random search into the PSO Algorithm. This resulted in increased ability to escape from stagnant states to reach the global optimum or minimum of objective functions.

With the PSO-Bees Algorithm, each individual is a point (or particle) in search space representing a candidate solution to the optimisation problem being addressed. The algorithm drives towards the optimal solution by controlling the movements of individual candidate solutions.

Unlike the conventional PSO Algorithm, three sets of particles make up the entire swarm in the PSO-Bees Algorithm, namely:

- regular particles
- neighbourhood particles
- random particles

The regular particles search the problem space as in the basic PSO Algorithm. The neighbourhood particles search the neighbourhoods of promising selected candidates including the global best particle previously found by the regular particles. The initial size of the neighbourhoods is kept unchanged provided that the neighbourhood particles are able to find better solutions in the neighbourhoods. If the neighbourhood search does not yield any progress, the size is reduced to make the local search more exploitative, searching more densely the areas around the most promising particles. If there is no improvement from reducing the size of the neighbourhoods, it is assumed that the global best particle has been found. Finally, global random search of the problem space is done

by the random particles in a similar fashion to the original Bees Algorithm (Pham *et al.* 2006a).

Figure 3.1 presents the pseudo code for the PSO-Bees Algorithm in its simplest form.

Initialise PSO-Bees population
Repeat:
for each particle n:
  Update particle position using equations (3.1) & (3.2)
  if $f(P_n) > f(Pbest_n)$
    then $Pbest_n = P_n$
  if $f(Pbest_n) > f(Gbest)$
    then $Gbest = Pbest_n$
//Bees Algorithm Section
//Adaptive Neighbourhood Search
Generate neighbourhood particles *neigh*
for each particle *neigh*:
  if $f(P_{neigh}) > f(P_{Selected\ Candidate})$
    then $P_{Selected\ Candidate} = P_{neigh}$
  Subsume P_{neigh} into regular population with initial velocity of replaced candidate
endfor
//Global Random Search
Generate random particles *rand*
for each particle *rand*:
  if $f(P_{rand}) > f(P_{Selected\ Candidate})$
    then $P_{Selected\ Candidate} = P_{rand}$
  Subsume P_{rand} into regular population with average velocity of swarm
endfor
reset each *rand*
endfor
Until stopping condition is true

Figure 3.1: Pseudo code of the PSO-Bees Algorithm

As mentioned earlier, the PSO-Bees Algorithm performs adaptive neighbourhood searches. This is done by sending neighbourhood particles one at a time around the selected candidates to conduct adaptive neighbourhood searches. If a neighbourhood

particle has a better fitness compared to the promising selected candidate it becomes selected. The same is done with the global random search.

In every iteration, the position and velocity of each particle are updated according to the following two best quantities:

- *Pbest*, the best position the particle has visited so far. This represents a local optimum.
- *Gbest*, the best position visited so far by any particle in the population. This represents the current global best solution.

The velocity update (2.9) and position update (2.10) equations are reproduced below:

$$V_{n+1} = wV_n + c_1 * rand_1 * (Pbest_n - P_n) + c_2 * rand_2 * (Gbest - P_n) \qquad (3.1)$$

$$P_{n+1} = P_n + k\,V_{n+1} \qquad (3.2)$$

where:

V_n (V_{n+1}) is the particle velocity in iteration n (n+1)

P_n (P_{n+1}) is the particle position (solution) in iteration n (n+1)

$Pbest_n$ is the "personal" best position for particle n

Gbest is the "global" best position

k is the time step

$rand_1$, $rand_2$ are random numbers between 0 and 1

c_1, c_2 are weighting factors (each usually a number in the range 0 to 4)

w is the 'inertia' weight. A large value of w facilitates global searching while a small value encourages local searching. Usually, w is allowed to decrease as the optimisation progresses.

The factors c_1 and c_2 determine the maximum jump that a particle makes in one step. Too large a jump can result in oscillation, while too small a displacement can cause slow convergence or even trapping of particles at local minima.

The number of particles in a PSO-Bees Algorithm population is usually in the range 20 to 40 while the weighting factors c1 and c2 typically have the value 2.

3.2 Operations of PSO-Bees Algorithm

As mentioned earlier, the PSO-Bees Algorithm incorporates features of adaptive neighbourhood and global random search into the PSO Algorithm, thus having the ability to escape from stagnant states in order to reach the global optimum or minimum of the objective function. Figure 3.2 illustrates the operations of the PSO-Bees Algorithm for a simple one-dimensional optimisation problem.

It is important to understand the significance of having adaptive neighbourhood and global random particles in the PSO-Bees Algorithm. This increased ability to locate the global optimum comes from the neighbourhood particles searching around the promising selected candidates and global random search with a small number of iterations. As the number of points in the search space grows exponentially with the number of dimensions, the PSO-Bees Algorithm is still rigorously competitive in finding the optimal solution. It

takes slightly longer time to find the global optimum of functions with higher dimensions.

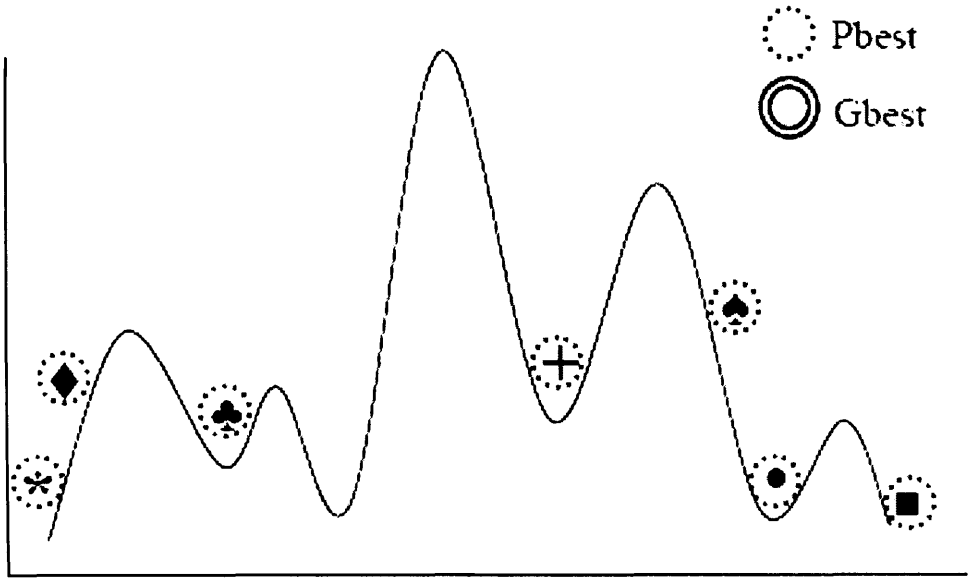


Fig 3.2a: Randomly initialise regular particles

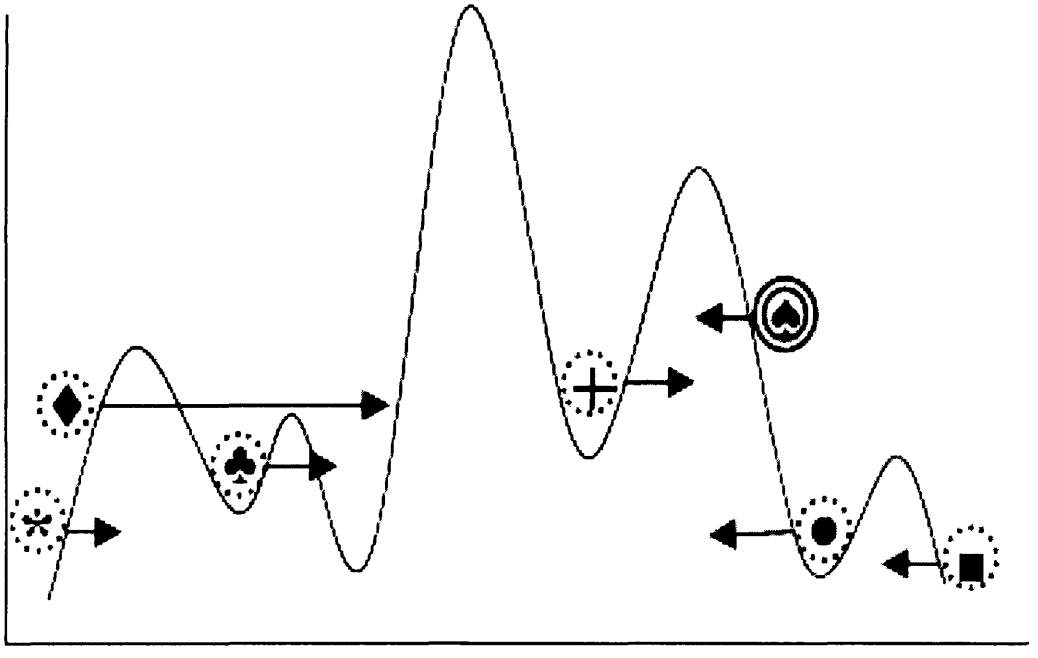


Fig 3.2b: Evaluate fitness of the population

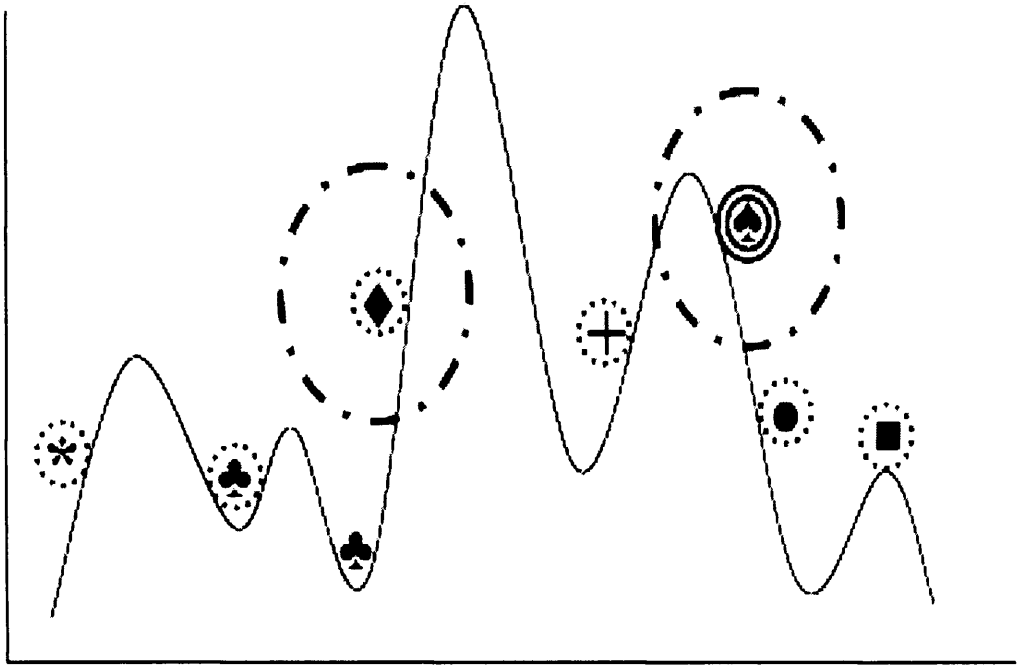


Fig 3.2c: Select candidates for adaptive neighbourhood search

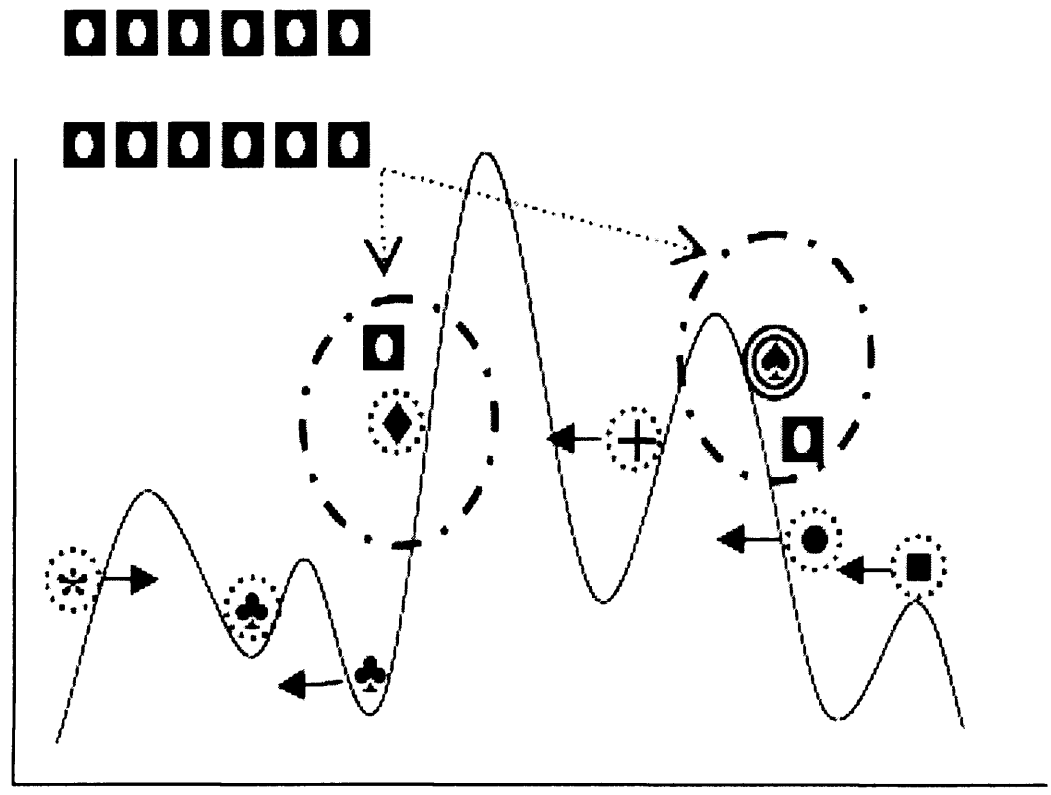


Fig 3.2d: Enlist neighbourhood particles to search around the promising selected candidates

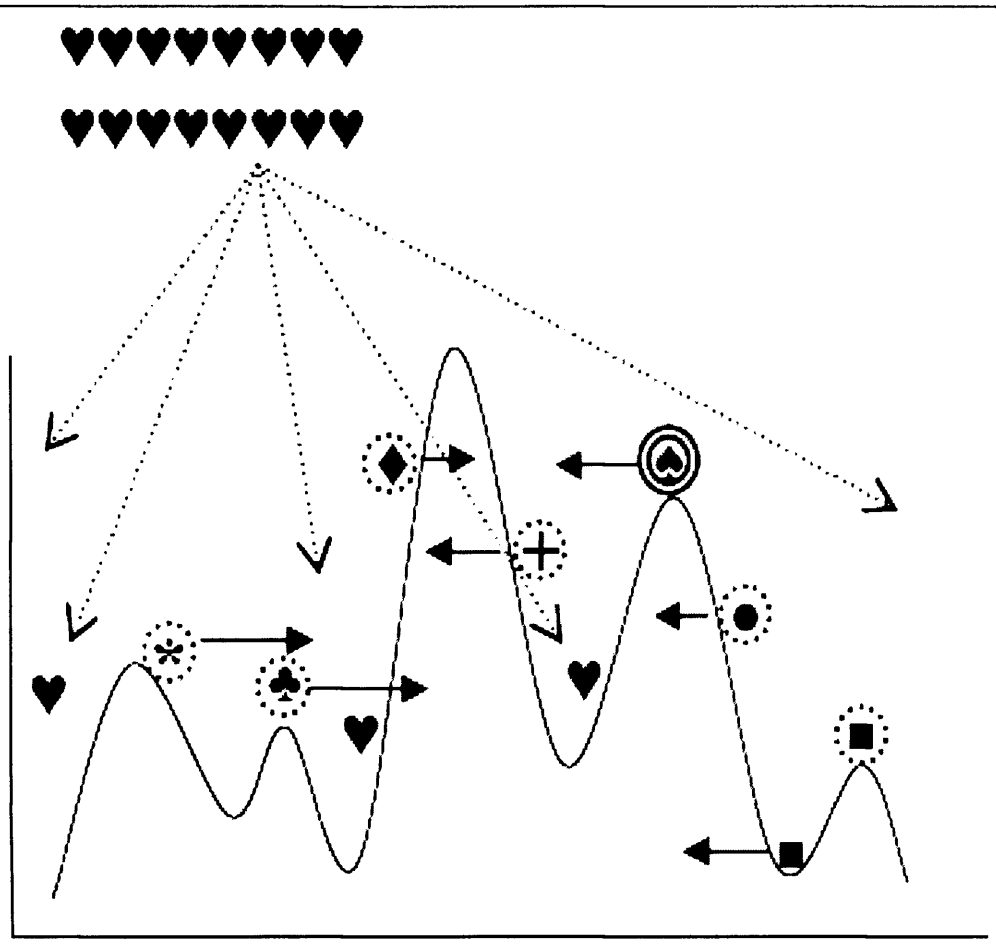


Fig 3.2e: Enlist random particles to search problem space

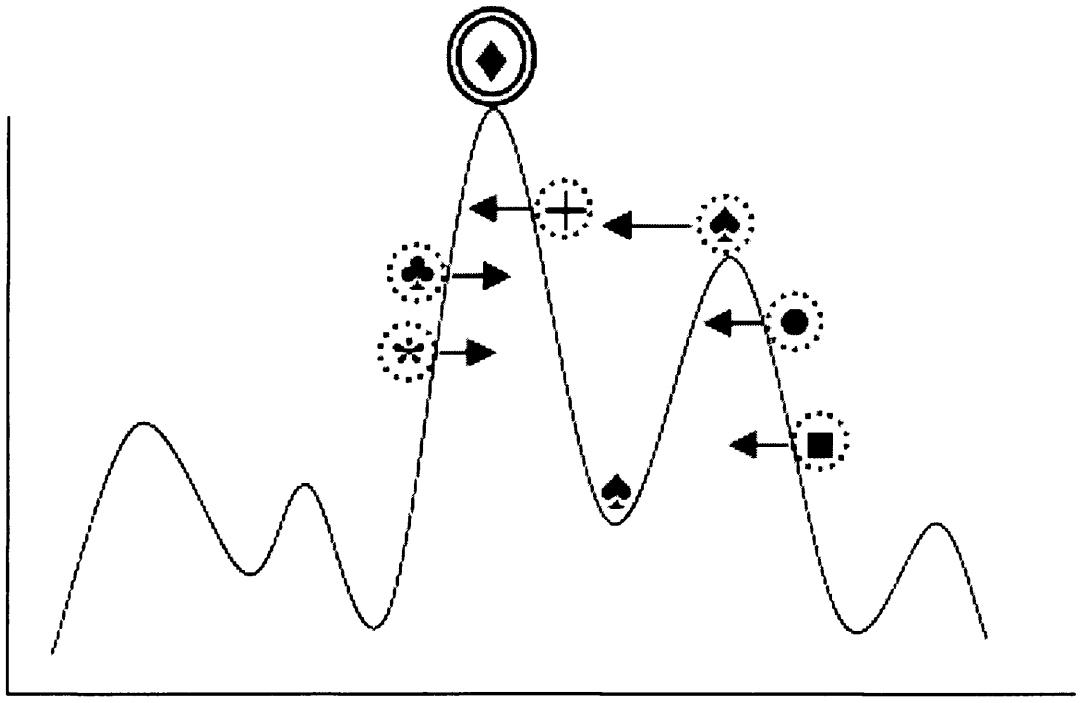


Fig 3.2f: Global optimum solution found

3.3 PSO-Bees Algorithm Parameters

Akin to the basic PSO Algorithm parameters, the PSO-Bees Algorithm is influenced by a number of control parameters. They include: the dimension of the problem, swarm size, acceleration coefficients, velocity clamping, inertia weight, neighbourhood size, constriction coefficient, number of iterations and the random values scaling the contributions of the cognitive and social components of the velocity update equation (3.1). These factors (Engelbrecht 2005) influencing the performance of the PSO-Bees Algorithm are discussed in the following subsections.

3.3.1 Velocity Clamping (VC)

In the earlier applications of the basic PSO Algorithm it was found that the velocity quickly reached large values for particles far from the neighbourhood best and personal best positions. As a consequence, particles with large position updates leave (explode) the boundaries of the search space and particles diverge. In order to control the global exploration of particles, the particles' velocities are clamped purposely to stay within the boundary constraints (Eberhart *et al.* 1996). When a particles' velocity exceeds a specified maximum velocity (V_{max}), the particles' velocity is reset to V_{max}. As a result, the maximum velocity controls the granularity of the search by clamping escalating velocities. The V_{max} is responsible for balancing the contradictory objectives of exploration and exploitation. Large values of V_{max} facilitate global exploration while smaller values encourage local exploitation. In a situation where the V_{max} is too small, the swarm will not explore sufficiently beyond locally good regions. Also, there is an increase in the number of time steps needed to reach an optimum and the swarm may

become trapped in a local optimum with no means of escape. Conversely, too large a value of V_{max} risks the possibility of missing good regions of the search space. The particles will most likely jump over good solutions and continue to search in fruitless regions of the problem space, mainly because the particles travel at very high speed. Velocity clamping does not confine the positions of particles but only the step size that is obtained from the particle velocity.

The above leaves the problem of finding an appropriate value for each V_{max} in order to have a balance between:

- Moving too fast or too slow
- Exploitation and exploration

Engelbrecht (Engelbrecht 2005) suggested the V_{max} value is selected to be a fraction of the domain (for each dimension if multidimensional search space). That is,

$$V_{max} = \delta (x_{max} - x_{min}) \qquad (3.3)$$

Where x_{max} and x_{min} are respectively the maximum and minimum values of the domain of x, and δ is problem dependent (Omran *et al.* 2002; Shi and Eberhart 1998a).

The velocity update equation of the PSO Algorithm is responsible for balancing the contradictory objectives of exploration and exploitation. The exploration-exploitation trade-off is crucially important in optimisation as it determines the efficiency and accuracy of any optimisation algorithm. *'Exploration'* is the ability of a search algorithm to explore different regions of the search space in order to locate a good optimum; conversely, *'exploitation'* is the ability to concentrate the search around a promising area in order to refine a candidate solution.

3.3.2 Inertial Weight (IW)

IW was introduced by Shi and Eberhart (Shi and Eberhart 1998a) as an apparatus to control the exploration and exploitation aptitude of the swarm in addition to eliminating the need to clamp the velocity (Eberhart and Shi 2001). The inertia weight *'w'* was found to be successful in achieving the first objective, but does not entirely eliminate the need for velocity clamping. The inertia weight *'w'* is responsible for controlling how much memory of the previous flight direction will influence the new velocity.

Initial implementations of the inertia weight used a static value for the entire search duration while later implementations made use of dynamically changing inertia weight through increasing inertia (Zheng *et al.* 2003), random adjustments (Engelbrecht 2005), linear decreasing (Ratnaweera *et al.* 2003; Suganthan 1999), non-linear decreasing (Schutte and Groenwold 2003) and fuzzy adaptive inertia (Shi and Eberhart 2001). Known approaches from the literature start with large inertia values that decrease over time to smaller values as the iteration progresses. As a result, particles are allowed to explore in the initial search steps while favouring exploitation as time increases.

The value of w is critical to ensure convergent behaviour and also to have equilibrium or a balance between exploration and exploitation. With $w \geq 1$, particle velocity increases over time towards the maximum velocity (if velocity clamping is implemented) making the swarm diverge because particles fail to change direction to move back towards promising areas. On the other hand, when $w < 1$, particle acceleration decreases until their velocities reach zero. This is dependent on the values of the acceleration coefficients. A large value of *'w'* facilitates exploration with increased diversity while a small '*w*'

promotes local exploitation and removes the exploration ability of the swarm. The smaller the *'w'*, the more will the cognitive and social components control velocity update.

Similar to the maximum velocity (V_{max}), the optimal value for the inertia weight is also problem dependent (Shi and Eberhart 1998b).

3.3.3 Constriction Coefficient (CC)

Recent work by Clerc (Clerc 1999) presented an approach akin to the inertia weight to balance the exploration-exploitation trade-off in which the particle velocity is constricted by a constant χ, known as the constriction coefficient. This model presents a method of choosing the values of w, c_1 and c_2 to ensure convergence to a stable point.

A modified velocity update equation using the constriction factor χ, is given below:

$$V_{n+1} = \chi\,(V_n + c_1 * rand_1 * (Pbest_n - P_n) + c_2 * rand_2 * (Gbest - P_n)) \qquad (3.4)$$

where

$$\chi = \frac{2}{|2 - \varphi - \sqrt{(\varphi^2 - 4\varphi)}|} \qquad (3.5)$$

and $\varphi = c_1 + c_2$, where $\varphi > 4$

Example

Let $c_1 = c_2 = 2.05$

Substituting $\varphi = c_1 + c_2 = 4.1$ into (3.5), yields $\chi = 0.7298$ and substituting this into equation (3.4) gives:

Equation 3.4 now becomes:

$$V_{n+1} = 0.7298\ (V_n + 2.05 * rand_1*(Pbest_n - P_n) + 2.05 * rand_2 * (Gbest_1 - P_n)) \quad (3.6)$$

Since 2.05 * 0.7298 = 1.4962, this is equivalent to using the values of $c_1 = c_2 = 1.4962$ and $w = 0.7298$ in the PSO with inertia weight in (3.1). Hence,

$$V_{n+1} = 0.7298 * V_n + 1.4962 * rand_1*(Pbest_n - P_n) + 1.4962 * rand_2 * (Gbest_1 - P_n) \quad (3.7)$$

So we have:

PSO with Constriction factor (3.6)

PSO with inertia weight (3.7)

According to Clerc, the constriction PSO produced good results with the Rastrigin function and other unimodal problems. The opposite is true for problems with many local minima including the Griewangk function and the non stationary or dynamic problems. This is confirmed in the paper by Carlisle and Dozier (Carlisle and Dozier 2000; Carlisle and Dozier 2002).

The constriction approach is effectively equivalent to the inertial weight approach. Both approaches have the objective of balancing exploration and exploitation, thus improving the convergence and the quality of the solution found. Low values of w and χ result in exploitation with little exploration, while large values result in exploration with difficulties in refining the solution. However, Engelbrecht (Engelbrecht 2005) highlighted the differences in the two approaches:

- Firstly, velocity clamping is not necessary for the constriction model.

- Secondly, the constriction model guarantees convergence under the given constraint (no information on the quality of the point converged to) and
- Thirdly, the change in direction of particles is done via constant φ.

Eberhart and Shi (Eberhart and Shi 2000) compared the performance of a swarm using velocity clamping and the constriction factor. Their results showed that using the constriction factor (without clamping the velocity) usually resulted in a better rate of convergence. However, on some test functions, the PSO Algorithm with constriction failed to reach the specified error threshold for that problem within the allocated number of iterations. (Eberhart and Shi 2000) discovered the problem was caused by particles straying too far from the desired search space. They were able to show empirically that when velocity clamping and constriction factor are used together, it results in faster convergence rates.

3.3.4 Swarm size (SZ)

The swarm size is affected by the initial scheme employed in the initialisation process. Provided a good uniform initialisation scheme is used for the particles, the more the particles in the swarm, the larger would be the initial diversity because a large swarm allows larger parts of the search space to be covered in each iteration. It has the demerit of increasing the computational complexity and eventually degrading to a parallel random search. However, on the other hand, it has the merit of needing fewer numbers of iterations to reach a good solution compared to smaller swarms. (Bergh and Engelbrecht 2001) showed that the PSO Algorithm has the ability to find optimal solutions with small

swarm sizes of 10 to 30 particles. (Brits *et al.* 2002) indicated success with fewer than 10 particles. Again worth mentioning, the optimal swarm size is problem dependent. A smooth search space will need fewer particles compared with a rough surface to locate optimal solutions. Engelbrecht (Engelbrecht 2005) suggested the swarm size be optimised for each problem using cross-validation.

3.3.5 Neighbourhood size (NS)

The neighbourhood size determines to what degree the level of social interaction that takes place in the swarm. The smaller the neighbourhood, the less interaction occurs and vice versa. Even as smaller neighbourhoods are very much slower in converging, it is better and more reliable in relation to converging to the global optimum. The work of (Suganthan 1999) took advantage of the merit of small and large neighbourhoods. He initially started the search with small neighbourhoods and later increasing the neighbourhood size proportionally with the corresponding increase in the number of iterations which ensures a high diversity with faster convergence because the particles moved towards a promising search area.

3.3.6 Number of Iterations

The number of iterations that is necessary for the PSO Algorithm to reach the global optimum is problem dependent. Too small a number of iterations will most likely terminate the search procedure too hastily and prematurely meaning that the algorithm has little time to exhaustively search the problem space. On the other hand, a too large number of iterations results in additional unnecessary computational complexity

(especially when the number of iterations is the only stopping criteria of the search process).

3.3.7 Acceleration Coefficient (AC)

Acceleration coefficients, c_1 and c_2 together with $rand_1$ and $rand_2$ control the stochastic influence of the cognitive and social components on the particle velocity. The PSO Algorithm velocity update equation (3.1) made use of two independent random sequences, $rand_1$ and $rand_2$ to direct or control the stochastic nature of the algorithm with their values scaled by the constants $0 < c_1, c_2 \leq 2$. The acceleration coefficients (c_1 and c_2) influence the maximum size of the step that a particle can undertake or move in a single iteration.

From (3.1), c_2 regulates the maximum step size in the direction of the global best particle while c_1 regulates the step size in the direction of the personal best position of that particle. As mentioned earlier, these factors (c_1 & c_2) determine the maximum jump that a particle can make in one step or iteration. Too large a jump can result in oscillation, while too small a displacement can cause slow convergence or even trapping of particles at local minima.

3.4 PSO / Hybrid PSO Stopping Criteria

The goal of optimisation algorithms is simple and clear: the global optimum should be found. Nevertheless, in general it is not clear when this goal is achieved, especially if real-world problems are optimised for which no knowledge about the global optimum is available. In reality, it is difficult and cumbersome to come to a decision when the

execution of any optimisation algorithm should be terminated. The only sole exception is when the value of the objective function in the global minimiser for instance is known in advance.

The interesting dilemma is when does one stop the algorithm and decide if the found stable state or solution is the optimal while taking into serious consideration the fact that the probability of sampling the optimality region decreases significantly as the number of dimensions of the problem space increases. Solis and Wets (Solis and Wets 1981) suggested some guidelines in choosing the correct number of iterations for stochastic search algorithms specifically to locate a global minimum.

The PSO-Bees Algorithm stopping criteria helps, firstly to determine when the algorithm has converged to a stable state and secondly to terminate execution of the algorithm and return the best particle from the swarm. In most cases with other algorithm, the execution is terminated after a specified number of iterations, at which point the best solution is considered or assumed found but unfortunately there is no assurance or guarantee that the solution found is the global optimum.

In contrast to using the maximum number of iteration / function evaluations as stopping criteria, other stopping criteria suggested by Zielinski and Laur (Zielinski and Laur 2007) reproduced below for convenience have the advantage of reacting adaptively to the state of the optimisation runs. These include: *improvement-based criteria*, *movement-based criteria*, *distribution based criteria* and *combination of conditions* or *criteria*. In all four,

instead of using the particle position (P_n) for the calculation of stopping criterion, the personal best positions (*Pbest*) are used. These are elaborated below:

Improvement-based criteria terminates the run if a small improvement is identified for the reason that in the beginning of an optimisation run, large improvements are achieved while in later stages, the improvement becomes small. There are three variants:

ImpBest: Improvement to the best objective function is monitored. If it falls below a given threshold *'thres'* for a number of generations *'gene'*, the run is terminated.

ImpAve: similar to *ImpBest*, but instead of monitoring the best objective function value, the average value calculated from the whole population is checked.

NoAcc: observed if any new P_{best} is accepted in a specified number of iterations.

In ***movement-based criteria***, the *movement* of individual particle is monitored and not the *improvement* to the *Pbest*. Two conditions apply:

MovObj: The movement of the individuals with respect to their objective function value (objective space) is examined if it is below a threshold *'thres'* for a number of generations *'gene'*. *MovObj* is different from *ImpAve* if the algorithm allows deterioration of the individuals' objective function value.

MovPar: The movement with respect to positions (parameter space) is checked if it is below a threshold *'thres'* for a number of generations *'gene'*.

Distribution-based criteria take into account the diversity in the population. When the diversity is low for instance, the individuals are close to each other, and there is the assumption that there is convergence. There are four main variants:

StdDev: Monitored if the standard deviation of positions is below a given threshold *'thres'*.

MaxDist: Monitoring the distance from every member of the swarm or population to the best individual or particle. The optimisation is terminated when the maximum distance is below a specified threshold *'thres'*.

MaxDistQuick: A generalisation of *MaxDist,* instead of using the whole population for the computation of the maximum distance to the best population member. A quick-sort algorithm is used for sorting the particles based on their objective function value and a percentage of the P_{best} is taken into account.

Diff: The difference between the best and the worst objective function value is monitored if it is below a threshold *'thres'*. In addition, at least a percentage of the P_{best} is also taken into account because *Diff* could lead to undesired results when, for example, only two particles are feasible but incidentally are close to each other. In contrast to the previous three criteria that are used in parameter space, *Diff* considers objective space.

Combined criteria: It is often beneficial to combine several criteria because functions have different features.

ComCrit: This is a combination of *ImpAve* and *MaxDist*. Only if the condition of *ImpAve* is true is *MaxDist* checked.

Diff_MaxDistQuick: Diff is an easily checked criterion but fails with flat surfaces. If this condition is true, then *MaxDistQuick* is checked.

3.5 Performance Measures (PM)

This section identifies PSO-Bees Algorithm's performance measures. These measures (Engelbrecht 2005) assess performance on six fronts: accuracy, reliability, robustness, efficiency, diversity and coherence. They represent a useful tool for checking the effectiveness and efficiency of optimisation algorithms.

3.5.1 Accuracy

The global best (G_{best}) is used as a yardstick for representing the accuracy and quality of the solution found. In a situation when prior knowledge of the optimum solution is known, the accuracy is expressed as the error of the G_{best} position.

$$\text{Accuracy} = |f(Gbest(t)) - f(x^*)| \tag{3.8}$$

Where x* is the theoretical optimum.

Conversely, if there is no information on the theoretical optimum, the accuracy at time step t is expressed as the fitness of the global best particle.

$$\text{Accuracy} = f(\text{Gbest}(t)) \tag{3.9}$$

Also, the accuracy can be obtained by approximating the derivative of the fitness function at the position of the global best particle at time *t*. At an optimum, the derivative of the fitness function is zero – the smaller the derivative of the global best position, the better the solution and vice versa. If the derivative of the global best position is zero, the global best can represent either a local or global optimum because the derivative of both local and global optima is zero.

On comparison with other optimisation algorithms, the accuracy of the solution found by the swarm is determined in relation to the number of function evaluations as an alternative to the number of iterations.

3.5.2 Reliability

Evaluating the performance of algorithms with random initial conditions is achieved over a large number of simulation runs; reliability in this case refers to the percentage of simulations that reached or coincide with a specified accuracy (fitness value or error). The more the simulation runs converge to the specified accuracy, the larger the accuracy of the algorithm, which is a good indication on the reliability of the swarm.

3.5.3 Robustness

A typical PSO Algorithm swarm is more robust or stable when the variance of a performance criterion over a number of simulation runs is smaller. Engelbrecht (Engelbrecht 2005) showed robustness of a swarm to be in the range:

$$\text{Robustness}(S(t)) = [\bar{\theta} - \sigma_\theta, \bar{\theta} + \sigma_\theta] \tag{3.10}$$

Where $\bar{\theta}$ is the average of the performance criterion over a number of simulation runs, and σ_θ is the variance in the performance criterion. The smaller the value of σ_θ the smaller the range performance values unto which the simulations converge – the more stable the swarm.

3.5.4 Efficiency

The efficiency of the swarm is usually expressed as the number of iterations or the number of function evaluations in order to find a solution with reference to a specified accuracy. Swarm efficiency expresses the relative time to reach a desired solution.

3.5.5 Diversity

Diversity is important, especially with population-based optimisation algorithms and has a close correlation with the global convergence of the PSO-Bees Algorithm. A large diversity directly implies that a large area of the search space needs to be explored which again defines the degree of dispersion of the swarm individuals. The equation of diversity by Vesterstrom *et al* (Vesterstrom *et al.* 2002) gave an indication on the range of the search space covered by the swarm but no indication on the quantification of the dispersion of the swarm particles.

Having a probabilistic divergent behaviour of the swarm can have a positive influence on the diversity of the solutions examined by the particles, thereby improving its exploration capabilities. This property is especially valuable when optimising functions having many local minima.

3.5.6 Coherence

Each particle within the swarm has a unique position to which it is attracted provided the swarm is properly initialised. The particles continue to search the problem space under the sway and control of the entire swarm performance and respective prior history. The information of the swarm movement or travel shapes the spread of particles within the swarm. When the swarm is centred or concentrated upon a solution, the particles move with less velocity from each other and the swarm converges. On the other hand, if the swarm moves or travels as a structured entity, all the particles will have a common velocity vector. Hence, there is need to stretch or widen the solution space searched by the swarm. This is achieved using a coherence velocity term.

Hendtlass and Randall (Hendtlass and Randall 2001) define swarm coherence as:

$$\text{coherence}(S(t)) = \frac{e_s(t)}{\bar{e}(t)} \tag{3.11}$$

where the speed of the swarm centre '$e_s(t)$' at time 't' is defined as

$$e_s(t) = \frac{\left| \sum_{i=1}^{n_s} V_i(t) \right|}{\left| n_s \right|} \tag{3.12}$$

and the average particle speed $\bar{e}(t)$ is given below as:

$$\bar{e}(t) = \sum_{i=1}^{s} \frac{|V_i(t)|}{s} \tag{3.13}$$

where 's' is the number of particles.

The following section describes the results obtained using the PSO-Bees Algorithm to train an MLP Network.

3.6 Results

This section presents the results of two different applications of the PSO-Bees Algorithm. The algorithm is applied to train feed forward Neural Networks (NN) to solve pattern recognition and classification problems, specifically Control Chart Pattern Recognition (CCPR) and the Wood Defect Classification (WDC) respectively. First, the section starts with an introduction to NN, why NN was chosen. The advantages and limitations of NN are also highlighted. This is then followed by an introduction to CCPR with the results obtained. Then an introduction to WDC and the results are presented. Finally, the presentation of the results obtained from tests on well-known mathematical benchmark functions concludes this chapter.

3.6.1 Neural Network Training

Introduction

An Artificial Neural Network (ANN), also called Neural Network (NN) is a mathematical or computational model based on the biological neural networks. The original inspiration for the technique was from the examination of the central nervous system, neurons, axons, dendrites and synapses. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation.

Attempts to mimic the human brain date back to works in the 1930s, 1940s and 1950s by Alan Turing, Warren McCullough, Walter Pitts, Donald Hebb and James von Neumann.

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pitts (Pitts and McCulloch 1943). The neurons were presented as conceptual components for circuits that could perform computational tasks.

There is no universally accepted definition for an artificial neural network although several definitions exist. Aleksander defined neural computing as 'the study of adaptable nodes which, through a process of learning from task examples, store experiential knowledge and make it available for use (Aleksander and Morton 1990). Haykin defines ANN as 'a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use' (Haykin and Bhattacharya 1992). Zurada defines ANNs as 'physical systems which can acquire, store and utilise experiential knowledge' (Zurada *et al.* 1997). Nigrin defines an ANN 'as a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore, each element operates asynchronously, thus there is no overall system clock' (Nigrin 1993). Fausett defines an ANN as 'an information processing system that has certain performance characteristics, such as adaptive learning, and parallel processing of information, in common with the biological neural networks' (Fausett 1994). From these definitions, it is reasonable to conclude that an ANN:

- consists of several simple processing elements called units;
- is well suited for parallel computations, since each unit operates independently of the other units;
- contains a high degree of interconnections between units;

- contains links between units, each with a weight (scalar value) associated with it; has adaptable weights that can be modified during training.

Why Artificial Neural Networks?

Artificial Neural Networks behave as trainable, adaptive and even self-organising information systems (Schalkoff 1997) and use a better strategy and methodology for problem solving. These make them more suitable to implement when compared to conventional computers that use the arithmetic approach (sets of instructions) for problem solving. Furthermore, conventional computers can only solve problems if the specific steps to follow are known in advance (problem solving by conventional computers is restricted to problems that we already understand and know how to solve). Neural networks have the remarkable ability to derive meaning from complicated or imprecise data and can extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Most importantly, the ability of neural networks to learn by example makes them suitable for tasks that cannot be solved algorithmically. A distinct strength of neural networks is their ability to generalise in the interpolation of input patterns that are new to the network. Neural networks provide, in many cases, input-output mappings with good generalisation capability.

Neural networks have been successfully trained to perform the task of control chart pattern recognition, for instance, (Pham and Oztemel 1996). The most popular type of neural network is the Multi-Layer Perceptron (MLP), which has found many applications related to Statistical Process Control (SPC), identification of abnormal patterns in control

charts and early detection of potential quality problems (Cheng 1995, 1997; Jacob and Luke 1993; Pham and Oztemel 1992; Pham and Oztemel 1996; Velasco and Rowe 1993).

Despite the capability and effectiveness of ANNs in a wide array of applications, there is need to highlight the advantages that make them suitable for use in juxtaposition with the PSO-Bees Algorithm that is applied to solve Control Chart Pattern Recognition (CCPR) and the Wood Defect Classification (WDC) problems. They include:

- **Adaptive learning:** A neural network is a dynamic system which has a built-in capability to adapt its weights to changing environments.
- **Self-organisation:** An artificial neural network can create its own organisation or representation of the information it receives during learning. There is little need for extensive characterisation of the problem other than through training.
- **Generalisation:** Neural networks are able to extrapolate to a certain extent from the training of previously unseen data.
- **Graceful degradation:** Partial destruction of a network leads to a corresponding degradation of performance. However, network capabilities such as generalisation may be retained even with major network damage.

  Neural networks have a gradual rather than sharp drop-off in performance as conditions worsen (Kohonen 1988).

Known limitations include:

- ANNs have poor explanation facilities. There are no facilities for justifying answers and responding to what or how questions.

- ANNs are not very good at performing symbolic computations. They cannot be used effectively for rule-based reasoning and arithmetic operations.
- The accuracy of an ANN's performance is dependent upon the quality of the training examples. It is difficult to find a complete and accurate set of training examples in real world problems.

MLP Neural Network Training with PSO-Bees Algorithm

Training an MLP network involves the minimisation of an error function which defines the total difference between the actual output and the desired output of the network over a set of training patterns. Training proceeds by presenting to the network a pattern of known class taken from the training set. The error component associated with that pattern is the sum of the squared differences between the desired and actual outputs of the network corresponding to the presented pattern. The procedure is repeated for all the patterns in the training set and the error components for all the patterns are summed to yield the value of the error function for an MLP network with a given set of connection weights.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (O_i^{actual} - O_i^{desired})^2 \qquad (3.14)$$

where

O_i^{actual} is the actual output vector $(y_1, \ldots, y_n)$

$O_i^{desired}$ is the desired output vector $(Y_1, \ldots, Y_n)$

N is the total number of training patterns.

In relation to the PSO-Bees Algorithm, each particle represents an MLP network with a particular set of weight vectors. The aim of the algorithm is to find the particle with the set of weight vectors producing the smallest value of the error function. The mathematical expressions for the velocity and position updates in the PSO-Bees Algorithm are given in equations 3.15 and 3.16 respectively.

$$V_{n+1} = wV_n + c_1 * rand_1 * (Pbest_n - P_n) + c_2 * rand_2 * (Gbest_n - P_n) \quad (3.15)$$

$$P_{n+1} = P_n + kV_{n+1} \quad (3.16)$$

The MLP network training procedure using the PSO-Bees Algorithm thus comprises the following steps:

1. Initialise the velocities and positions of the particles.

2. Apply the training data set to determine the value of the error function associated with each particle.

3. Using Equations (3.15) and (3.16), compute the new velocity and position of each particle based on the error values obtained in step 2 and in previous iterations.

4. Stop if the value of the error function has fallen below a predetermined threshold or the maximum allowed number of iterations has been exceeded.

5. Else, return to step 2.

The above procedural steps are applied to solve the CCPR and WDC problems presented in the next two sections.

3.6.2 Application to Control Chart Pattern Recognition Problem

This section presents the use of the PSO-Bees Algorithm to train an MLP neural network for the task of recognising different types of patterns in Statistical Process Control (SPC) charts and compares the results with those obtained by back-propagation (BP) training.

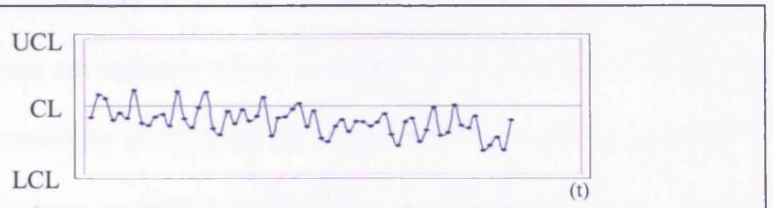
An informal definition of pattern recognition is telling things apart. Pattern recognition is the process of extracting information from an unknown data stream or signal and assigning it to one of the prescribed classes or categories (Haykin 1999).

This is especially important to industry. To gain the edge in today's competitive environment, companies must employ effective tools to ensure that their products are of the highest quality. They must also keep improving their production processes in order to raise quality standards. SPC is a quality improvement tool widely adopted in industry. It involves using control charts to enable a manufacturing engineer to compare the actual performance of a process with customer specifications and provide a process capability index to assess and guide quality improvement efforts. By means of simple rules, it is possible to determine if a process is out of control and needs corrective action. It is also possible to detect incipient problems and prevent the process from going out of control by identifying the type of patterns displayed by the control charts (Pham and Liu 1995; Pham and Oztemel 1992, 1995; Pham and Oztemel 1996).

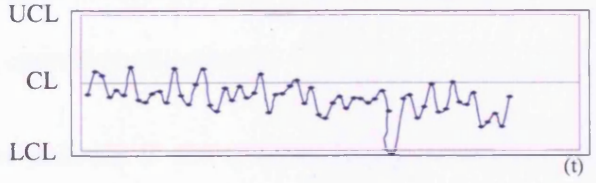
Observed variation of quality characteristics results from either natural variation (common cause) or unnatural variation (assignable cause). Natural variation exists in the manufacturing process regardless of how well the product is designed or how adequately

the process is maintained. By contrast, unnatural patterns resulting from unnatural variation are often associated with a specific set of assignable causes. The unnatural patterns contain valuable information relevant not only to the process parameters but also to the process changes.

Control charts are a graphical display of a quality characteristic that has been measured from a sample versus the sample number or time. The chart contains a centre line (CL) that represents the average value and the upper (UCL) and lower (LCL) lines allow variation limits of the quality characteristic under consideration (see Figure 3.3 (a) showing a typical chart for a process in statistical control and (b) a process out of statistical control).



(a) A typical control chart; control chart indicates the process is in statistical control



(b) A typical control chart; control chart indicates the process is out of statistical control

Figure 3.3: Process in and out of statistical control

The limits (UCL & LCL) are taken as the mean value plus or minus three standard deviations and they represent the boundaries of the range for unavoidable variations.

$$UCL = \mu + \frac{3\sigma}{\sqrt{n}} \tag{3.17}$$

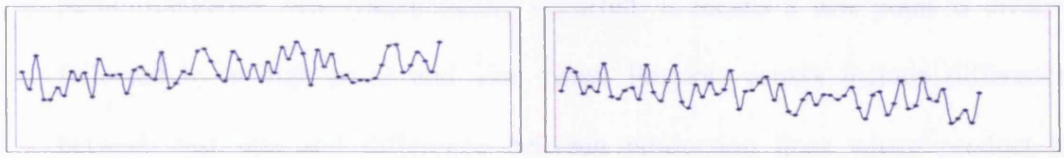
$$LCL = \mu - \frac{3\sigma}{\sqrt{n}} \tag{3.18}$$

The standard deviation is used because there is a high probability of 99.73% (http://en.wikipedia.org/wiki/Standard_deviation) that a sample measurement will fall within this range if the process is in control.

Control rules are used to detect out-of-control situations taking into consideration the very recent history of a process. A meagre X-bar chart only indicates when to look for disturbances but does not indicate where to look or the type / nature of the disturbance. This scenario is avoided by monitoring the long term behaviour of the process compared to allowing it to happen and later finding out. As mentioned earlier, the problem of monitoring a process to predict possible fault or malfunction is consequently reduced to that of recognising control chart patterns.

These patterns can indicate if the process being monitored exhibits gradual changes (trends – Figure 3.4), sudden changes (shifts – Figure 3.5), or periodic changes (cycles – Figure 3.6) or if it is operating normally (see Figure 3.7).

- **Trend patterns:** A trend can be defined as a continuous movement in either positive or negative direction. Possible causes include tool wear, operator fatigue, and equipment deterioration.



(a) Increasing Trend (b) Decreasing Trend

Figure 3.4: Increasing and Decreasing Trends

- **Shift patterns:** A shift can be defined as a sudden change above or below the average of the process. This change may be caused by an alternation in process setting, replacement of raw materials, minor failure of machine parts, or introduction of new workers, and so forth.

(a) Upward Shift (b) Downward Shift

Figure 3.5: Upward and Downward Shifts

- **Cyclic patterns:** Cyclic behaviours can be observed by a series of peaks and troughs occurring in the process. Typical causes are the periodic rotation of operators, systematic environmental changes or fluctuation in the production equipment.

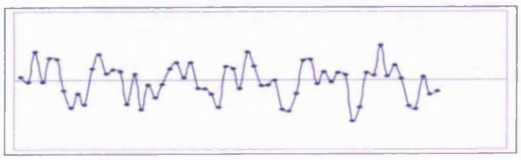


Figure 3.6: Cyclic Patterns

- **Systematic patterns:** The characteristic of systematic patterns is that a point-to-point fluctuation has systematically occurred. It means a low point is always followed by a high point and vice versa. Possible causes include difference between test sets and difference between production lines where product is sampled in rotation.

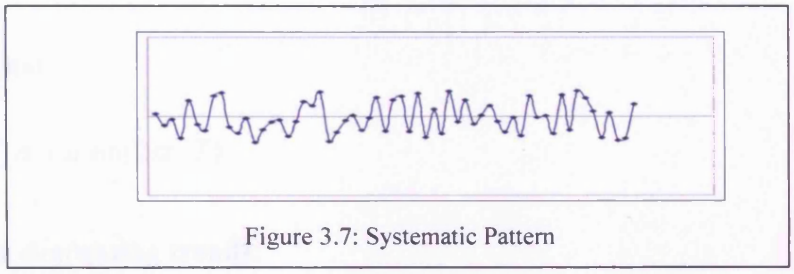


Figure 3.7: Systematic Pattern

In this work, each pattern was a time series comprising 60 points. The value $\bar{y}(t)$ at each point 't' was normalised to fall in the range [0, 1] according to the following equation:

$$\bar{y}(t) = \frac{y(t) - y_{\min}}{y_{\max} - y_{\min}} \tag{3.19}$$

where

$\bar{y}(t)$ = scaled pattern value (in the range 0 to 1)

$y_{\min}$ = minimum allowed value (taken as 35)

$y_{\max}$ = maximum allowed value (taken as 125)

Training and Test Data

A total of 1500 patterns (250 patterns in each of the six classes) were generated using the following equations:

1. **Normal patterns:**

$$y(t) = \mu + r(t)\,\sigma \tag{3.20}$$

2. **Cyclic patterns:**

$$y(t) = \mu + r(t)\,\sigma + a\sin(2\pi t / T) \tag{3.21}$$

3. **Increasing or decreasing trends**

$$y(t) = \mu + r(t)\,\sigma \pm g\,t \tag{3.22}$$

4. **Upwards or downwards shifts:**

$$y(t) = \mu + r(t)\,\sigma \pm k\,s \tag{3.23}$$

where

μ mean value of the process variable being monitored (taken as 80 in this work)

σ standard deviation of the process (taken as 5)

a amplitude of cyclic variations (taken as 15 or less)

g magnitude of the gradient of the trend (taken as being in the range 0.2 to 0.5)

k parameter determining the shift position (= 0 before the shift position; = 1 at the shift position and thereafter)

r normally distributed random number (between – 3 and +3)

s magnitude of the shift (taken as being in the range 7.5 to 20)

t	discrete time at which the pattern is sampled (taken as being within the range 0 to 59)
T	period of a cycle (taken as being in the range 4 to 12 sampling intervals)
$y(t)$	sample value at time t

In total, 498 patterns (83 in each class) were used for training the MLP classifier and 1002 patterns (167 in each class) were employed for testing the trained classifier.

MLP Network Configuration used for the CCPR Problem

The MLP configuration adopted had three layers: an input layer, a hidden layer and an output layer (Figure 3.8).

- The input layer had 60 neurons, one for each point in a pattern.
- The hidden layer consisted of 35 neurons. The number of hidden neurons adopted was the same as in previous work on identifying control chart patterns using BP-trained networks (Pham and Oztemel 1992).
- The output layer comprised 6 neurons, one for each of the six pattern classes.

The input neurons performed no processing roles, acting only as buffers for the input signals. Processing was carried out by the hidden and output neurons. The activation function used was the sigmoid function.

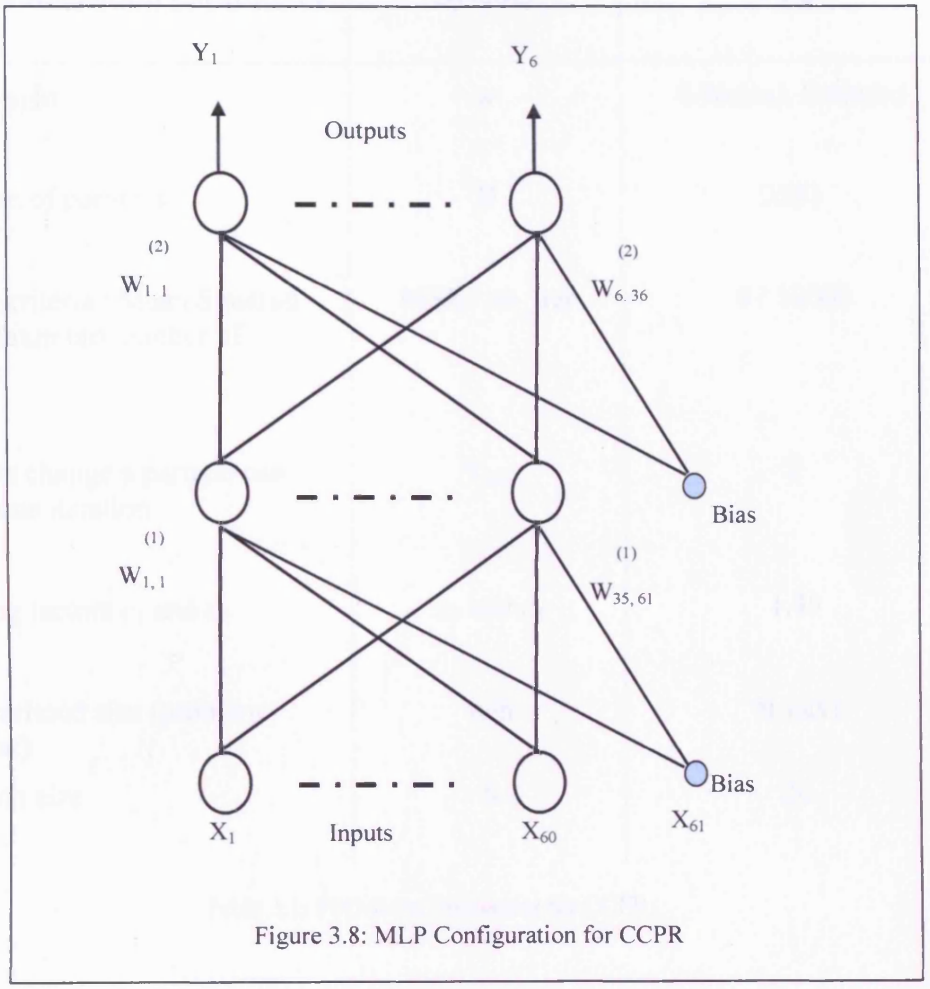


Figure 3.8: MLP Configuration for CCPR

PSO-Bees Algorithm Parameters

Table 3.1 shows the parameter values empirically chosen for the PSO-Bees Algorithm. The positions of the particles were initialised by setting all weight values randomly within the range -1 to 1.

PSO-Bees Algorithm Parameters	Symbol	Value
Inertia weight	w	0.9(max), 0.4(min)
Dimension of particles	D	2351
Stopping criteria : Mean Squared Error / Maximum number of iterations	MSE / no_iter	8 / 10000
Maximum change a particle can make in one iteration	V_{max}	2
Weighting factors c_1 and c_2	c_1 and c_2	1.49
Neighbourhood size (problem dependent)	ngh	5(max)
Population size	S	20

Table 3.1: PSO-Bees Parameters for CCPR

Control Chart Pattern Recognition Results

Table 3.2 presents the classification (training and test) results obtained for ten separate runs of the PSO-Bees Algorithm. A typical plot of how the classification accuracy evolves during training is shown in Figure 3.9. For comparison, Table 3.3 summarises the results produced using other classifiers including the conventional BP-trained classifier.

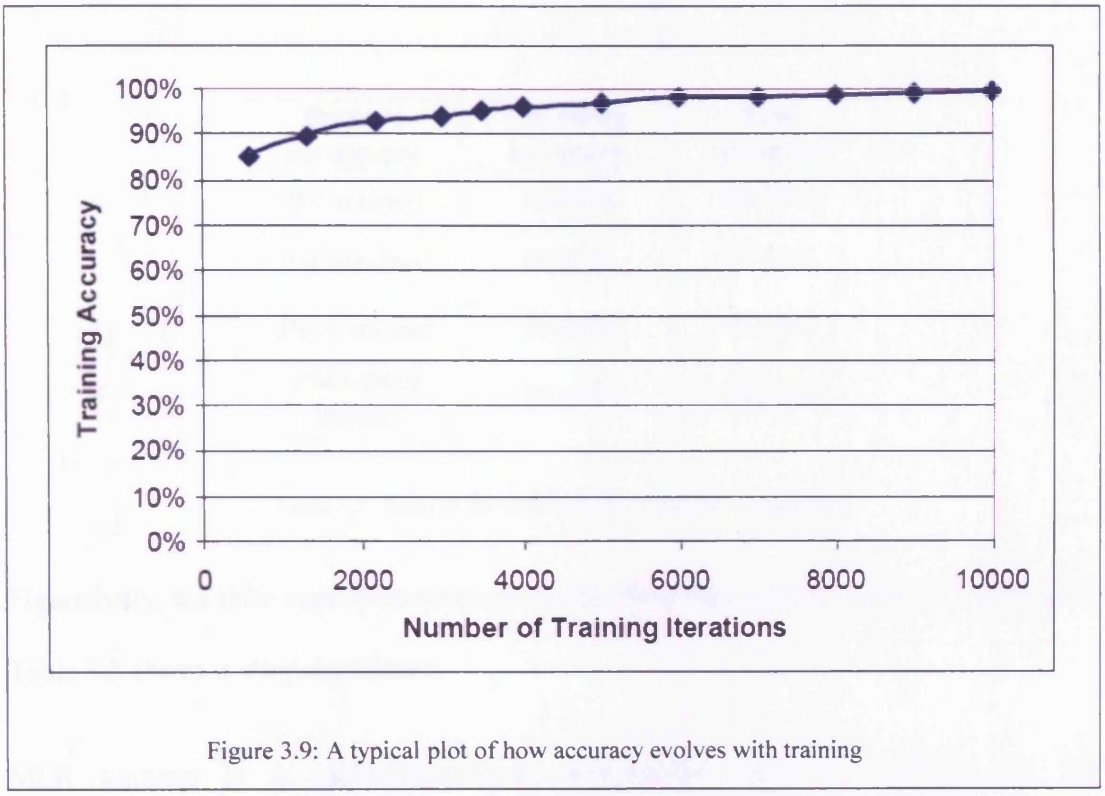


Figure 3.9: A typical plot of how accuracy evolves with training

Run number	Training accuracy	Test accuracy
1	99.62%	99.21%
2	99.78%	99.18%
3	99.61%	99.19%
4	99.65%	99.13%
5	99.64%	99.17%
6	99.63%	99.18%
7	99.64%	99.21%
8	99.62%	99.18%
9	99.69%	99.22%
10	99.65%	99.17%
Maximum	99.78%	99.22%
Minimum	99.61%	99.13%
Mean	**99.65%**	**99.18%**

Table 3.2: Classification results obtained with PSO-Bees Algorithm

Pattern recogniser	Learning accuracy	Test accuracy
BP-trained	96.00%	95.20%
PSO-trained	99.22%	97.13%
Bees-trained	98.20%	99.10%
PSO-Bees trained	99.65%	99.18%

Table 3.3: Results for different MLP pattern recognisers

Figuratively, the little improvement made by the PSO-Bees trained classifier presented in Table 3.3 above is very significant.

MLP training is a multidimensional optimisation problem. Despite the high dimensionality of the problem (each particle represented 2351 (61 * 35 + 36 * 6) parameters that had to be determined), the algorithm still succeeded in training more accurate classifiers than did the well-established BP algorithm.

A lingering question persists 'what is the statistical significance of the result presented in Table 3.3'?

To check the statistical significance of the result, I performed the T-TEST. The T-TEST checks the relationship between two variables, in this case two different algorithms and it tries to answer two questions:

1. what is the probability that a relationship exists?
2. if it does, how strong is the relationship?

In other words, tests for statistical significance are used to address the question: what is the probability that the relationship between two variables is really just a chance occurrence?

Using T-Tests

T-Tests are tests for statistical significance used with interval and ratio level data. T-tests are often employed in several different types of statistical tests:

- to test whether there are differences between two groups on the same variable, based on the mean (average) value of that variable for each group;
- to test whether a group's mean (average) value is greater or less than some standard;
- to test whether the same group has different mean (average) scores on different variables;

The T-Test assesses whether the means of two groups are *statistically* different from each other. This is shown graphically in Figure 3.10 (Web Centre for Social Research Methods). A distribution for the treated group is in red while that for the control group is in green.

Alpha (α) is the result from the T-Test and it has three values of 0.05, 0.01, or 0.001. When:

- $\alpha < 0.05$, there is significant difference in the group means.
- $\alpha < 0.01$, there is more significant difference in the group means.
- $\alpha < 0.001$, there is most significant difference in the group means.

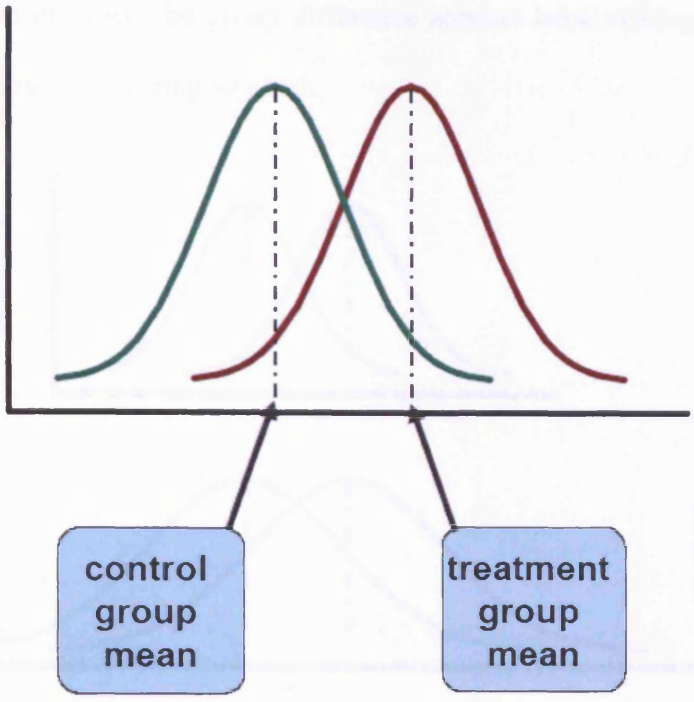


Figure 3.10: Idealised distributions for treated and comparison group post test values

Another question persists 'what does it mean to say that the averages for the two groups are statistically different'?

The answer is shown in Figure 3.11 (Web Centre for Social Research Methods). The first thing to notice about the three situations is that *the difference between the means is the same in all three*. Figure 3.11 shows that the three situations don't look the same; they tell very different stories. The top distribution shows a case with moderate variability of scores within each group. The second distribution shows the high variability case while the third distribution shows the case with a low variability. Clearly, one can conclude that two groups appear most different or distinct in the bottom or low-variability case. *Why?* There is relatively little overlap between the two bell-shaped curves. On the other hand,

in the high variability case, the group difference appears least striking because the two bell-shaped distributions overlap so much.

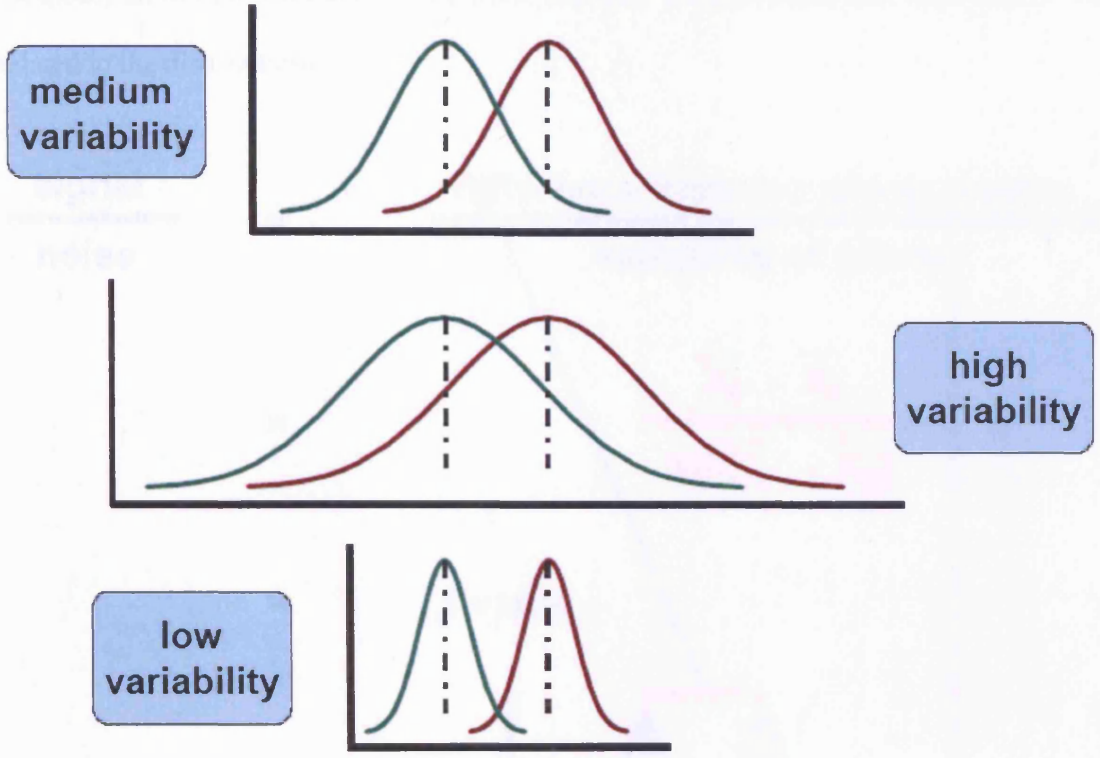


Figure 3.11: Three scenarios for differences between means

This leads to a very important conclusion: when looking at the differences between scores for two groups, there is the need to judge the difference between their means relative to the spread or variability of their scores. *The t-test does just this.*

Statistical Analysis of the t-test

The formula for the t-test is a ratio. The top part of the ratio is the difference between the two means or averages. The bottom part is a measure of the variability or dispersion of the scores. The formula is an example of the signal-to-noise metaphor in research. The difference between the means is the signal that is introduced by the program into the data.

The bottom part of the formula is a measure of variability that is essentially the noise that makes it harder to see the group difference. Figure 3.12 (Web Centre for Social Research Methods) shows the formula for the t-test and how the numerator and denominator are related to the distributions.

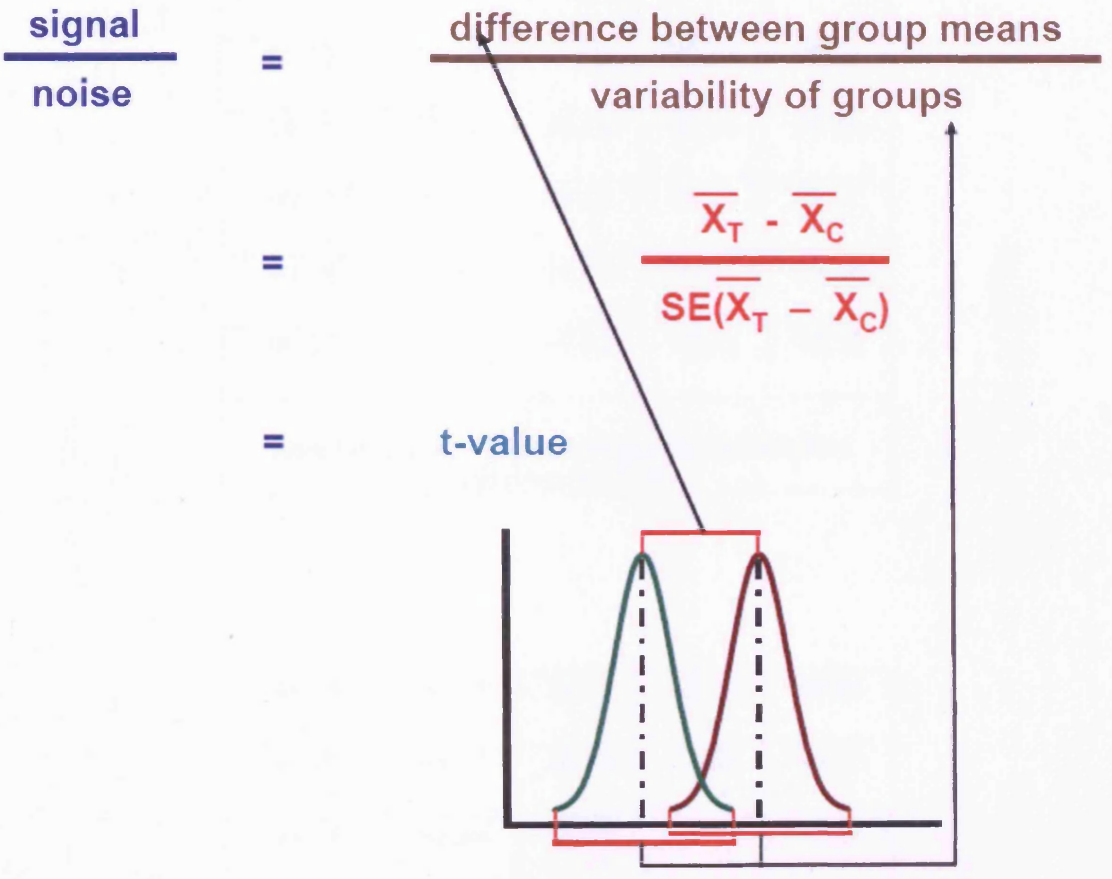


Figure 3.12: formula for the t-test and how the numerator and denominator are related to the distributions

The T-Test was conducted between the PSO-Bees Algorithm and the original Bees Algorithm. As mentioned earlier, both algorithms were applied 30 times to train an MLP neural network for the Control Chart Pattern Recognition problem.

Figure 3.13 shows a plot of the test accuracies produced by both algorithms. The values of the plot are presented in Tables 3.4 and 3.5 for the PSO-Bees Algorithm and the original Bees Algorithm respectively.

99.21	99.13	99.19	99.13	99.17
99.22	99.21	99.22	99.13	99.22
99.21	99.22	99.13	99.22	99.14
99.13	99.15	99.18	99.13	99.17
99.18	99.15	99.22	99.22	99.16
99.22	99.13	99.21	99.13	99.13

Table 3.4: Testing accuracies obtained by the PSO-Bees Algorithm for CCPR

98.28	98.15	98.51	98.46	98.44
98.99	98.43	98.84	98.45	98.95
98.28	98.84	98.49	98.92	98.41
98.17	98.44	98.41	98.12	98.43
98.15	98.79	98.84	98.43	98.15
98.46	98.15	98.79	98.28	98.49

Table 3.5: Testing accuracies obtained by the original Bees Algorithm for CCPR

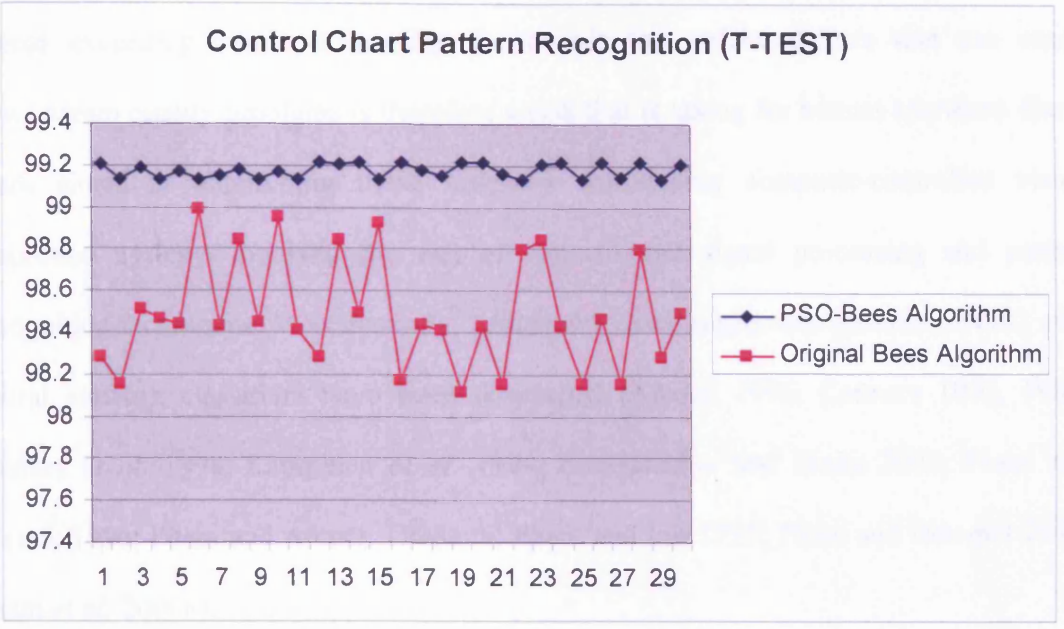


Figure 3.13: Plot of test accuracies obtained by the PSO-Bees Algorithm and the original Bees Algorithm for CCPR

I obtained an alpha value of 2.51E-20 from the T-Test. This value indicates the results obtained by both the PSO-Bees Algorithm and the original Bees Algorithm is most significantly different with a confidence level above 99%.

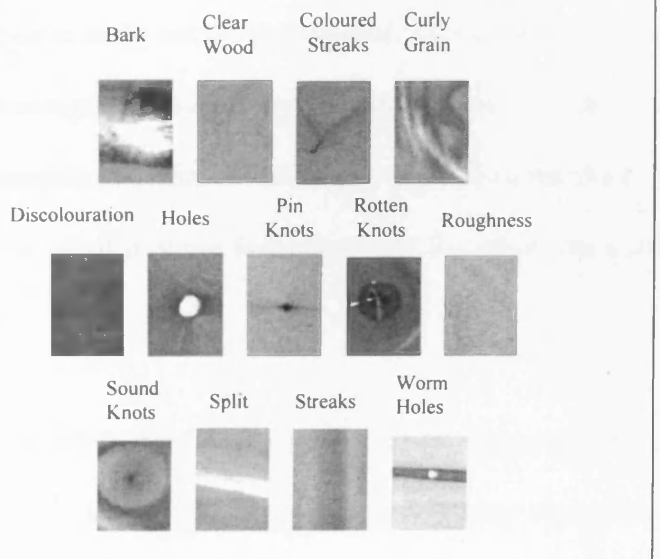
3.6.3 Application to Wood Defect Classification

This section presents a system employing a Multi-Layer Perceptron network as a pattern classifier. Multi-Layer Perceptrons are usually trained by back-propagation. However, the training technique which is based on gradient information sometimes produces classifiers with poor performances because of the existence of local optima where the gradient is null. The Multi-Layer neural classifier developed in this work for Wood Defect Classification was trained using the PSO-Bees Algorithm.

Wood veneer boards are manufactured on fast production lines where boards can move at speeds exceeding 20m/s. Inspecting the boards for surface defects that can cause downstream quality problems is therefore a task that is taxing for human operators. Early work aimed at automating these tasks by introducing computer-controlled visual inspection systems involved the use of conventional signal processing and pattern recognition techniques. More recently, automated visual inspection systems (AVIS) with neural network classifiers have been developed (Alcock 1996; Conners 1992, 1983; Estévez *et al.* 1998; Lampinen *et al.* 1994; Packianather and Drake 2005; Pham and Alcock 1996; Pham and Alcock 1998a, b; Pham and Liu 1995; Pham and Oztemel 1996; Pham *et al.* 2006b).

Wood Defect Classification Problem

There are twelve common types of defects on wood veneer surfaces. These are shown in Figure 3.14, together with a photograph of defect free (clear) wood.



3.14: Categories of veneer wood images

As mentioned above, defect classification was performed with a trained Multi-Layer Perceptron. Features were first extracted from different wood images containing known defect types or no defects and the Multi-Layer Perceptron was taught to distinguish between the features of those images. In total, as performed in previous work (Conners 1983; Koivo and Kim 1986; Koivo 1994), seventeen features were extracted from the wood images and used to train the Multi-Layer neural classifier.

The wood defect classification problem is thus reduced to that of mapping a given set of seventeen features extracted from an image onto one of the image categories shown in Figure 3.14.

Multi-Layer Perceptron training

The Multi-Layer Perceptron network used had three layers: an input layer, a hidden layer, and an output layer of neurons. The neurons between adjacent layers are fully linked by connections, the weights of which are to be determined through training. The training of a Multi-Layer Perceptron to carry out a mapping task such as that of transforming a feature vector into an image category is essentially an optimisation problem. The aim is to select the values of the connection weights of the neural network to minimise the total mapping error calculated over a set of training feature vectors for which the corresponding image categories are known.

In an application of the PSO-Bees Algorithm to the training problem, each particle is a multi-dimensional weight vector P_n representing a candidate classifier. When a training feature vector is provided, the weight vector is used to calculate the response of the

classifier. The difference between that response and the correct known response is the classification error corresponding to that particular training feature vector. The average of the squared error for all the feature vectors in the training set gives the overall performance (fitness) of the candidate classifier. By adjusting the position of each particle P_n according to equations (3.1) and (3.2), the PSO-Bees Algorithm changes the weight vectors and hence the performance of each candidate classifier, eventually directing the swarm of particles towards the position with the minimum classification error.

The optimum is considered found and the algorithm stops when the mean squared error has fallen below a given threshold. Alternatively, the algorithm also stops when the maximum number of iterations is reached.

Wood Defect Classification Results

A classifier structure with 17 input neurons (each neuron to receive a component of the feature vector), 13 output neurons (each neuron corresponding to an image category) and 51 hidden neurons were adopted. The number of hidden neurons was same as that used by Packianather (Packianather and Drake 2005) who employed the Taguchi Design of Experiments technique to determine the most appropriate value for this parameter. The input neurons acted only as buffers and performed no processing function, transmitting directly the values of the features (regularised between -1 and +1) to the hidden layer neurons and then onward to the output neurons. A diagram of the MLP configuration used is presented in Figure 3.15.

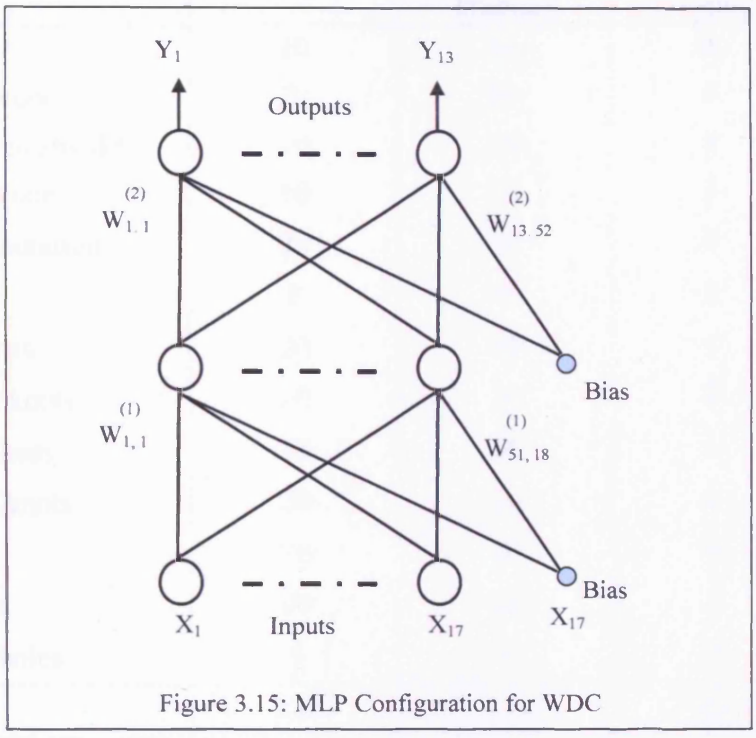


Figure 3.15: MLP Configuration for WDC

Processing was carried out by the hidden and output neurons. The output function employed was the hyperbolic tangent function. A constant bias was added to the activation of each neuron prior to the calculation of the neuron output. The classifier comprises in total 1594 connections; 17 x 51 from the input layer to the hidden layer, 51 x 13 from the hidden layer to the output layer and 51 + 13 bias connections. The classifier was trained using a set of 185 feature vectors. The trained classifier was tested on a different set of 47 feature vectors. Table 3.6 gives details of the training and test vectors.

Image Class	Total	Used for Training	Used for Testing
Bark	20	16	4
Clear wood	20	16	4
Coloured streaks	20	16	4
Curly grain	16	13	3
Discolouration	20	16	4
Holes	8	6	2
Pin knots	20	16	4
Rotten knots	20	16	4
Roughness	20	16	4
Sound knots	20	16	4
Splits	20	16	4
Streaks	20	16	4
Wormholes	8	6	2
Total	232	185	47

Table 3.6: Training and test sets for WDC

Table 3.7 shows the parameter values empirically chosen for the PSO-Bees Algorithm. The positions of the particles were initialised by setting all weight values randomly within the range -1 to 1.

PSO-Bees Algorithm Parameters	Symbol	Value
Inertia weight	w	1(max), 0(min)
Dimension of particles	D	1594
• Stopping criteria : Mean Squared Error	MSE	6
• Maximum number of iterations	I	1,100
Maximum change a particle can make in one iteration	V_{max}	2
Weighting factors c1, c2	$c1, c2$	1.49
Neighbourhood size	ngh	3
Population size	S	40

Table 3.7: PSO-Bees Algorithm Parameters for WDC

The PSO-Bees Algorithm with the parameters given in Table 3.7 was applied 30 times to train 13 different classifiers. Table 3.8 below shows the results for the wood defect classification obtained by previously applied algorithms.

Method	Mean Accuracy
MDC (Non NN)	63.12 %
NN – Back-propagation	86.52 %
NN – Bees Algorithm	86.52 %
NN – Particle Swarm Optimisation Algorithm	89.79 %
NN – PSO-Bees Algorithm	92.16 %

Table 3.8: Results of wood defect identification

As shown in Table 3.8, the mean classification accuracy obtained with the PSO-Bees Algorithm is 92.16% while that produced by the conventional PSO Algorithm is 89.79%. By comparison, the accuracy for 13 Multi-Layer Perceptron classifiers trained by the Bees Algorithm (Pham *et al.* 2006b) and back propagation (Packianather and Drake 2006) was 86.52%. Clearly, the PSO-Bees Algorithm gave classifiers with a superior performance.

Despite the high dimensionality of the problem (each particle represented 1594 parameters that had to be determined), the PSO-Bees Algorithm trained classifiers were able to identify the defects more accurately than did classifiers trained using the original PSO Algorithm and the well-established back-propagation method.

> *A question persists 'what is the statistical significance of the result presented in Table 3.8'?*

To check the statistical significance of the result, a T-TEST had to be performed which checks the relationship between two variables, in this case two different algorithms.

The T-Test was conducted between the PSO-Bees Algorithm and the original Particle Swarm Optimisation Algorithm. As mentioned earlier, both algorithms were applied 30 times to train neural networks for the wood defect classification problem.

Figure 3.16 shows a plot of the test accuracies produced by both algorithms. The values of the plot are presented in Tables 3.9 and 3.10 for the PSO-Bees Algorithm and the original Particle Swarm Optimisation Algorithm respectively.

92.31	92.36	91.93	92.33	92.11
92.24	92.13	92.37	91.78	92.12
92.53	92.21	92.15	92.32	92.48
92.11	92.27	92.33	92.13	91.98
92.17	92.18	92.09	92.17	92.19
91.33	92.01	92.13	92.19	92.11

Table 3.9: Testing accuracies obtained by the PSO-Bees Algorithm for WDC

89.79	88.91	89.20	89.12	89.94
89.47	89.69	89.55	89.79	89.86
90.18	89.78	89.97	89.76	89.96
89.94	89.74	89.98	89.79	90.20
89.94	89.96	89.74	89.93	90.16
89.96	89.74	89.92	89.64	89.97

Table 3.10: Testing accuracies obtained by the original Particle Swarm Optimisation Algorithm for WDC

- as benchmarks for comparing different optimisation approaches (Zitzler *et al.* 2000),
- to derive theoretical results since they are normally well understood in a mathematical sense (Jansen and Wegener 2007),
- as a basis to verify theories (Burke *et al.* 2002a),
- as a playground to test new ideas, research, and developments,
- as easy-to-understand examples to discuss the features and problems of optimisation.

Mathematical benchmark functions are useful for testing and comparing techniques based on real vectors ($X = R^n$). Nonetheless, they only require such vectors as solution candidates, i.e. elements of the problem space X.

In this work, ten standard tests on function optimisation problems were used to benchmark the PSO-Bees Algorithm as a global optimiser. The results obtained from each of the standard benchmark test functions were compared with other global optimisation algorithms such as the deterministic simplex method (SIMPSA), the stochastic simulated annealing optimisation procedure (NESIMPSA), the standard Genetic Algorithm (GA), the Ants Colony System (Ants), the Bees Algorithm (BA) and the standard Particle Swarm Optimisation Algorithm (PSO).

The test functions include: DeJong, Goldstein & Price, Branin, Martin & Gaddy, Rosenbrock1 (a & b), Rosenbrock2, Hyper Sphere, Griewangk, Ackley and the Schwefel functions.

Table 3.11 shows the test functions and their global optima while Table 3.12 presents the results obtained by the PSO-Bees Algorithm for 100 independent runs.

Detailed information (visualisation) on these functions used to benchmark the PSO-Bees Algorithm is provided in Appendix E.

No	Reference	Interval	Test Function	Global Optimum
1	De Jong	[-2.048, 2.048]	$\max F = (3905.93) - 100\,(x_1^2 - x_2^2) - (1 - x_1)^2$	X[1,1] F=3905.93
2	Goldstein & Price	[-2, 2]	$\min F = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $X[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	X[0,-1] F=3
3	Branin	[-5, 10]	$\min F = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e$ $a = 1, b = \frac{5.1}{4}\left(\frac{7}{22}\right)^2, c = \frac{5}{22}X7, d = 6, e = 10, f = \frac{1}{8}X\frac{7}{22}$	X[-22/7,12.275] X[22/7,2.275] X[66/7,2.475] F=0.3977272
4	Martin & Gaddy	[0, 10]	$\min F = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$	X[5,5] F=0
5	Rosenbrock -1	(a) [-1.2, 1.2] (b) [-10, 10]	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	X[1,1] F=0
6	Rosenbrock - 2	[-1.2, 1.2]	$\min F = \sum_{i=1}^{3} \{100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\}$	X[1,1,1,1] F=0
7	Hyper sphere model	[-5.12, 5.12]	$\min F = \sum_{i=1}^{6} x_i^2$	X[0,0,0,0,0,0] F=0
8	Griewangk	[-512, 512]	$\max F = \frac{1}{0.1 + \left(\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)}$	X[0,0,0,0,0,0,0,0,0,0] F=10
9	Ackley	[-5.12, 5.12]	$f(x) = 20 + e - 20e^{-0.2\cdot\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	X [0, …, 0] F=0
10	Schwefel	[-500, 500]	$f(x) = 418.9829 \cdot n - \sum_{i=1}^{n} (-x_i \sin(\sqrt{\lvert x_i \rvert}))$	X [1, …, 1] F=0

Table 3.11: Test Functions (Mathur *et al.* 2000)

func no	SIMPSA		NE SIMPSA		GA		ANT		Bees Algorithm		PSO Algorithm		PSO-Bees Algorithm	
	success %	mean no of func. evals	success %	mean no of func. evals	success %	mean no of func. evals	success %	mean no of func. evals	success %	mean no of func. evals	success %	mean no of func. evals	success %	mean no of func. evals
1	***	****	****	****	100	10160	100	6000	100	868	100	872	100	815
2	***	****	****	****	100	5662	100	5330	100	999	100	1008	100	879
3	***	****	****	****	100	7325	100	1936	100	1657	100	1594	100	1463
4	***	****	****	****	100	2844	100	1688	100	526	100	507	100	486
5a	100	10780	100	4508	100	10212	100	6842	100	631	100	609	100	594
5b	100	12500	100	5007	***	****	100	7505	100	2306	100	2281	100	1829
6	99	21177	94	3053	***	****	100	8471	100	28529	100	27736	100	21105
7	***	****	****	****	100	15468	100	22050	100	7113	100	6930	100	6794
8	***	****	****	****	100	200000	100	50000	100	1847	100	1851	100	1798
9	***	***	***	****	***	****	***	****	***	****	100	2247	100	1979
10	***	***	***	****	***	****	***	****	***	****	100	4583	100	3927

**** Data not available

Table 3.12: Results of test functions

Table 3.12 presents the mean number of function evaluations obtained from 100 independent runs. The table is used to compare ten benchmark functions examined by the PSO-Bees Algorithm with the deterministic simplex method and the stochastic simulated annealing optimisation procedure (SIMPSA and NE SIMPSA), the genetic algorithm (GA), the ant colony approach (ANT), the Bees Algorithm and the Particle Swarm Optimisation (PSO) Algorithm.

The optimisation stopped when the difference between the maximum fitness obtained and the global optimum was less than 0.1% of the optimum value, or less than 0.001, whichever is smaller. In the case when the optimum was 0, the solution was accepted if it differed from the optimum by less than 0.001.

As shown in Table 3.12, the PSO-Bees Algorithm performed significantly better compared to the other global optimisation algorithms as indicated by the smallest number of function evaluations converging to the global optimum of the respective functions. The PSO-Bees Algorithm found the optimum with better accuracy in less time.

3.7 Summary

This chapter has presented the Particle Swarm Optimisation - Bees Algorithm (PSO-Bees Algorithm), a modification to the Particle Swarm Optimisation algorithm. The algorithm incorporates adaptive neighbourhood and global random search around the global best particle. It combines the fast convergence property of the PSO Algorithm and the inherent ability of the Bees Algorithm to avoid being trapped in local optima.

Furthermore, the chapter showed that the PSO-Bees Algorithm is robust and exhaustively searches the problem space producing optimum result. The algorithm solved the problem of premature convergence of the PSO Algorithm that limits the ability of the algorithm to find the global optimum of objective functions. The results obtained on applying the algorithm to train neural networks for CCPR and the WDC problems have been presented to further reinforce the performance and aptitude of the algorithm as a global optimiser. Finally, the presentation of results on mathematical benchmark functions shows the enhanced performance of the PSO-Bees Algorithm. The algorithm is proficient and capable of performing efficiently and effectively well in varied applications.

Chapter 4: Improving the Bees Algorithm with the Particle Swarm Optimisation Algorithm - *Improved* Bees Algorithm

What no spouse of a writer can ever understand is that a writer is working when he's staring out the window.

This chapter presents the *improved* Bees Algorithm, an enhanced version of the original Bees Algorithm. The *improved* Bees Algorithm integrates cooperation and communication between different neighbourhoods of the original Bees Algorithm in order to find the global optimum. The proposed communication and cooperation strategies enhanced the performance and convergence of the algorithm. It ensures the algorithm search only the promising areas of the search space and secondly, stops the need for 'killing' Bees as previously employed in other variants of the Bees Algorithm. Thirdly, this approach reduces the number of function evaluations of the algorithm in finding the global optimum of functions. Next, the *improved* Bees Algorithm is described in detail followed by the graphical representation of the operations of the algorithm. Finally, the chapter concludes with a presentation of the results obtained from its application to the mechanical design optimisation problems, specifically, the designs of welded beams (single & multi objectives), coiled springs and tests on mathematical benchmark functions.

4.1 The *improved* Bees Algorithm

Section 2.2 of Chapter 2 details extensively an introduction to the original Bees Algorithm.

With reference to Figure 2.4 of Chapter 2 (Pham *et al.* 2005, 2006a) showing the pseudo code of the original Bees Algorithm, an observation of the aerial view of the operations of the algorithm show a swarm of bees flying across the search space as shown in Figure 4.1. On the contrary, on zooming into the algorithm, it can be seen that there are independent patches of bees searching the problem space with no communication or cooperation amongst these patches to essentially help in the search process as in the case of the PSO Algorithm. See Figure 4.2.

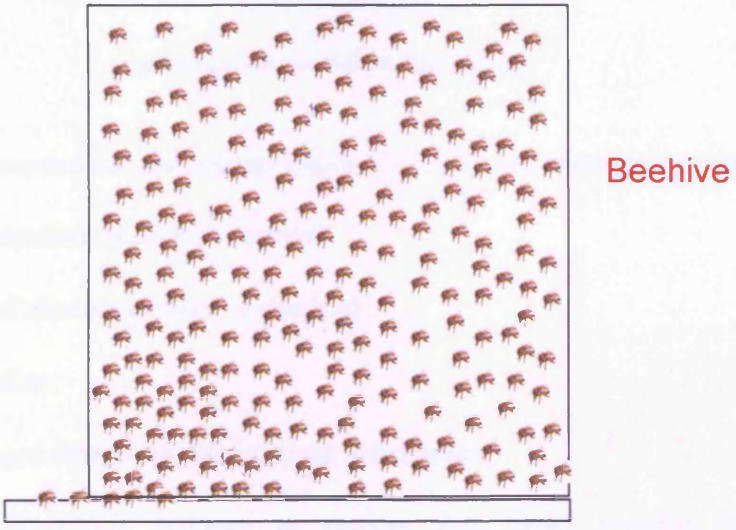


Figure 4.1: Swarm of Bees

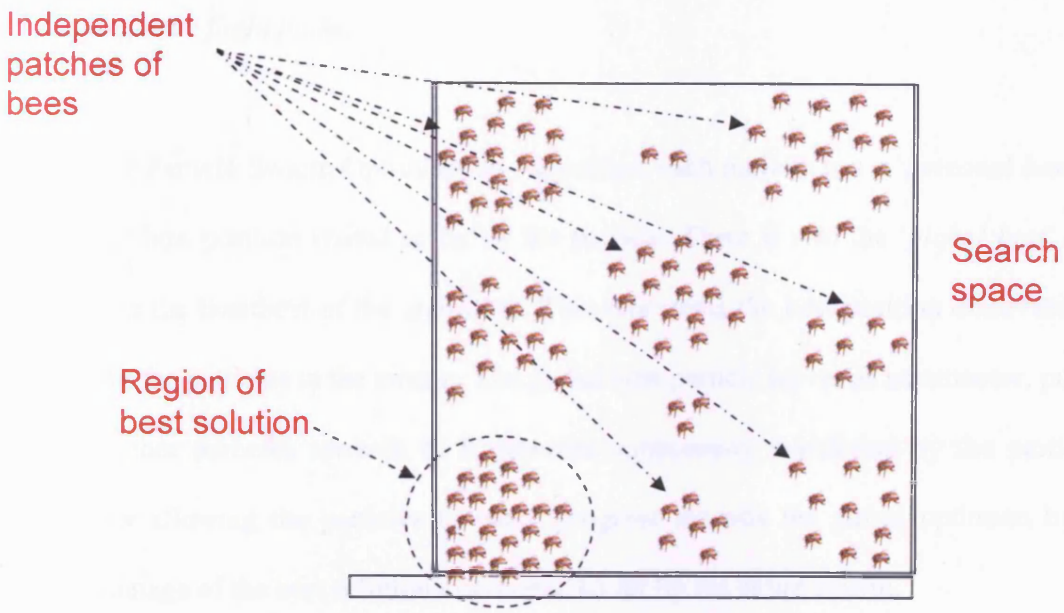


Figure 4.2: Swarm of Bees (Zoomed in)

The introduction of cooperation and communication is achieved through the use of so called *momentum*. Momentum takes into account:

- the number of elite bees (number of patches);
- the current solution;
- the neighbourhood size using the Gaussian distribution;
- the number of weights assigned to patches with better solution (weights proportional to the quality solution).

In the proposed *improved* Bees Algorithm with momentum, there is a sort of biased random search around and in the direction of the current best solution. In other words, there is global information shared amongst the patches (neighbourhoods) influencing the

search process. *In addition, as the other bees are attracted and move at a faster pace toward the region of the best solution, they will discover even better solutions (if any) along their flight paths.*

In the Particle Swarm Optimisation Algorithm, each particle has a '*personal best*' which is the best position visited so far by the particle. There is also the '*global best*' quantity that is the heartbeat of the algorithm. This represents the best position discovered so far by all the particles in the swarm. The global best particle serves as an attractor, pulling all the other particles towards it. It prevents unnecessary wandering by the particles but rather allowing the particles to make progress towards the global optimum by taking advantage of the best solution discovered so far by the entire swarm.

These unique and fascinating features of the PSO Algorithm are introduced to the Bees Algorithm. The result is called the "*Improved* Bees Algorithm". Figure 4.3 shows the effect of the momentum. Bees in other patches are attracted and all move towards the region of the best solution by exploiting the global information shared between patches according to the quality and quantity of the solution found.

Figure 4.4 shows the pseudo code of the *improved* Bees Algorithm for a simple one dimensional problem.

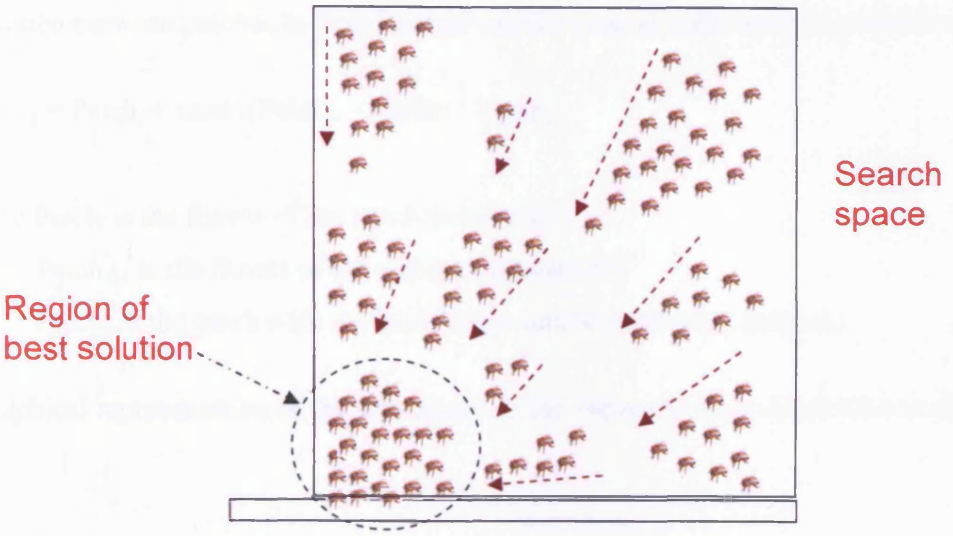


Figure 4.3: Swarm of Bees (zoomed in) with momentum attracted to the region of best solution

```
1.  Initialise population with random solutions.
2.  Evaluate fitness of the population.
3.  While (stopping criterion not met).
    //Forming new population.
4.  Select patches for adaptive neighbourhood search.
5.  Assign more weights to patches with better solution (weights
    proportional to the quality and quantity of the solution).
6.  Propagate global information of best known patch across the
    entire swarm.
7.  According to the globally shared information, recruit bees for
    the selected patches (more bees for patches with more weights)
    and evaluate their fitness.
8.  Select the fittest bee from each patch.
9.  Patch with best fitness attracts patches with low fitness
10. Assign the remaining bees to search randomly and evaluate their
    fitness.
11. End While.
```

Figure 4.4: Pseudo code of the *improved* Bees Algorithm

Attraction between patches in Step 9 of the pseudo code is achieved using equation 4.1.

$$Patch_{i+1} = Patch_i + rand\ ((Patch_b - Patch_i) / Patch_i) \quad (4.1)$$

Where $Patch_i$ is the fitness of the patch at iteration i

$Patch_{i+1}$ is the fitness of the patch at iteration i+1

$Patch_b$ is the patch with the best fitness (attractor of other patches)

A graphical representation of the operations of the *improved* Bees Algorithm is presented next.

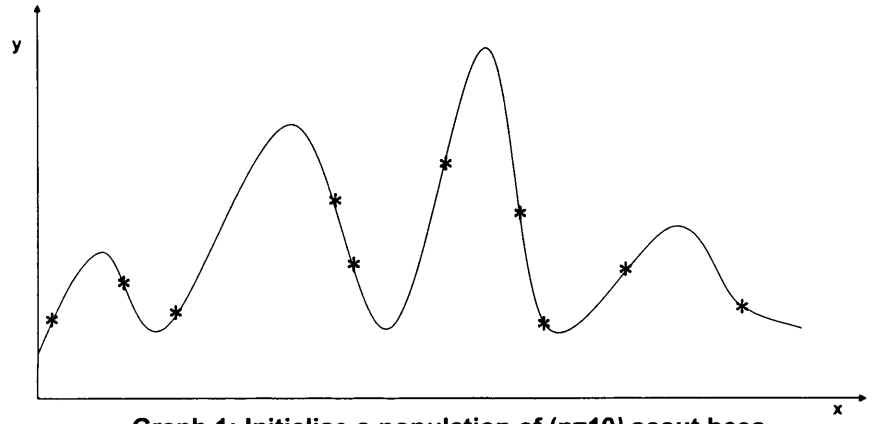
4.2 Operation of the *improved* Bees Algorithm

As mentioned earlier, the *improved* Bees Algorithm incorporates cooperation and communication between different neighbourhoods of the original Bees Algorithm in order to find the global optimum in a methodology that is similar to the cooperation and communication strategies found in the PSO Algorithm.

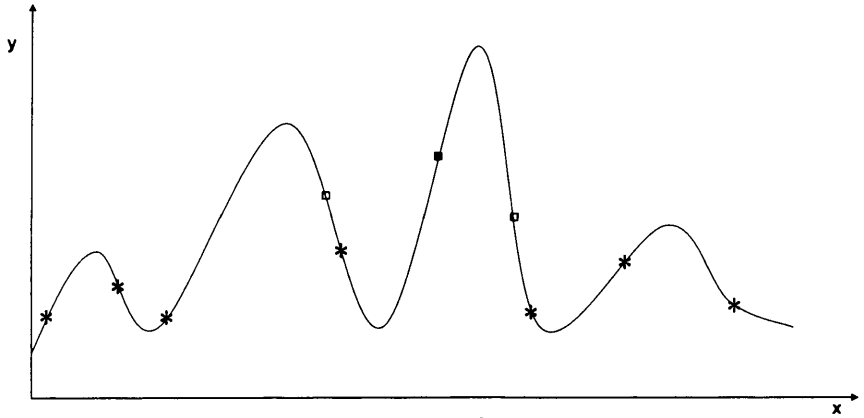
The proposed cooperation and communication strategies influence the search process by ensuring the algorithm search only in the promising areas of the search space. Secondly, it stops the need for 'killing' Bees as previously employed in other variants of the Bees Algorithm and thirdly, this approach reduces the number of function evaluations of the algorithm in finding the global optimum of objective functions.

Figure 4.5 illustrates the operations of the *improved* Bees Algorithm for a simple one-dimensional optimisation problem.

A Typical Instance

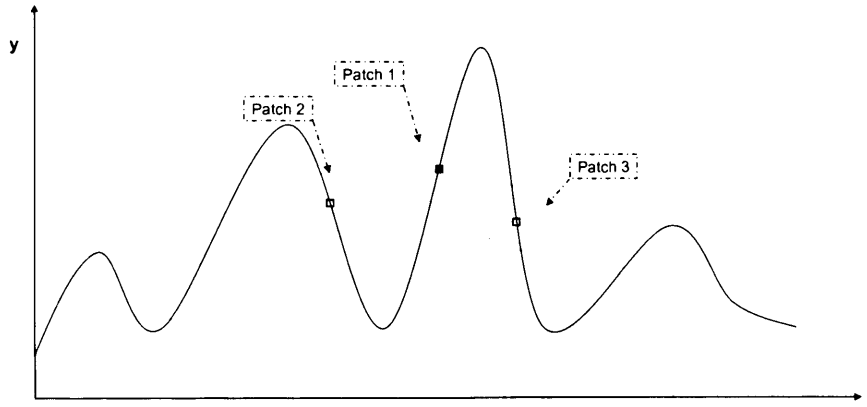


Graph 1: Initialise a population of (n=10) scout bees with random search and evaluate the fitness

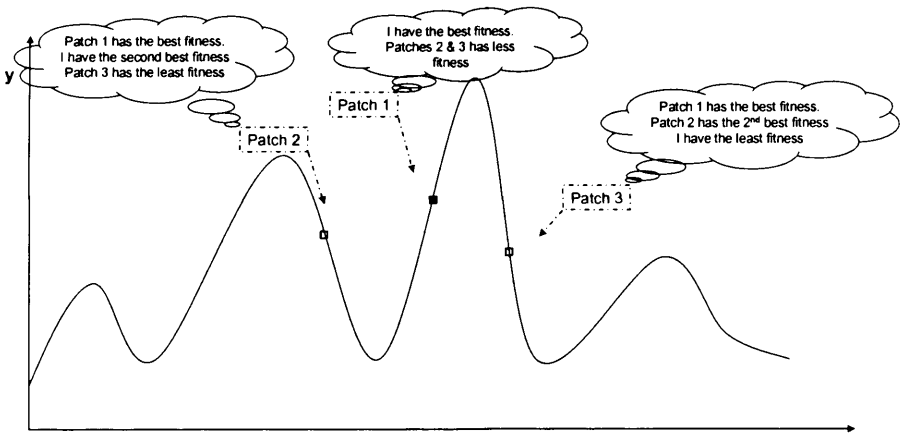


Graph 2: Select best (m=3) sites for neighbourhood search:
the best e=1 sites "▪" and (m – e = 2) other selected sites"▫"

Figure 4.5: Operation of the improved Bees Algorithm (To be continued)

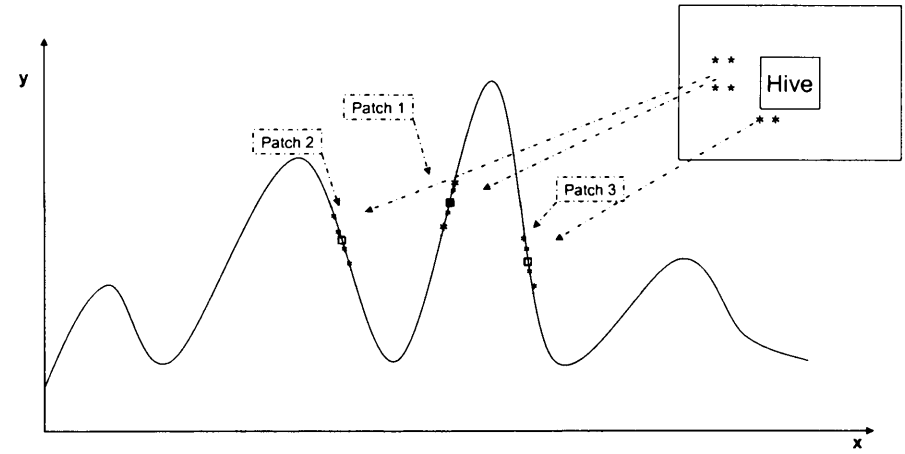


Graph 3: Select patches for adaptive neighbourhood search (more weights to patches with better solution

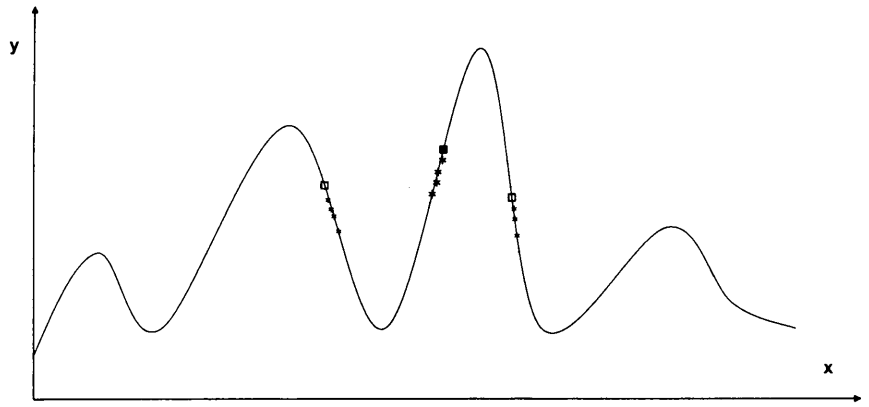


Graph 4: Propagate global information of best known patch across entire swarm

Figure 4.5: Operation of the improved Bees Algorithm (To be continued)

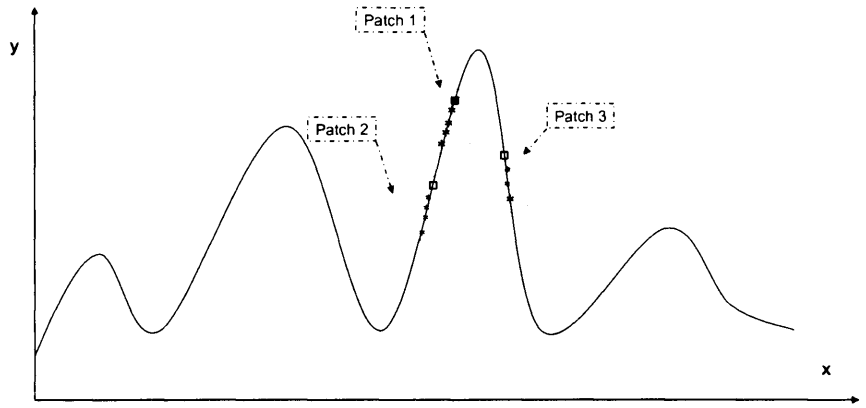


Graph 5: According to the globally shared information, recruit bees for the selected patches (more bees for patches with more weight)

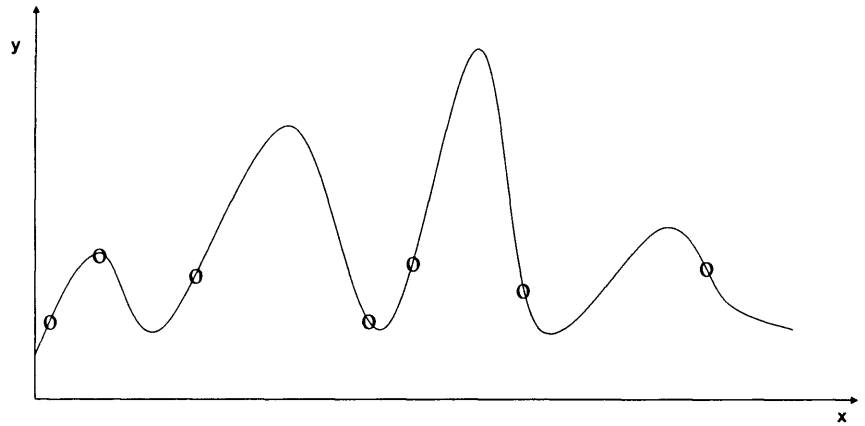


Graph 6: Select the fittest bee " * " from each site

Figure 4.5: Operation of the improved Bees Algorithm (To be continued)

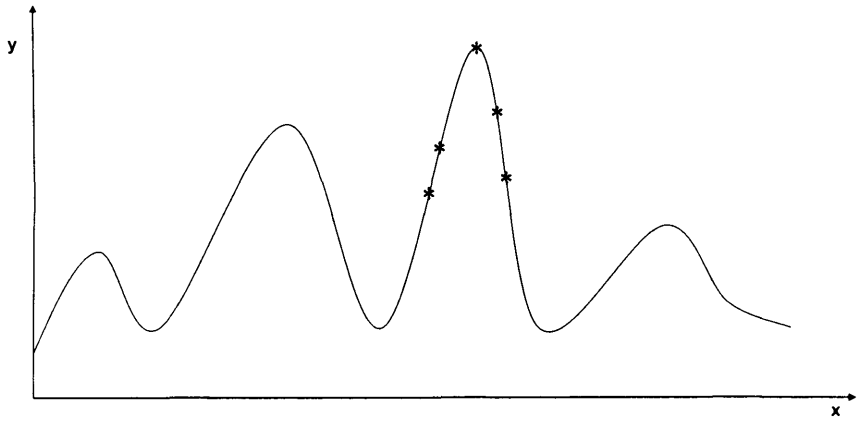


Graph 7: Patches 2 & 3 are attracted to Patch 1; Patch 3 now becomes the 2nd best; Patch 2 now has the least fitness



Graph 8: Assign the (n–m) remaining bees to random search

Figure 4.5: Operation of the improved Bees Algorithm (To be continued)



Graph 9: Global optimum is found

Figure 4.5: Operation of the improved Bees Algorithm (Cont'd)

4.3 Results

This section presents the results of four different applications of the *improved* Bees Algorithm. The algorithm is applied to three standard mechanical design optimisation problems: the design of a welded beam structure (single objective), welded beam structure (multi objective), and the design of coil springs. These three applications are used to benchmark the *improved* Bees Algorithm against other optimisers. The welded beam design problem entails a non-linear objective function with eight constraints; whilst the design of coil spring problem is also a non-linear objective function having four constraints. The section starts with the application of the *improved* Bees Algorithm to the welded beam (single objective) with results presented; next a multi-objective version of

the design of welded beam problem is tackled by the *improved* Bees Algorithm and the results are again shown. Later, the algorithm is applied to the design of coil springs and the results obtained are presented. Finally, the algorithm is tested on mathematical benchmark problems shown in Table 3.6 of Chapter 3 and the presentation of the results concludes the chapter.

4.3.1 Application to Mechanical Design Optimisation - Welded Beam Design Problem

One of the benchmark problems used to test optimisation algorithms is the standard mechanical design problem, the design of the well-known welded beam structures (Rekliatis *et al.* 1983). The welded beam design problem encompasses a non-linear objective function with eight constraints. Previously, a number of optimisation methods were tested on this design problem. Afshin (Ghanbarzadeh 2007) used the Bees Algorithm. Ragsdell and Phillips (Ragsdell and Phillips 1976) implemented geometric programming that required extensive problem formulation while that employed by Leite and Topping (Leite and Topping 1998) used specific domain knowledge which may not be available for other problems. The work by Ragsdell and Phillips (Ragsdell and Phillips 1976) was found to be computationally expensive or gave poor results.

A uniform beam of rectangular cross section needs to be welded to a base to carry a load of 6000 *lbf*. The design is shown in Figure 4.6. The beam is made of steel 1010. The length L is specified as 14 *in*. The intention of the design is to minimise the cost of fabrication while finding a feasible combination of weld thickness h, weld length l, beam

thickness t and beam width b. The objective function (Rekliatis *et al.* 1983) is formulated as:

$$\text{Min } f = (1+c_1)h^2 l + c_2 tb(L+l) \tag{4.2}$$

where

f = Cost function including setup cost, welding labour cost and material cost;

c_1 = Unit volume of weld material cost = 0.10471 $\$/in.^3$;

c_2 = Unit volume of bar stock cost = 0.04811 $\$/in.^3$;

L = Fixed distance from load to support = 14 *in.*;

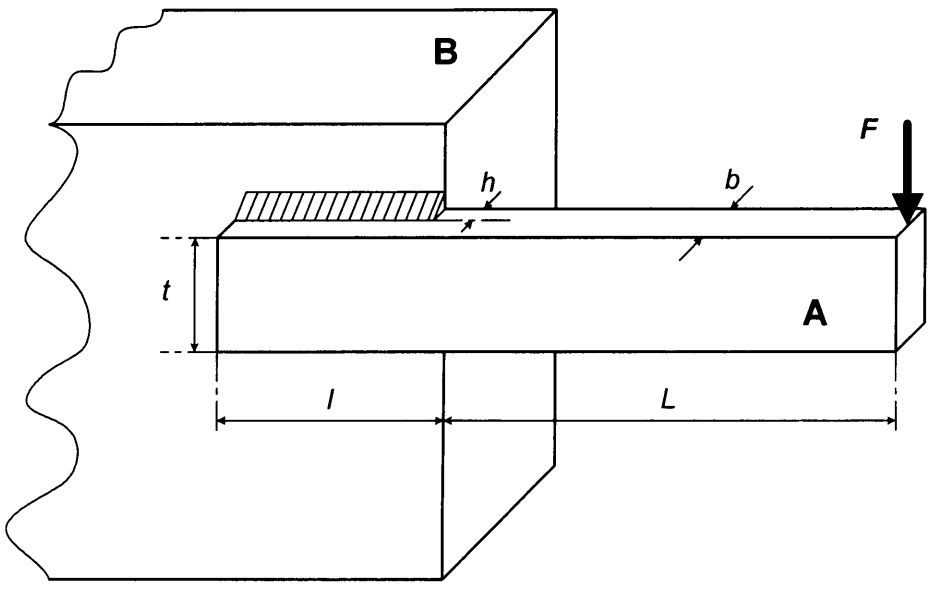


Figure 4.6: A welded beam

Because of the existence of limitations that need to be taken into account concerning the mechanical properties of the weld and bar such as the shear and normal stresses, physical constraints (no length less than zero) and the maximum deflection, not all the combinations of h, l, t and b that can support F are satisfactory within the acceptable limits.

From (Rekliatis *et al.* 1983), these constraints are defined as follows:

$$g_1 = \tau_d - \tau \geq 0 \tag{4.3}$$

$$g_2 = \sigma_d - \sigma \geq 0 \tag{4.4}$$

$$g_3 = b - h \geq 0 \tag{4.5}$$

$$g_4 = l \geq 0 \tag{4.6}$$

$$g_5 = t \geq 0 \tag{4.7}$$

$$g_6 = P_c - F \geq 0 \tag{4.8}$$

$$g_7 = h - 0.125 \geq 0 \tag{4.9}$$

$$g_8 = 0.25 - \delta \geq 0 \tag{4.10}$$

where

τ_d = Allowable shear stress of weld $= 13600\ Psi$;

τ = Maximum shear stress in weld;

σ_d = Allowable normal stress for beam material $= 30000\ Psi$;

σ = Maximum normal stress in beam;

P_c = Bar buckling load;

F = Load = 6000 lbf;

δ = Beam end deflection.

Table 4.1 below shows the properties of the constraints g_1 to g_8.

g_1	ensures that the maximum developed shear stress is less than the allowable shear stress of the weld material.
g_2	checks that the maximum developed normal stress is lower than the allowed normal stress in the beam.
g_3	ensures that the beam thickness exceeds that of the weld.
g_4 and g_5	are practical checks to prevent negative lengths or thickness.
g_6	makes sure that the load on the beam is not greater than the allowable buckling load.
g_7	checks that the weld thickness is above a given minimum.
g_8	is to ensure that the end deflection of the beam is less than a predefined amount.

Table 4.1: Properties of constraints g_1 to g_8

From (Rekliatis *et al.* 1983; Shigley 1977), the normal and shear stresses and the buckling force are formulated as:

$$\sigma = \frac{2.1952}{t^3 b} \tag{4.11}$$

$$\tau = \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}} \tag{4.12}$$

where

$$\tau' = \frac{6000}{\sqrt{2}hl} \quad \text{(Primary Stress)} \tag{4.13}$$

$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\left\{0.707hl\left(l^2/12 + 0.25(h+t)^2\right)\right\}} \quad \text{(Secondary Stress)} \tag{4.14}$$

$$P_c = 64746.022(1 - 0.0282346t)tb^3 \tag{4.15}$$

Table 4.2 below shows the parameters used by the *improved* Bees Algorithm for the welded beam design problem with stopping criterion of 750 generations. For comparison, the parameters are same as those used by (Pham *et al.* 2008).

***Improved* Bees Algorithm parameters**	**Symbol**	**Value**
Population	*n*	80
Number of selected sites	*m*	5
Number of top-rated sites out of m selected sites	*e*	2
Initial patch size	*ngh*	0.1
Number of bees recruited for best e sites	*nep*	50
Number of bees recruited for the other (*m-e*) selected sites	*nsp*	10

Table 4.2: Parameters of the *improved* Bees Algorithm for the welded beam design problem

From (Deb 1991), the search space is defined with explicit bounds:

$$0.125 \le h \le 5 \tag{4.16}$$

$$0.1 \le l \le 10 \tag{4.17}$$

$$0.1 \le t \le 10 \tag{4.18}$$

$$0.1 \le b \le 5 \tag{4.19}$$

With equations (4.16) to (4.19), the constraints g_4, g_5 and g_7 are already satisfied and does not need to be checked in the code. Figure 4.7 shows how the lowest value of the objective function changes with the number of iterations (generations) for three independent runs of the algorithm. It can be seen that the objective function decreases rapidly in the early iterations and then gradually converges to the optimum value.

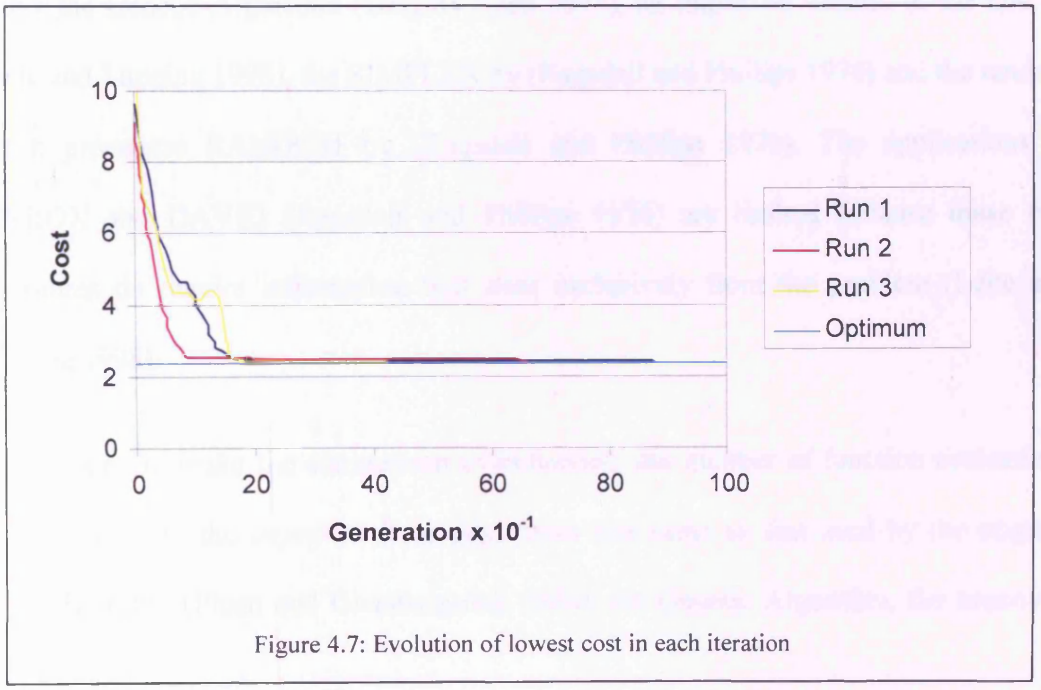


Figure 4.7: Evolution of lowest cost in each iteration

An array of optimisation methods have previously been applied to this problem by other researchers (Deb 1991; Leite and Topping 1998; Pham *et al.* 2008; Ragsdell and Phillips 1976). The results of these other optimisation methods together with that of the *improved* Bees Algorithm are given in Table 4.3. APPROX is a method of successive linear approximation (Siddall 1972). DAVID is a gradient method with a penalty (Siddall 1972). Geometric Programming (GP) is a method capable of solving linear and nonlinear optimisation problems that are formulated analytically (Ragsdell and Phillips 1976). SIMPLEX is the Simplex algorithm for solving linear programming problems (Siddall 1972).

As shown in Table 4.3, the *improved* Bees Algorithm produced better result compared with the listed algorithms including the original Bees Algorithm (BA) by (Pham *et al.* 2008), the Genetic Algorithm (GA) by (Deb 1991), an improved version of the GA by (Leite and Topping 1998), the SIMPLEX by (Ragsdell and Phillips 1976) and the random search procedure RANDOM by (Ragsdell and Phillips 1976). The applications of APPROX and DAVID (Ragsdell and Phillips 1976) are limited because these two algorithms do require information that stem exclusively from the problem (Leite and Topping 1998).

Furthermore, to make the comparison even-handed, the number of function evaluations implemented by the *improved* Bees Algorithm was same as that used by the original Bees Algorithm (Pham and Ghanbarzadeh 2006), the Genetic Algorithm, the improved

GA, the APPROX and DAVID technique, the SIMPLEX method and the random search procedure RANDOM.

The improved Bees Algorithm used less information to find the best result.

Methods	Design variables				Cost
	h	*l*	*t*	*b*	
APPROX (Ragsdell and Phillips 1976)	0.2444	6.2189	8.2915	0.2444	2.38
DAVID (Ragsdell and Phillips 1976)	0.2434	6.2552	8.2915	0.2444	2.38
GP (Ragsdell and Phillips 1976)	0.2455	6.1960	8.2730	0.2455	2.39
GA (Deb 1991) **Three independent runs**	0.2489	6.1730	8.1789	0.2533	2.43
	0.2679	5.8123	7.8358	0.2724	2.49
	0.2918	5.2141	7.8446	0.2918	2.59
IMPROVED GA (Leite and Topping 1998) **Three independent runs**	0.2489	6.1097	8.2484	0.2485	2.40
	0.2441	6.2936	8.2290	0.2485	2.41
	0.2537	6.0322	8.1517	0.2533	2.41
SIMPLEX (Ragsdell and Phillips 1976)	0.2792	5.6256	7.7512	0.2796	2.53
RANDOM (Ragsdell and Phillips 1976)	0.4575	4.7313	5.0853	0.6600	4.12
BEES ALGORITHM Three independent runs (Pham and Ghanbarzadeh 2006)	0.24429	6.2126	8.3009	0.24432	2.3817
	0.24428	6.2110	8.3026	0.24429	2.3816
	0.24432	6.2152	8.2966	0.24435	2.3815
***Improved* BEES ALGORITHM Three independent runs**	0.24427	6.2131	8.3012	0.24431	2.381738
	0.24426	6.2019	8.3180	0.24401	2.381437
	0.24429	6.2122	8.3019	0.24426	2.381421

Table 4.3: Comparison of results of the *improved* Bees Algorithm on welded beam design problem with other optimisers

A lingering question persists 'what is the statistical significance of the result presented in Table 4.3'?

To check the statistical significance of the result, a T-TEST had to be performed which checks the relationship between two variables, in this case two different algorithms.

The T-Test was conducted between the *improved* Bees Algorithm and the original Bees Algorithm. As mentioned earlier, both algorithms were applied 30 times to the welded beam design problem.

Figure 4.8 shows a plot of the minimum cost produced by both algorithms. The values of the plot are presented in Tables 4.4 and 4.5 for the *improved* Bees Algorithm and the original Bees Algorithm respectively.

2.381738	2.381437	2.381421	2.381435	2.381481
2.381441	2.381411	2.381571	2.381421	2.381411
2.381431	2.381451	2.381491	2.381411	2.381541
2.381431	2.381411	2.381429	2.381481	2.381491
2.381441	2.381451	2.381471	2.381427	2.381461
2.381471	2.381431	2.381451	2.381411	2.381421

Table 4.4: Minimum cost obtained by the *improved* Bees Algorithm for the welded beam design problem

2.3817	2.3816	2.3815	2.3819	2.38147
2.3815	2.38146	2.3816	2.38144	2.38156
2.3817	2.3815	2.38144	2.38154	2.38147
2.38157	2.38152	2.3816	2.38145	2.3815
2.38146	2.38153	2.38148	2.38152	2.38155
2.38148	2.3815	2.3815	2.38146	2.38157

Table 4.5: Minimum cost obtained by the original Bees Algorithm for the welded beam design problem

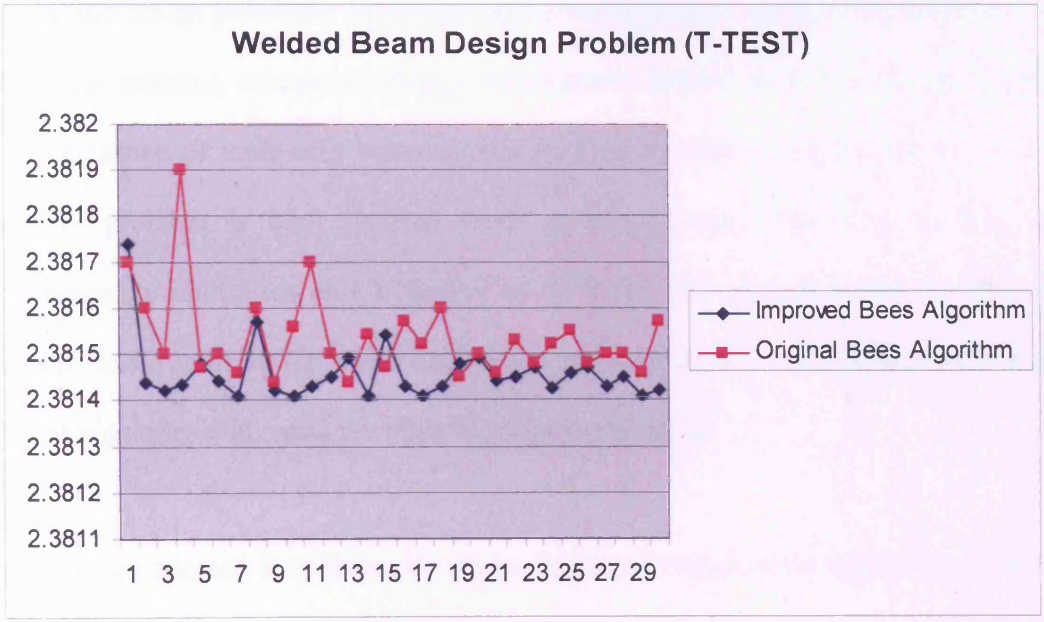


Figure 4.8: Plot of the minimum cost obtained by the *improved* Bees Algorithm and the original Bees Algorithm for welded beam design problem

I obtained an alpha value of $0.000626213984 \approx 0.00063$ from the T-Test. This value indicates that the result obtained by both the *improved* Bees Algorithm and the original Bees Algorithm is most significantly different with a confidence level above 99%.

4.3.2 Application to Multi-Objective Optimisation - Welded Beam Design Problem

From Wikipedia, multi-objective optimisation also known as multi-criteria or multi-attribute optimisation is the process of simultaneously optimising two or more conflicting objectives subject to certain constraints.

Today, multi-objective optimisation problems are found in various fields, for instance in engineering design problems, in product and process design, finance, aircraft design, the oil and gas industry, automobile design, or wherever optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. If a multi-objective problem is well defined there is usually more than one solution that simultaneously minimises each objective to its fullest. For each objective function, the aim is to find a solution for which each objective is optimised to the point where further efforts to optimise will cause the other objective(s) to suffer.

The approach adopted for solving the multi-objective version of the welded beam design problem is to simultaneously consider all objective functions. In a multi-objective optimisation task, the goal is not to find a single optimal solution, but instead to compute the set of all non-dominated solutions, that is, the Pareto optimal set. A solution belonging to the Pareto set is not better than any other solution belonging to the same set.

For this reason, they are not comparable and each of them is called a *feasible solution.* Different techniques to solve multi-objective optimisation tasks and their characteristics are explained in (Deb 1991).

Pareto efficiency (also called Pareto optimality) is an important notion in neoclassical economics with broad applications in game theory, engineering and the social sciences (Fudenberg and Tirole 1991). It defines the frontier of solutions that can be reached by trading-off conflicting objectives in an optimal manner. Thus, a decision maker (either human or an algorithm) can finally choose the configurations that, in his / her opinion, suites best (Chankong and Haimes 1983; Galperin 1997; Steuer 1989). The notation of optimal solution in the sense of Pareto efficiency is strongly based on the definition of *domination.*

Domination: An element x_1 dominates (is preferred to) an element x_2 ($x_1 \vdash x_2$) if x_1 is better than x_2 in at least one objective function and not worse with respect to all other objectives. Based on the set F of objective functions f, it is sufficient to write:

$$x_1 \vdash x_2 \Leftrightarrow \forall i : 0 < i \leq n \Rightarrow \omega_i f_i(x_1) \leq \omega_i f_i(x_2) \wedge \exists j : 0 < j \leq n : \omega_j f_j(x_1) < \omega_j f_j(x_2) \tag{4.20}$$

$$\omega_i = \begin{cases} 1 & \text{if } f_i \text{ should be minimised} \\ -1 & \text{if } f_i \text{ should be maximised} \end{cases} \tag{4.21}$$

The factor 'ω' only carries the sign information which allows the maximisation and the minimisation of objective functions while the Pareto domination relation (4.19) defines a strict partial order in the space of possible objective values (Weise 2008). In contrast, the weighted sum approach imposes a total order by projecting it onto the real numbers $\mathbb{R}$.

Pareto Optimal: An element $x^* \in X$ is Pareto optimal (and hence, part of the optimal set X^*) if it is not dominated by any other element in the problem space X. In terms of Pareto optimisation, X^* is called the Pareto set or the Pareto Frontier.

$$x^* \in X^* \Leftrightarrow \exists x \in X : x \vdash x^* \tag{4.22}$$

Problems of Pure Pareto Optimisation

The complete Pareto optimal set is not often the wanted result of an optimisation algorithm. Instead only some special areas of the Pareto front are crucial.

The application of the *improved* Bees Algorithm to the multi-objective version of the design of welded beam is identical to that discussed in the previous section. The only difference is the objective function as defined below by (Rekliatis *et al.* 1983).

$$\text{Min } f_1 = (1+c_1)h^2 l + c_2 tb(L+l) \tag{4.23}$$

$$\text{Min } f_2 = \delta \tag{4.24}$$

Constraint g_8 is converted into a fitness function.

Table 4.6 shows the parameters of the *improved* Bees Algorithm used to solve the multi-objective version of the welded beam design problem. For ease of comparison, these parameters are same as that used by (Pham and Ghanbarzadeh 2007).

***Improved* Bees Algorithm parameters**	**Symbol**	**Value**
Population	*n*	150
Number of selected sites	*m*	30
Initial patch size	*ngh*	0.1
Number of bees recruited for selected sites	*nsp*	50
Number of iterations	*n_iter*	1000

Table 4.6 Parameters of the *improved* Bees Algorithm for multi-objective welded beam design problem

Result of Multi-Objective welded beam design problem

Figure 4.9 shows the non-dominated solutions obtained by the *improved* Bees Algorithm. The total number is 229 non-dominated solutions distributed along the Pareto front. Deb investigated this problem using the non-dominated sorting GA (or NSGA) and a fast elitist NSGA, called NSGA-II (Deb *et al.* 2000) for finding multiple Pareto optimal solutions (Figure 4.10b).

The *improved* Bees Algorithm found more non-dominated solutions in comparison to those by the non-dominated sorting genetic algorithms and the original Bees Algorithm. From (Deb *et al.* 2000), the NSGA-II established the best cost solution with a cost of 2.79 units. Unlike the multi-objective original Bees Algorithm that obtained a quantity of 2.39

units cost, the multi-objective *improved* Bees Algorithm found a better quantity of 2.38371 units cost, which again is closer to the best solution obtained by the single objective *improved* Bees Algorithm (with a cost of 2.381421 units).

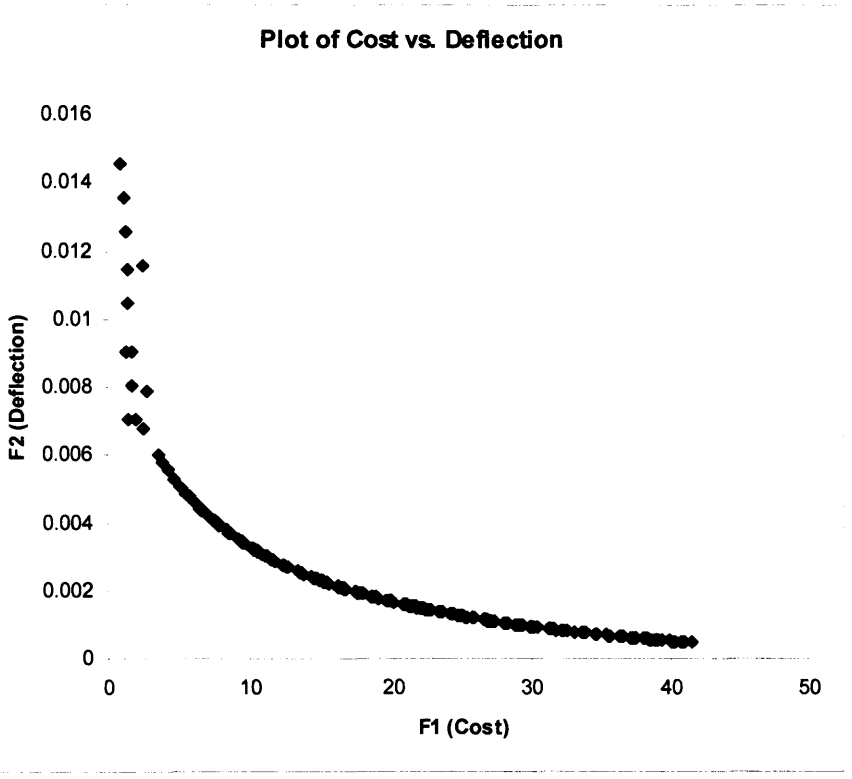


Figure 4.9: Non-dominated solutions obtained using the *improved* Bees Algorithm

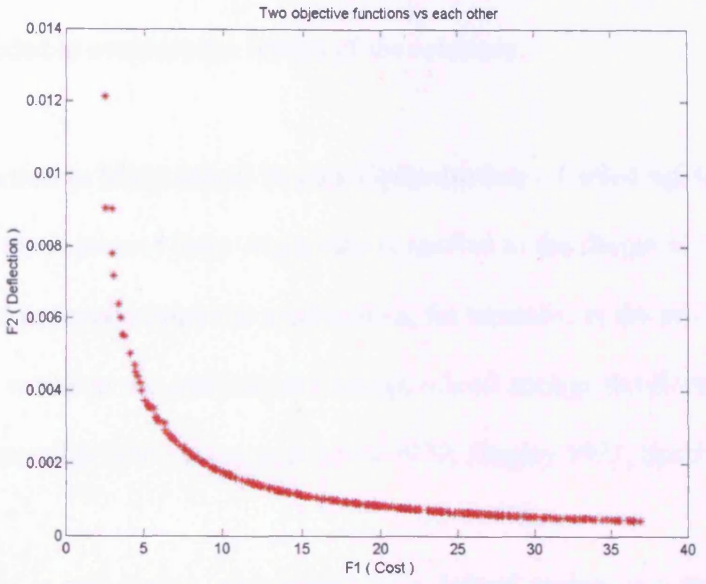


Figure 4.10a: Non-dominated solutions obtained using the original Bees Algorithm (Ghanbarzadeh 2007)

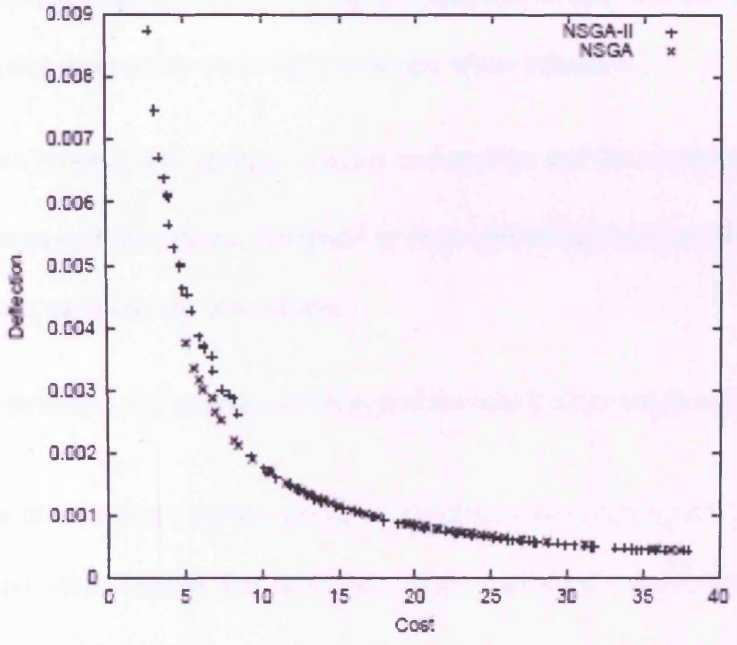


Figure 4.10b: Non-dominated solutions obtained using the two different versions of genetic algorithms (Deb *et al.* 2000)

The *improved* Bees Algorithm is competent to decipher multi-solution and multi-objective function optimisation problems with no prior domain knowledge except the information needed to evaluate the fitness of the solutions.

4.3.3 Application to Mechanical Design Optimisation – Coiled Spring Problem

In this section, the *improved* Bees Algorithm is applied to the design of coil springs. Coil springs are used in several practical applications, for instance, in the automotive industry. Some previous works in the analysis and design of coil springs developed over the years include (Ghanbarzadeh 2007; Haug and Arora 1979; Shigley 1977; Spotts 1971).

From Wikipedia, a coil spring, also known as a *helical spring*, is a mechanical device used to store energy and subsequently releases it to absorb shock or to maintain a force between contacting surfaces. Coil springs are made of an elastic material formed into the shape of a helix that returns to its natural length when unloaded.

There are two types of coil springs: tension coil springs and the compression coil springs.

- Tension coil springs are designed to resist stretching and they have a hook or eye form at each end for attachment.
- Compression coil springs are designed to resist being compressed.

The purpose is to design a coiled spring of minimum mass shown in Figure 4.11, to carry a given axial load without any material failure and at the same time satisfying two performance requirements:

- the spring must deflect by at least Δ (in.)

- the frequency of surge waves must not be less than ω_0 (Hertz, Hz).

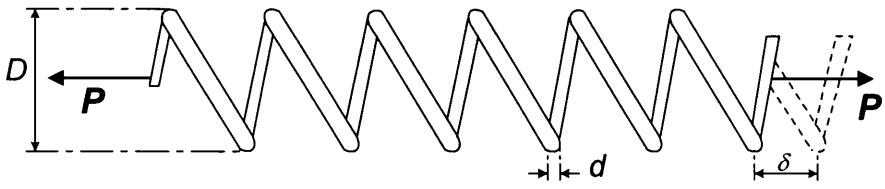


Figure 4.11: A coil spring

The three design variables to be optimised are the wire diameter d, the mean coil diameter D and the number of active coils N.

The intention of applying the *improved* Bees Algorithm to the design of coil spring is to minimise the mass of the spring M, given as the product of the volume and mass density as defined in (Arora 2004), shown explicitly in Equation 4.25.

$$M = \frac{1}{4}(N+Q)\pi^2 D d^2 \rho \tag{4.25}$$

The list of constraints as formulated by (Arora 2004) includes:

Deflection limit: $g_9 = \Delta - \frac{8PD^3N}{d^4G} \leq 0$ (4.26)

Shear stress: $g_{10} = \frac{8PD}{\pi d^3}\left[\frac{(4D-d)}{4(D-d)} + \frac{0.615d}{D}\right] - \tau_d \leq 0$ (4.27)

Frequency of surge waves: $g_{11} = \omega_0 - \omega \leq 0$ (4.28)

Diameter constraint: $g_{12} = D + d - D_0 \leq 0$ (4.29)

The notations used to formulate the problem of designing the coil spring are listed below in Table 4.7:

Deflection along the axis of the spring	δ, in.
Mean coil diameter	D, in.
Wire diameter	d, in.
Number of active coils	N
Gravitational constant	$g = 386$ in./s^2
Frequency of surge waves	$\omega = \frac{d}{2\pi ND^2}\sqrt{\frac{G}{2\rho}}$
Material properties:	
Shear modulus	$G = (1.15 \times 10^7)$ lb/in.^{2}
Weight density of spring material	$\gamma = 0.285$ lb/in.^{3}
Mass density of material ($\rho = \gamma / g$)	$\rho = (7.38342 \times 10^{-4})$ lb-s^2/in.^{4}
Allowable shear stress	$\tau_d = 80{,}000$ lb/in.^{2}
Other information:	
Number of inactive coils	$Q = 2$
Applied load	$P = 10$ lb
Minimum spring deflection	$\Delta = 0.5$ in.
Lower limit on surge wave frequency	$\omega_0 = 100$ Hz
Limit on outer diameter of the coil	$D_0 = 1.5$ in.

Table 4.7: Notations used to formulate the problem of designing the coil spring

Using the information in Table 4.7, the above constraints ($g_9 - g_{12}$) are rewritten as:

Deflection limit: $g_9 = 1.0 - \frac{D^3 N}{71875 d^4} \leq 0$ (4.30)

Shear stress: $g_{10} = \frac{D(4D-d)}{12566 d^3 (D-d)} + \frac{2.46}{12566 d^2} - 1.0 \leq 0$ (4.31)

Frequency of surge waves: $g_{11} = 1.0 - \frac{140.54 d}{D^2 N} \leq 0$ (4.32)

Diameter constraint: $g_{12} = \frac{D+d}{1.5} - 1.0 \leq 0$ (4.33)

The properties of these constraints are given next in Table 4.8.

g_9	Ensures the deflection of the coil spring is greater than the specified minimum value.
g_{10}	Verifies the maximum shear stress in the coil spring is less than the allowable shear stress.
g_{11}	Verifies the frequency of surge waves is greater than the given lower limit.
g_{12}	Regulates the outer diameter of the spring.

Table 4.8: Properties of constraints

In order to make the comparison even-handed, the parameters used by the *improved* Bees Algorithm shown in Table 4.9 below are the same as that used by (Pham and Ghanbarzadeh 2006).

Improved Bees Algorithm parameters	Symbol	Value
Population	*n*	60
Number of selected sites	*m*	5
Number of top-rated sites out of m selected sites	*e*	2
Initial patch size	*ngh*	0.1
Number of bees recruited for best *e* sites	*nep*	40
Number of bees recruited for the other (*m-e*) selected sites	*nsp*	10

Table 4.9: The *improved* Bees Algorithm parameters

Explicit bounds (minimum and maximum size limits of the wire, coil diameter and number of turns) on design variables were introduced to avoid fabrication and other practical difficulties. They are listed in equations (4.34) to (4.36).

$$0.05 \leq d \leq 0.2 \tag{4.34}$$

$$0.25 \leq D \leq 1.3 \tag{4.35}$$

$$2 \leq N \leq 15 \tag{4.36}$$

Figure 4.12 below shows the evolution of the best value of the objective function with the number of iterations.

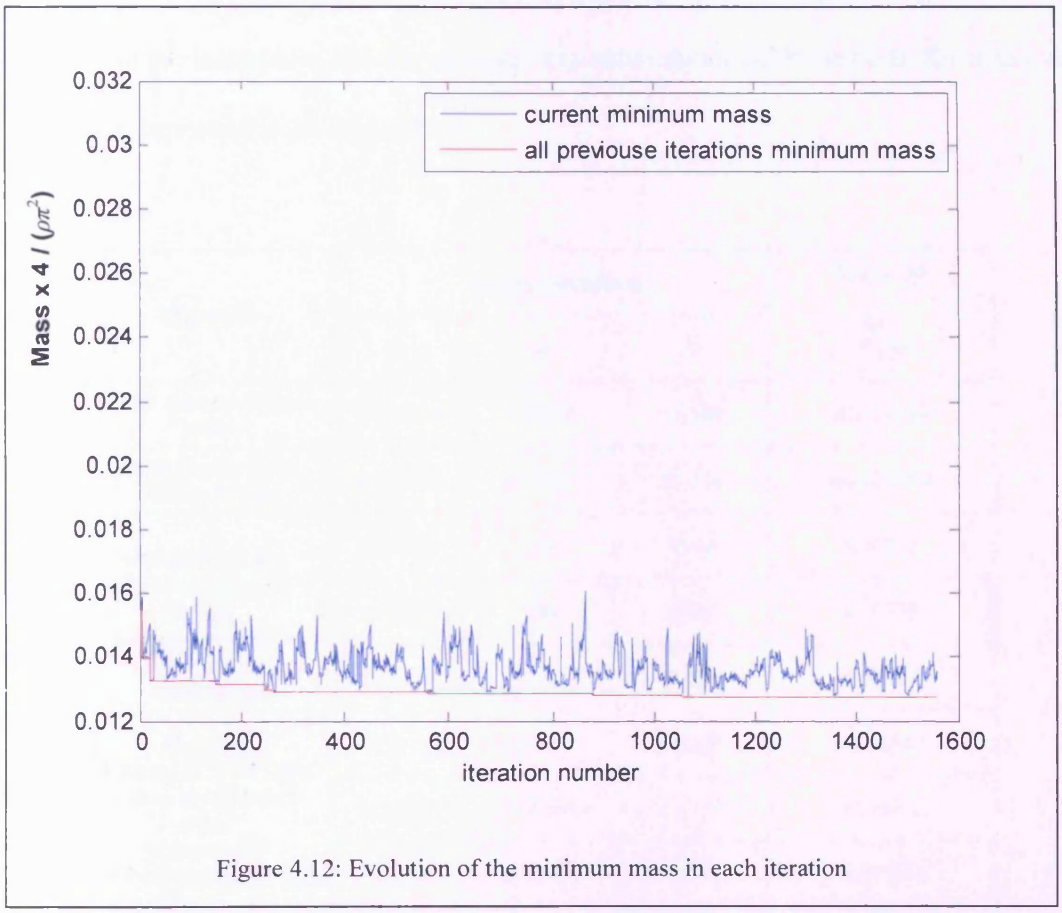


Figure 4.12: Evolution of the minimum mass in each iteration

Preceding this application of the *improved* Bees Algorithm to the problem of the design of coiled spring, the original Bees Algorithm (Pham and Ghanbarzadeh 2006), the Sequential Quadratic Programming (SQP) methods in a batch environment & in an interactive mode (Arora 2004) in addition to the improved Genetic Algorithm (Leite and Topping 1998) had earlier been implemented. The results obtained by these earlier optimisers are presented in Table 4.10 together with the result of three independent runs performed by the *improved* Bees Algorithm.

Table 4.10 below show that the *improved* Bees Algorithm produced superior results compared to the interactive solution process, the batch–mode SQP methods, the improved GA and the *improved* Bees Algorithm.

Methods	**Design variables**			**Mass M** $\left(\times 4/\rho\pi^2\right)$
	d	D	N	
SQP (batch) (Arora 2004)	0.051699	0.35695	11.289	0.0126787
SQP (interactive) (Arora 2004)	0.05340	0.3992	9.1854	0.0127300
IMPROVED GA (Leite and Topping 1998) **Best three solutions not violating constraints**	0.05235	0.3721	10.48	0.01272
	0.05323	0.3947	9.383	0.01273
	0.05396	0.4132	8.697	0.01287
Original BEES ALGORITHM Three independent runs (Pham and Ghanbarzadeh 2006)	0.051759	0.35839	11.207	0.012680
	0.051807	0.35956	11.139	0.012680
	0.051779	0.35886	11.179	0.012681
Improved BEES Algorithm Three independent runs	0.051544	0.353238	11.511	0.012679756
	0.051855	0.360722	11.072	0.012679315
	0.051852	0.360651	11.076	0.012679234

Table 4.10: Comparison of the *improved* Bees Algorithm results with other optimisers

A question persists 'what is the statistical significance of the result presented in Table 4.10?'

To check the statistical significance of the result, a T-TEST had to be performed which checks the relationship between two variables, in this case two different algorithms.

The T-Test was conducted between the *improved* Bees Algorithm and the original Bees Algorithm. The two algorithms were applied 30 times to the design of the coil spring problem.

Figure 4.13 shows a plot of the minimum mass produced by both algorithms. The values of the plot are presented in Tables 4.11 and 4.12 for the *improved* Bees Algorithm and the original Bees Algorithm respectively.

0.012679756	0.012679315	0.0126792	0.0126797	0.0126794
0.012679334	0.012679515	0.0126797	0.0126796	0.0126793
0.012679415	0.012679556	0.0126793	0.0126792	0.0126793
0.012679715	0.012679338	0.0126795	0.0126794	0.0126792
0.012679434	0.012679306	0.0126795	0.0126795	0.0126793
0.012679237	0.012679382	0.0126795	0.0126792	0.0126793

Table 4.11: Minimum mass produced by the *improved* Bees Algorithm for the design of coil spring problem

0.01268	0.01268	0.0126805	0.0126801	0.01268
0.0126802	0.0126804	0.0126806	0.0126803	0.0126809
0.0126805	0.0126807	0.0126807	0.0126809	0.0126806
0.0126808	0.0126801	0.012681	0.0126801	0.01268
0.012682	0.0126809	0.0126808	0.012681	0.0126811
0.0126802	0.0126809	0.0126802	0.0126807	0.01268

Table 4.12: Minimum mass produced by the original Bees Algorithm for the design of coil spring problem

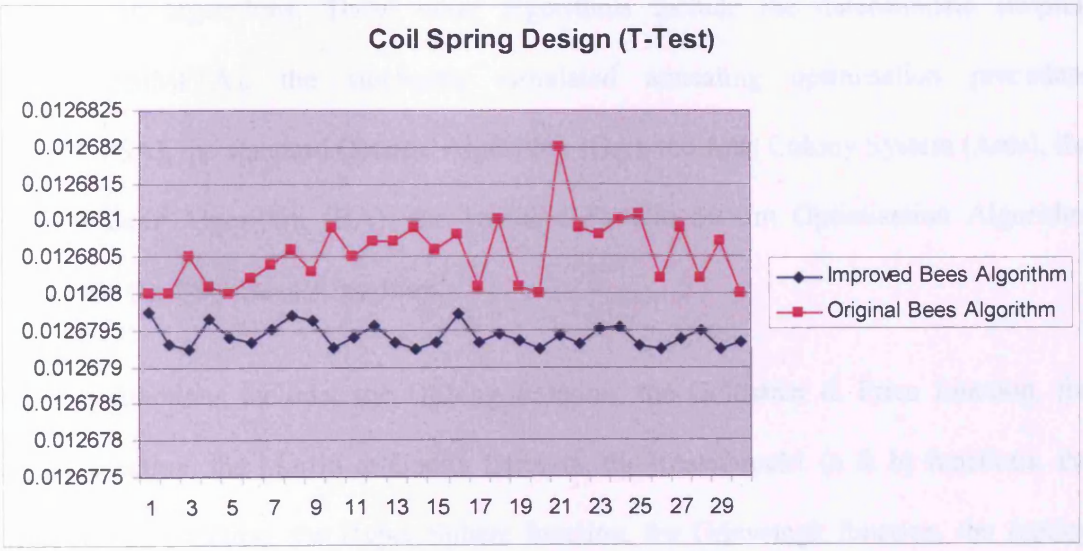


Figure 4.13: Plot of the minimum mass produced by the *improved* Bees Algorithm and the original Bees Algorithm for the design of coil spring problem

I obtained an alpha value of 2.15804E-18 from the T-Test. This value indicates that the results obtained by both the *improved* Bees Algorithm and the original Bees Algorithm are significantly different with a confidence level above 99%.

4.3.4 Tests on Mathematical Benchmark Functions / Comparison with Other Global Optimisation Algorithms

The work presented in this section is a continuation of Section 3.6.4 of Chapter 3 involving tests on mathematical benchmark functions. Ten standard tests on function optimisation problems were used to examine the *improved* Bees Algorithm as an effective global optimiser and the results obtained are compared with other global optimisation algorithms. These other algorithms include the deterministic simplex method (SIMPSA), the stochastic simulated annealing optimisation procedure (NESIMPSA), the standard Genetic Algorithm (GA), the Ants Colony System (Ants), the original Bees Algorithm (BA), the standard Particle Swarm Optimisation Algorithm (PSO) and the PSO-Bees Algorithm.

The test functions include: the DeJong function, the Goldstein & Price function, the Branin function, the Martin & Gaddy function, the Rosenbrock1 (a & b) functions, the Rosenbrock2 function, the Hyper Sphere function, the Griewangk function, the Ackley function and the Schwefel function. Table 4.13, presented below show the properties of the test functions, interval and their global optimum while Table 4.14 presents the results obtained by the *improved* Bees Algorithm for 100 independent runs in comparison with the results from other previously applied optimisers.

No	Reference	Interval	Test Function	Global Optimum
1	De Jong	[-2.048, 2.048]	$\max F = (3905.93) - 100\,(x_1^2 - x_2^2) - (1 - x_1)^2$	X[1,1] F=3905.93
2	Goldstein & Price	[-2, 2]	$\min F = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $X[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	X[0,-1] F=3
3	Branin	[-5, 10]	$\min F = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e$ $a = 1, b = \frac{5.1}{4}\left(\frac{7}{22}\right)^2, c = \frac{5}{22} X 7, d = 6, e = 10, f = \frac{1}{8} X \frac{7}{22}$	X[-22/7,12.275] X[22/7,2.275] X[66/7,2.475] F=0.3977272
4	Martin & Gaddy	[0, 10]	$\min F = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$	X[5,5] F=0
5	Rosenbrock -1	(a) [-1.2, 1.2] (b) [-10, 10]	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	X[1,1] F=0
6	Rosenbrock - 2	[-1.2, 1.2]	$\min F = \sum_{i=1}^{3} \{100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\}$	X[1,1,1,1] F=0
7	Hyper sphere model	[-5.12, 5.12]	$\min F = \sum_{i=1}^{6} x_i^2$	X[0,0,0,0,0,0] F=0
8	Griewangk	[-512, 512]	$\max F = \frac{1}{0.1 + \left(\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)}$	X[0,0,0,0,0,0,0,0,0,0] F=10
9	Ackley	[-5.12, 5.12]	$f(x) = 20 + e - 20\,e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}$	X [0, …, 0] F=0
10	Schwefel	[-500, 500]	$f(x) = 418.9829 \cdot n - \sum_{i=1}^{n} (-x_i \sin(\sqrt{\lvert x_i \rvert}))$	X [1, …, 1] F=0

Table 4.13: Test Functions (Mathur *et al.* 2000)

func no	SIMPSA		NE SIMPSA		GA		ANT		Bees Algorithm		PSO Algorithm		PSO-Bees Algorithm		*Improved* Bees Algorithm	
	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals
1	***	****	****	****	100	10160	100	6000	100	868	100	872	100	815	100	806
2	***	****	****	****	100	5662	100	5330	100	999	100	1008	100	879	100	851
3	***	****	****	****	100	7325	100	1936	100	1657	100	1594	100	1463	100	1387
4	***	****	****	****	100	2844	100	1688	100	526	100	507	100	486	100	462
5a	100	10780	100	4508	100	10212	100	6842	100	631	100	689	100	594	100	573
5b	100	12500	100	5007	***	****	100	7505	100	2306	100	2281	100	1829	100	1794
6	99	21177	94	3053	***	****	100	8471	100	28529	100	27736	100	21105	100	20729
7	***	****	****	****	100	15468	100	22050	100	7113	100	6930	100	6794	100	6485
8	***	****	****	****	100	200000	100	50000	100	1847	100	1891	100	1798	100	1671
9	***	***	***	****	***	****	***	****	***	****	100	2247	100	1979	100	1829
10	***	***	***	****	***	****	***	****	***	****	100	4583	100	3927	100	3150

**** Data not available

Table 4.14: Results of test functions

The parameters of the PSO-Bees Algorithm were empirically chosen. This probably is a reason why the *improved* Bees Algorithm found the optimum of the functions using a fewer number of function evaluations.

The optimisation stopped when the difference between the maximum fitness obtained and the global optimum was less than 0.1% of the optimum value, or less than 0.001, whichever is smaller. In the case where the optimum was 0, the solution was accepted if it differed from the optimum by less than 0.001.

As shown in Table 4.14, the *improved* Bees Algorithm performed better compared to the other global optimisation algorithms indicated by the smallest number of function evaluations.

4.4 Summary

This chapter has presented the *improved* Bees Algorithm, a modification and improvement to the original Bees Algorithm. The *improved* Bees Algorithm incorporates cooperation and communication between different patches (neighbourhoods) in the original Bees Algorithm to find the global optimum.

The results showed that the proposed cooperation and communication strategies implemented enhanced the performance and convergence of the *improved* Bees Algorithm. Secondly, it influenced the search process by ensuring the algorithm searches only the promising areas of the search space. Thirdly, it stops the need for 'killing' Bees

as employed in other variants of the original Bees Algorithm. Finally, it reduces the number of function evaluations of the algorithm in finding the global optimum of functions.

Furthermore, the results obtained from the application of the algorithm to mechanical design optimisation of the design of welded beams (single and multi objectives) and coiled springs are also presented.

Finally, the chapter concluded with the presentation of the enhanced results obtained by the *improved* Bees Algorithm on the mathematical benchmark problems.

Chapter 5: Novel Sequential Number-Theoretic Optimisation - Bees Algorithm

Q: What do little WASPs want to be when they grow up?
A: The very best person they can possibly be.

This chapter presents the Sequential Number-Theoretic Optimisation - Bees (SNTO-Bees) Algorithm, a modification and improvement to the original Bees Algorithm. The SNTO-Bees Algorithm came into existence while trying to resolve the limitations of the original Bees Algorithm on problems with high dimensions. The inspiration and motivation for the development of the SNTO-Bees Algorithm came from the wide use of the SNTO, a fairly new and powerful global optimisation technique that was widely employed in the field of statistics. The SNTO is a global optimisation method where many points are generated in a multi-dimensional capacity. The optimum point is selected and the domain is contracted around the neighbourhood of this point. The technique of point generation in multi-dimensional capacity is introduced to the original Bees Algorithm. The resulting algorithm, called the SNTO-Bees Algorithm is applied to solve mechanical design optimisation problems, in particular, the design of welded beams (single and multi objectives), the design of coil springs and the design of pressure vessel. In addition, the algorithm is tested on a number of deceptive multi-modal mathematical benchmark functions. Finally, the results obtained from another set of well-known mathematical benchmark functions are compared to those obtained by the SIMPSA, NESIMPSA, the GA, the ANT Algorithm, the original Bees Algorithm, the original PSO Algorithm together with the PSO-Bees and the *improved* Bees Algorithms presented in chapters three and four respectively.

5.1 Preamble

The SNTO is a global optimisation method where many points are generated in a multi-dimensional capacity. The optimum point is selected and the domain is contracted around the neighbourhood of this point. The inspiration and motivation of the SNTO-Bees Algorithm came from the wide use of the SNTO, a powerful new global optimisation technique in statistics. The essence of the SNTO method is to find a set of points that are universally scattered over an *s*-dimensional domain.

The SNTO technique is very attractive due to:

- its simplicity
- ease of implementation
- effective optimisation performance
- its ability to handle general optimisation problems
- its avoidance of the need to calculate the derivatives of objective functions.

The global optimisation techniques are superior to the classical optimisation techniques, such as the simplex methods because they can jump out from the local optima. Although the classical methods can be implemented by running several optimisation processes from different initial locations in the search space, it is still hard to guarantee that the algorithm will converge to the global optimum due to the fact that these methods only search locally (Gan *et al.* 2001). Prior to detailing the proposed SNTO-Bees Algorithm, it was proven that the purported global optimisation methods such as the GA, for instance, can reach the global optimum when the number of runs is infinite or to be exact, make a real

estimation of the global optimum. For a real problem with only one optimum, it is not difficult for the existing global optimisation methods to make a real estimation of the optimum in limited runs. However, practically in a real life problem there could be many local optima, the existing methods cannot guarantee a real estimation of the global optimum in limited runs. It is possible to estimate a local minimum, but at the cost of being trapped into local optimisation. This suggests that if estimations of all the potential optima are obtained, it is possible to reach the real global optimum by comparing these estimations. On occasion, it is not possible to find all the potential optima as many that are needed. As a result, since the global optimum is a maximum (minimum), it would be selected into the potential optima set if the search for all potential optima is performed in the same manner.

Let f be a function over a domain G, a subset of R^s. It is required to find the global maximum (minimum) M of f over G, and also a point $x^* \in G$, such that

$$M = f(x^*) = \begin{cases} \max_{x \in G} f(x) \\ \\ \min_{x \in G} f(x) \end{cases} \qquad (5.1)$$

M is called the global maximum (minimum) of the objective function f over G, and x^* a maximising (minimising) point on G.

There are many numerical methods for solving this problem, such as the downhill simplex method, Newton-Gauss method, quasi-Newton method, steepest descent method, conjugate gradient method and the restricted step methods. However, most of these

methods require that the function $f(x)$ is unimodal and / or differentiable to ensure that the global optimum can be attained. Otherwise, only a local maximum may be attained. Furthermore, these methods will have difficulties in finding the maxima of functions containing the expression 'max', 'min' or $|x|$ if $f(x)$ is defined strictly such that:

$$f(x) = \begin{cases} f_1(x), \text{ if } x \in D_1 \\ \quad \ldots\ldots \\ f_m(x), \text{ if } x \in D_m \end{cases} \qquad D_1 \cup \ldots \cup D_m = D, \tag{5.2}$$

where the derivative often does not exist or is not easily computed on the boundary of each D_i. The book written by Horst and Tuy (Horst and Tuy 1990) has a large number of diverse algorithms for solving a wide variety of multi-extremal global optimisation problems.

As mentioned earlier, the inspiration for the SNTO-Bees Algorithm came from statistics. There are many problems / applications in statistics needing powerful algorithms for optimisation, for example:

- maximum likelihood estimation
- non-linear regression
- model selection
- evaluation of discrepancy of a set of points
- projection pursuit
- experimental design

just to mention a few.

These examples share some or all of the following difficulties in solving optimisation problems (Fang *et al.* 1996):

- the objective function f is multi-extremal;
- the objective function f is not differentiable or even continuous everywhere in G;
- the dimension of the domain G is high;
- the domain G is large in extent, for example $G = R^s$;
- the domain G is the surface of a sphere or some other geometric object;
- the domain G is a finite set with a large number of elements.

The choice of a suitable optimisation algorithm for a specific problem is not an easy task, and it is difficult to objectively compare different results. The SNTO, a powerful new global optimisation technique in statistics is known to comfortably handle the above listed issues (Fang *et al.* 1996).

5.2 Sequential Number-Theoretic Optimisation (SNTO) Algorithm

One probabilistic method for solving optimisation problems as defined in equation 5.2 is to draw a simple random sample, $\mathcal{P}$, on n points from the domain G. If n is large enough, the optimum of f on $\mathcal{P}$ will be close to the global optimum M. If the points in $\mathcal{P}$ are statistically independent, they will not be evenly distributed over the domain (the second point is as likely to be close to the first point as it is to be far away from it) as shown in Figure 5.1a. This makes the convergence of a random search slow and this is applicable to the original Bees Algorithm. A better choice is a set of deterministic quasi-random (having low discrepancy) points, sometimes called an NT-net. These points (obtained by

a so-called *good lattice point modulo n* see the *glp* set in Appendix F below) are uniformly scattered in an s-dimensional unit cube C^s.

(Niederreiter and Peart 1986) and (Fang and Wang 1990) independently proposed quasi-random searches over contracting domains called the Sequential Number-Theoretic method for Optimisation (SNTO).

Figure 5.1 (Fang *et al.* 1994) below shows two kinds of sets: (a) a random number distribution and (b) an NT-net distribution respectively.

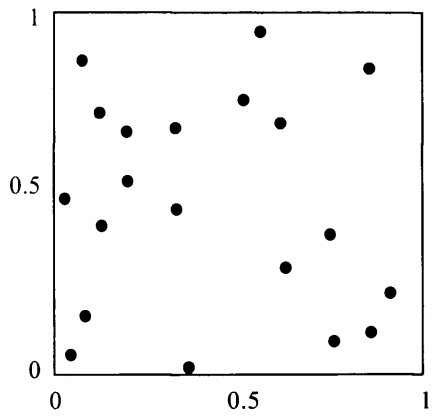


Figure 5.1a: A random numbers distribution

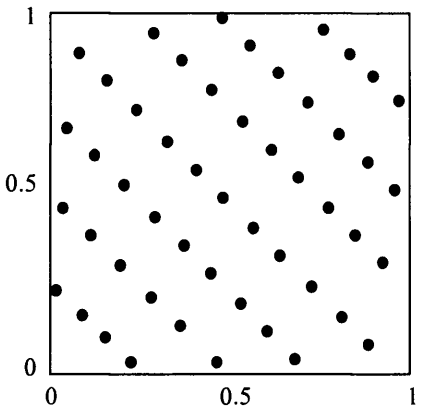


Figure 5.1b: an NT-net distribution

The operation of the SNTO technique is shown in Figure 5.2 (Fang and Wang 1994).

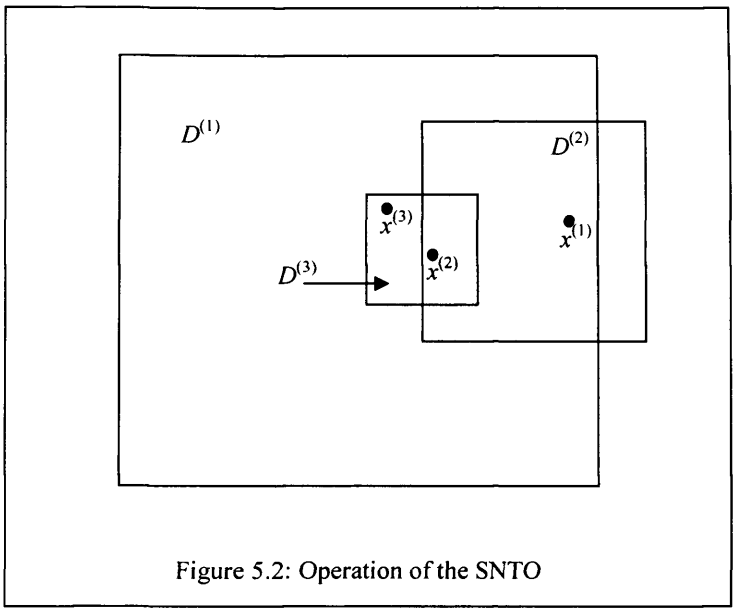


Figure 5.2: Operation of the SNTO

where each edge-length of $D^{(t)}$ is ½ times those of $D^{(t-1)}$. If $x^{(t)}$ is near the boundary of D as $x^{(1)}$ as shown in Figure 5.2, then $D^{(t)}$ should be required to fall in D; the edge-length may contract to less than ½ that of $D^{(t-1)}$. It is not continuously necessary to have $D^{(t+1)} \subset D^{(t)}$ ($t > 1$). On the other hand, the domain can be contracted by selecting $D^{(t+1)} = [\mathbf{a}^{(t+1)}, \mathbf{b}^{(t+1)}]$ to be the smallest box containing the ρn_t points in $D^{(t)}$ with the maximum function values for some predefined proportion, say $\rho = 0.3$.

5.3 Sequential Number-Theoretic Optimisation (SNTO) - Bees Algorithm

As previously mentioned, the inspiration and motivation for developing the SNTO-Bees Algorithm came from the wide use of the SNTO, a powerful new global optimisation technique in statistics with initial studies conducted to introduce this technique to chemistry. As a result of the impressive attractive features mentioned in Section 5.1, in addition to its well documented exceptional sturdiness, capability and performance in the

literature on problems with high dimension scope stimulated the idea of making the Bees Algorithm a better optimisation tool by incorporating this technique.

At the moment, the Bees Algorithm (Section 2.2 of Chapter 2) uses random initialisation but as shown in Figures 5.1a and 5.1b, whereas theoretically, the bees are evenly distributed but practically, the bees are not *properly* evenly distributed across the search space. A poorer random distribution is even observed with problems having high dimensions.

Implementing the SNTO technique of generating points in a multi dimensional capacity in the original Bees Algorithm would result in:

- a robust (evenly distributed in all dimensions from initialisation) algorithm
- fast convergence to the global optimum of objective functions
- eliminating the need for 'killing' bees as previously required in some variants of the original Bees Algorithm
- avoiding being trapped in local optima
- large exploration across all dimensions and later, an exploitative local search to improve the solution.

The pseudo code of the SNTO-Bees Algorithm is presented below in Table 5.1.

Step 0	**Initialisation**
	Set iteration index = $t = 0$
	Set initial search domain = $G^{(0)} = G$; $G^{(0)} = [\boldsymbol{a}^{(0)}, \boldsymbol{b}^{(0)}]$,
	where $\boldsymbol{a}^{(0)} = \boldsymbol{a}$, and $\boldsymbol{b}^{(0)} = \boldsymbol{b}$; and $\boldsymbol{x}^{(-1)}$ be the empty set.
Step 1	**Generate an NT-net**
	Use a number-theoretic method to generate n_t points $\mathfrak{P}^{(t)}$ uniformly scattered on domain $G^{(t)} = [\mathbf{a}^{(t)}, \mathbf{b}^{(t)}]$.
Step 2	**While** (Stopping criterion not met)
	// Forming new population
Step 3	**Compute a new approximation (Global Search)**
	Find the point $x^{(t)} \in \mathfrak{P}^{(t)} \cup$ f and $M^{(t)}$ that minimises f, that is,
	$M^{(t)} = f(\boldsymbol{x}^{(t)}) \leq f(\mathbf{y}) \qquad \forall y \in \mathfrak{P}^{(t)} \cup \{\boldsymbol{x}^{(t-1)}\}$
	$x^{(t)}$ and $M^{(t)}$ are the best approximations to x^* and M so far.
Step 4	**Improve already found solution (Local Search)**
	Select patch around $x^{(t)}$ for adaptive neighbourhood search
	Recruit bees for the selected patch and evaluate their fitness
	Select fittest bee from patch (new $x^{(t)}$)
Step 5	**Termination Criterion**
	Let $\mathbf{c}^{(t)} = (\mathrm{c}_1^{(t)}, \ldots . \mathrm{c}_s^{(t)}) = (\mathbf{b}^{(t)} - \mathbf{a}^{(t)}) / 2$.
	If max $\mathbf{c}^{(t)} < \delta$, a pre-assigned small number or tolerance, then $G^{(t)}$ is small enough; $x^{(t)}$ and $M^{(t)}$ are deemed acceptable; terminate the algorithm. Otherwise, proceed to step 6.
Step 6	**Contract search domain**
	Construct new domain $G^{(t+1)} = [\mathbf{a}^{(t+1)}, \mathbf{b}^{(t+1)}]$ as follows:
	$\mathbf{a}_i^{(t+1)} = \max(x_i^{(t)} - \gamma \mathrm{c}_i^{(t)}, \mathbf{a}_i^{(t)})$
	and
	$\mathbf{b}_i^{(t+1)} = \min(x_i^{(t)} + \gamma \mathrm{c}_i^{(t)}, \mathbf{b}_i^{(t)}) \qquad$ for $i = 1, \ldots, s$
	where γ is a predefined **contraction ratio**. Set $t = t + 1$.
	Go to Step 1.

Table 5.1: Pseudo code of the SNTO-Bees Algorithm

A large value of γ contracts the domain too fast thus making the algorithm to jump over good solutions and continue to search in fruitless regions of the problem space. On the other hand, a small value will result in large computations that increases the number of function evaluations in finding the global optimum of objective functions because it would take longer time for the termination criteria to be true (max $\mathbf{c}^{(t)} < \delta$ - step 5).

The above leaves the problem of finding an appropriate value for γ in order to have a balance between:

- contracting the domain too fast or too slow
- exploitation and exploration

the optimal value of γ is problem dependent. A smooth search space will need a large value compared with a rough surface to locate optimal solutions.

Furthermore, the more the n_t points $\mathbb{P}^{(t)}$ uniformly scattered on domain $G^{(t)}$, the larger would be the initial diversity because a large swarm allows larger parts of the search space to be covered in each iteration. It has the demerit of increasing the computational complexity. However, on the other hand, it has the merit of needing fewer numbers of iterations to reach a good solution compared to smaller n_t points.

Again worth mentioning, the optimal number of n_t points $\mathbb{P}^{(t)}$ uniformly scattered on domain $G^{(t)}$ is problem dependent. A smooth search space will need fewer n_t points compared with a rough surface to locate optimal solutions.

5.4 Results

This section presents the results of six different applications of the SNTO-Bees Algorithm. First, the algorithm is applied to four mechanical design optimisation problems:

- the design of a welded beam structure (single objective)
- the design of a welded beam structure (multi-objective)
- the design of a coil spring and
- the design of a pressure vessel.

These four mechanical design optimisation problems are used to benchmark the algorithm against other previously applied global optimisers. The welded beam design problem entails a non-linear objective function with eight constraints; the coil spring design problem has a non-linear objective function having just four constraints whilst the design of the pressure vessel is also a non-linear objective function with four constraints.

Secondly, the algorithm was applied to a number of deceptive multi-modal benchmark functions (the visualisation and contour diagrams of these functions plotted in Matlab are presented in Appendix E), in addition to the test functions presented in Section 3.6.4 of Chapter 3 and Section 4.3.4 of Chapter 4. The presentation of the results obtained concludes the chapter.

5.4.1 Application to Mechanical Design Optimisation – Welded Beam Design Problem

In this section, the SNTO-Bees Algorithm is applied to mechanical design optimisation – the welded beam design problem which is the same as the single objective design optimisation problem described in Section 4.3.1 of Chapter 4. This involved a non-linear objective function with eight constraints. Please refer to Section 4.3.1 of Chapter 4 for the detailed information on the objective function, constraints and the diagram of the welded beam structure. Table 5.2 below presents the results obtained.

From Table 5.2 below, the SNTO-Bees Algorithm produced even better results compared to the *improved* Bees Algorithm that was presented in Chapter Four while Figure 5.3 shows the evolution of lowest cost in each iteration.

Figure 5.3 shows how the lowest value of the objective function changes with the number of iterations (generations) for three independent runs of the algorithm. It can be seen that the objective function decreases rapidly in the early iterations and then gradually converges to the optimum value.

Methods	Design variables				Cost
	h	*l*	*t*	*b*	
GA (Deb 1991) **Three independent runs**	0.2489	6.1730	8.1789	0.2533	2.43
	0.2679	5.8123	7.8358	0.2724	2.49
	0.2918	5.2141	7.8446	0.2918	2.59
IMPROVED GA (Leite and Topping 1998) **Three independent runs**	0.2489	6.1097	8.2484	0.2485	2.40
	0.2441	6.2936	8.2290	0.2485	2.41
	0.2537	6.0322	8.1517	0.2533	2.41
SIMPLEX (Ragsdell and Phillips 1976)	0.2792	5.6256	7.7512	0.2796	2.53
RANDOM (Ragsdell and Phillips 1976)	0.4575	4.7313	5.0853	0.6600	4.12
BEES ALGORITHM Three independent runs (Pham and Ghanbarzadeh 2006)	0.24429	6.2126	8.3009	0.24432	2.3817
	0.24428	6.2110	8.3026	0.24429	2.3816
	0.24432	6.2152	8.2966	0.24435	2.3815
Improved **BEES ALGORITHM Three independent runs**	0.24427	6.2131	8.3012	0.24431	2.381738
	0.24426	6.2019	8.3180	0.24401	2.381437
	0.24429	6.2122	8.3019	0.24426	2.381421
SNTO-BEES ALGORITHM Three independent runs	0.24379	6.2164	8.2832	0.24489	2.381064
	0.24385	6.2279	8.2948	0.24427	2.380903
	0.24398	6.2094	8.2962	0.24451	2.380587

Table 5.2: Comparison of results of the SNTO-Bees Algorithm on welded beam design problem with other optimisers

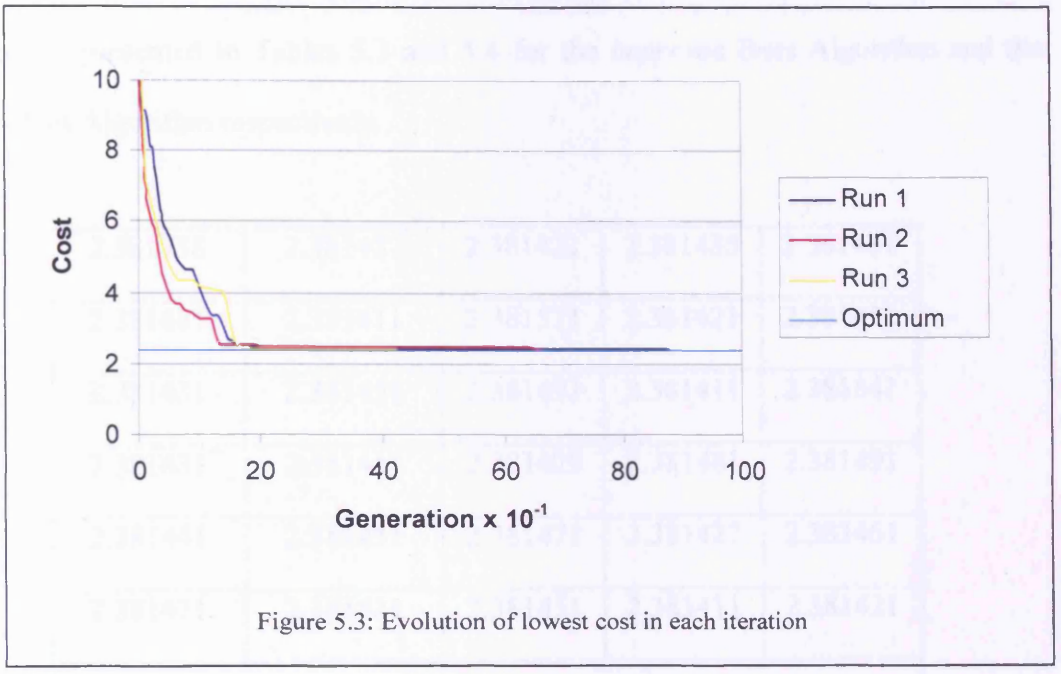


Figure 5.3: Evolution of lowest cost in each iteration

A question persists 'what is the statistical significance of the result presented in Table 5.2?'

To check the statistical significance of the result, a T-TEST had to be performed which checks the relationship between two variables, in this case two different algorithms

The T-Test was conducted between the *improved* Bees Algorithm and the SNTO-Bees Algorithm. The two algorithms were applied 30 times to the design of the welded beam problem.

Figure 5.4 shows a plot of the minimum cost produced by both algorithms. The values of the plot are presented in Tables 5.3 and 5.4 for the *improved* Bees Algorithm and the SNTO-Bees Algorithm respectively.

2.381738	2.381437	2.381421	2.381435	2.381481
2.381441	2.381411	2.381571	2.381421	2.381411
2.381431	2.381451	2.381491	2.381411	2.381541
2.381431	2.381411	2.381429	2.381481	2.381491
2.381441	2.381451	2.381471	2.381427	2.381461
2.381471	2.381431	2.381451	2.381411	2.381421

Table 5.3: Minimum cost produced by the *improved* Bees Algorithm for the welded beam design problem

2.381064	2.380903	2.380587	2.380564	2.380613
2.380597	2.380619	2.380749	2.380607	2.380621
2.380587	2.380697	2.380593	2.380782	2.380874
2.380623	2.380587	2.380749	2.380598	2.380641
2.380801	2.380791	2.380587	2.380631	2.380587
2.381002	2.380657	2.380635	2.380782	2.380623

Table 5.4: Minimum cost produced by the SNTO-Bees Algorithm for the welded beam design problem

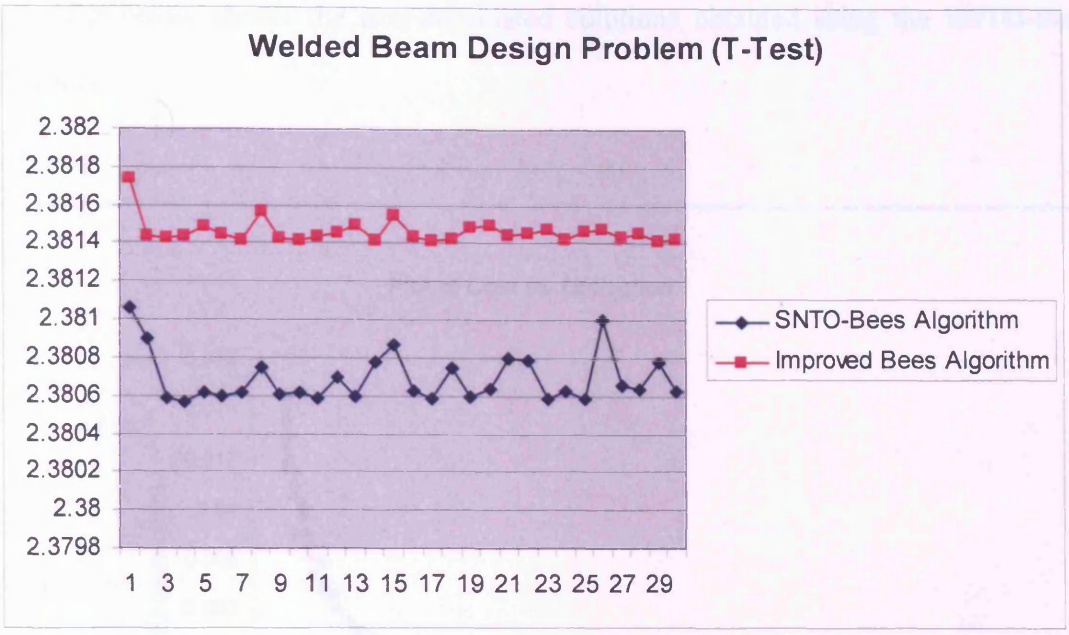


Figure 5.4: Plot of the minimum cost produced by the *improved* Bees Algorithm and the SNTO-Bees Algorithm for the welded beam design problem

I obtained an alpha value of 5.82624E-36 from the T-Test. This value indicates that the result obtained by both the *improved* Bees Algorithm and the SNTO-Bees Algorithm is significantly different with a confidence level above 99%.

5.4.2 Application to Multi-Objective Optimisation – Welded Beam Design Problem

In this section, the application of the SNTO-Bees Algorithm to multi-objective mechanical design optimisation – the design of welded beam problem which is the same as the multi-objective design optimisation problem described in Section 4.3.2 of Chapter 4, is presented. Please refer to Section 4.3.2 of Chapter 4 for the detailed information on the objective function, constraints and the diagram of the welded beam structure.

Figure 5.5 below shows the non-dominated solutions obtained using the SNTO-Bees Algorithm.

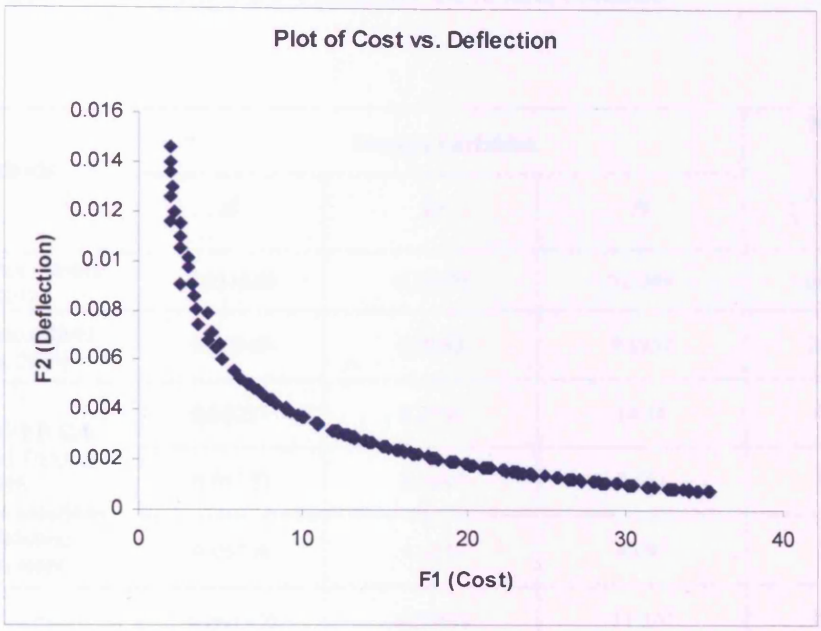


Figure 5.5: Non-dominated solutions obtained using the SNTO-Bees Algorithm

The SNTO-Bees Algorithm found more non-dominated solutions in comparison with the number of solutions obtained by the non-dominated sorting genetic algorithms, the original Bees Algorithm and the *improved* Bees Algorithm that was presented in Chapter 4.

5.4.3 Application to Mechanical Design Optimisation – A Coiled Spring Problem

In this section, the SNTO-Bees Algorithm is applied to mechanical design optimisation – the design of coiled spring problem which is the same as the single objective design

optimisation problem described in Section 4.3.3 of Chapter 4, that encompasses a non-linear objective function with four constraints is presented. The detailed information on the objective function, constraints and the diagram of the coiled spring is given in Section 4.3.3 of Chapter 4. Table 5.5 below presents the results obtained.

Methods	**Design variables**			**Mass M** $\left(\times {}^{4}/_{\rho\pi^2}\right)$
	d	D	N	
SQP (batch) (Arora 2004)	0.051699	0.35695	11.289	0.0126787
SQP (interactive) (Arora 2004)	0.05340	0.3992	9.1854	0.0127300
IMPROVED GA (Leite and Topping 1998) **Best three solutions not violating constraints**	0.05235	0.3721	10.48	0.01272
	0.05323	0.3947	9.383	0.01273
	0.05396	0.4132	8.697	0.01287
BEES ALGORITHM Three independent runs (Pham and Ghanbarzadeh 2006)	0.051759	0.35839	11.207	0.012680
	0.051807	0.35956	11.139	0.012680
	0.051779	0.35886	11.179	0.012681
Improved BEES Algorithm Three independent runs	0.051544	0.353238	11.511	0.012679756
	0.051855	0.360722	11.072	0.012679315
	0.051852	0.360651	11.076	0.012679234
SNTO-BEES Algorithm Three independent runs	0.051564	0.3353712	11.482	0.012679352
	0.051544	0.353238	11.5104	0.012679197
	0.051563	0.353693	11.4829	0.012679023

Table 5.5: Comparison of the SNTO-Bees Algorithm results on coiled spring with other optimisers

Table 5.5 above illustrates that the SNTO-Bees Algorithm produced superior results compared to the original Bees Algorithm, the *improved* Bees Algorithm, the improved GA, the SQP (batch) and the SQP (interactive).

Figure 5.6 shows the evolution of the minimum mass in each iteration.

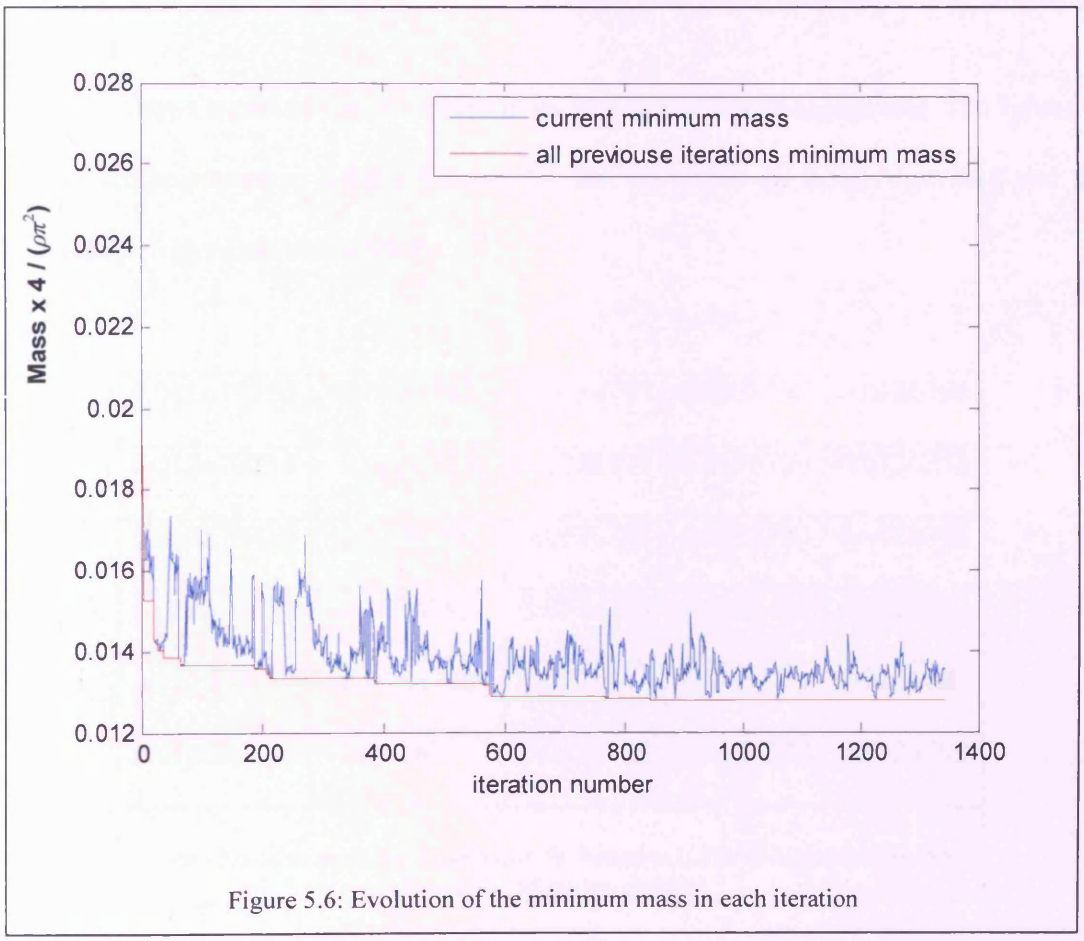


Figure 5.6: Evolution of the minimum mass in each iteration

A question persists 'what is the statistical significance of the result presented in Table 5.5?'

To check the statistical significance of the result, a T-TEST had to be performed which checks the relationship between two variables, in this case two different algorithms.

The T-Test was conducted between the *improved* Bees Algorithm and the SNTO-Bees Algorithm. The two algorithms were applied 30 times to the design of the coil spring problem.

Figure 5.7 shows a plot of the minimum mass produced by both algorithms. The values of the plot are presented in Tables 5.6 and 5.7 for the *improved* Bees Algorithm and the SNTO-Bees Algorithm respectively.

0.012679756	0.0126793	0.0126792	0.0126797	0.0126794
0.012679334	0.0126795	0.0126797	0.0126796	0.0126793
0.012679415	0.0126796	0.0126793	0.0126792	0.0126793
0.012679715	0.0126793	0.0126795	0.0126794	0.0126792
0.012679434	0.0126793	0.0126795	0.0126795	0.0126793
0.012679237	0.0126794	0.0126795	0.0126792	0.0126793

Table 5.6: Minimum mass produced by the *improved* Bees Algorithm for the design of coil spring problem

0.0126793	0.0126793	0.0126797	0.0126792	0.0126790
0.0126792	0.0126790	0.0126791	0.0126794	0.0126792
0.0126791	0.0126792	0.0126790	0.0126793	0.0126793
0.0126792	0.0126790	0.0126792	0.0126793	0.0126792
0.0126791	0.0126792	0.0126791	0.0126791	0.0126790
0.0126791	0.0126790	0.0126791	0.0126790	0.0126791

Table 5.7: Minimum mass produced by the SNTO-Bees Algorithm for the design of coil spring problem

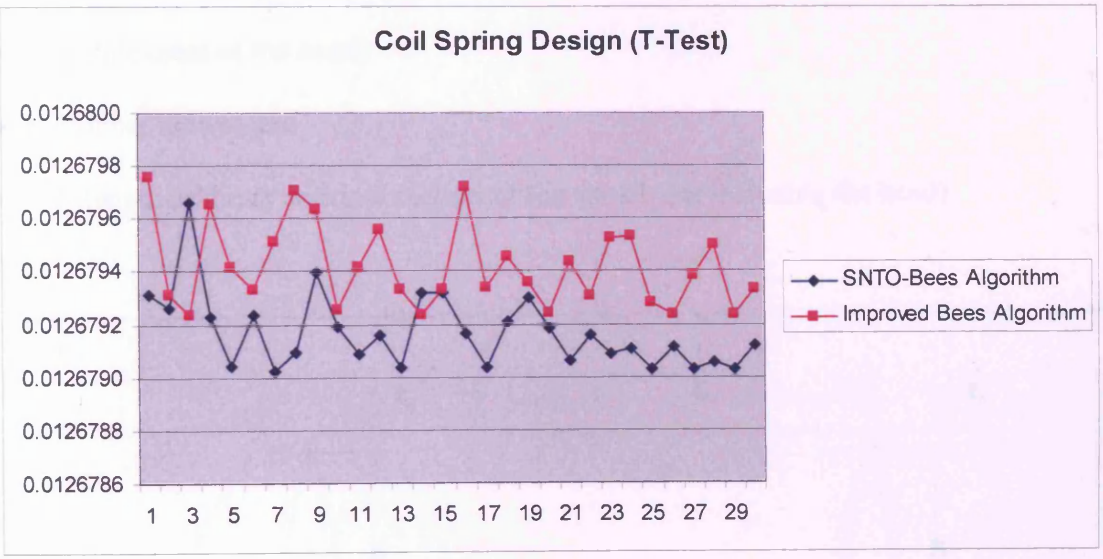


Figure 5.7: Plot of the minimum mass produced by the *improved* Bees Algorithm and the SNTO-Bees Algorithm for the design of coil spring problem

I obtained an alpha value of 2.35816E-08 from the T-Test. This value indicates that the result obtained by both the *improved* Bees Algorithm and the SNTO-Bees Algorithm is most significantly different with a confidence level above 99%.

5.4.4 Application to Mechanical Design Optimisation – Design of a Pressure Vessel Problem

In this section, the SNTO-Bees Algorithm is applied to mechanical design optimisation – the design of a pressure vessel. A **pressure vessel** is a closed container designed to hold gases or liquids at a pressure different from the ambient pressure. Figure 5.8 below shows the diagram of a cylindrical vessel capped at both ends by hemispherical heads (center and the end section of pressure vessel) as described in Problem 1 of (Mezura-Montes *et al.* 2003). The objective is to minimise the total cost, including the cost of the material, forming and welding. There are four design variables:

- T_s (thickness of the shell)
- T_h (thickness of the head)
- R (inner radius) and
- L (length of the cylindrical section of the vessel, not including the head)

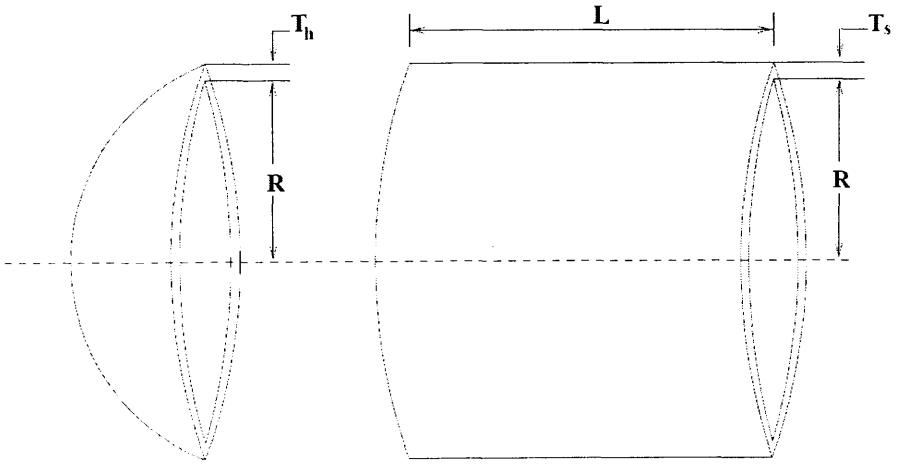


Figure 5.8: A Cylindrical vessel capped at both ends by hemispherical heads - Center and the end section of pressure vessel (Mezura-Montes *et al.* 2003)

T_s and T_h are integer multiples of 0.0625 inch, which are the available thickness of rolled steel plates while R and L are continuous. Applying the same notation as specified in (Kannan and Kramer 1994), the problem of the pressure vessel is defined as follows:

$$\text{Min: } f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661\,x_1^2x_4 + 19.84\,x_1^2x_3 \tag{5.3}$$

Subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0 \tag{5.4}$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0 \tag{5.5}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1{,}296{,}000 \leq 0 \tag{5.6}$$

$$g_4(\vec{x}) = -x_4 - 240 \leq 0 \tag{5.7}$$

where

$1 \leq x_1 \leq 99$; $1 \leq x_2 \leq 99$; $10 \leq x_3$ 200 and $10 \leq x_4 \leq 200$.

Table 5.8 below presents the results obtained by the SNTO-Bees Algorithm from 30 runs. The results obtained by the Simple Evolution Strategy (Mezura-Montes *et al.* 2003) and the Socio-Behavioural (SB) approach (Akhtar *et al.* 2002) are also included for comparison.

Parameters	Details of best solution found		
	Socio-Behavioural (SB) approach (Akhtar *et al.* 2002)	**Simple Evolution Strategy** (Mezura-Montes *et al.* 2003)	**SNTO-Bees Algorithm**
$x1$	0.8125	0.812500	0.812572
$x2$	0.4375	0.437500	0.43745
$x3$	41.9768	42.098370	42.10215
$x4$	182.2845	176.637146	176.5612
$g_1(x)$	-0.0023	-0.000001	-0.000000005
$g_2(x)$	-0.0370	-0.035882	-0.0357954890
$g_3(x)$	-23420.5966	-0.835772	-3.8109075427
$g_4(x)$	-57.7155	-63.362858	-63.438790000
$f(x)$	**6171.0**	**6059.714355**	**6058.9090080**

Table 5.8: Comparison of the SNTO-Bees Algorithm results on the design of pressure vessel with other optimisers

As indicated in Table 5.8 above, the SNTO-Bees Algorithm produced results of better quality and robustness compared to the Socio-Behavioural approach (Akhtar *et al.* 2002) and the Simple Evolution Strategy (Mezura-Montes *et al.* 2003).

5.4.5 Application to Multi-modal Deceptive functions (MCastellani 1 – 10)

To demonstrate the ability of the SNTO-Bees Algorithm to tackle deceptive optimisation problems, it is applied to a number of deceptive multi-modal mathematical benchmark functions. Mathematical benchmark functions are useful for testing and comparing techniques based on real vectors ($X = \mathbb{R}^n$). However, they only require such vectors as solution candidates, i.e., elements of the problem space X.

In this research, ten deceptive multi-modal optimisation test functions (Castellani 2008) was used to assess the SNTO-Bees Algorithm as a global optimiser, their properties are shown in Table 5.9. The results obtained from each of these benchmark test functions are presented in Table 5.10. The mathematical equations along with the graphical representations (visualisation and contour plots) of the test functions identified as MCastellani TF 1 through 10 are presented in Appendix E.

No	Reference	Interval	Description of Test Function	Global Optimum
1	MCastellani TF 1	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Two valleys at different angles located at opposite sides of the solution space	X[-100,-30] F=0
2	MCastellani TF 2	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Two 'holes' of different eccentricity located at opposite sides of the solution space	X[-60,-60] F=0
3	MCastellani TF 3	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal surface with the minimum located far from the centre	X[75,75] F=0
4	MCastellani TF 4	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	More complex multimodal surface with the minimum located far from the centre	X[80,80] F=0
5	MCastellani TF 5	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal surface having many minima regularly spaced with the global optimum located far from the centre	X[75,75] F=0
6	MCastellani TF 6	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal function having many local minima that are not regularly spaced	X[50,50] F=0
7	MCastellani TF 7	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal surface with the minimum located at the end of a narrow valley near to the borders of the surface	X[0,0,] F=0
8	MCastellani TF 8	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Very deceptive function with the minimum located at the end of an ellipsoidal valley	X[90,0] F=0
9	MCastellani TF 9	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal function with the minimum located in the central position surrounded by flat area – the area where the minimum is located is very small.	X [0,0] F=0
10	MCastellani TF 10	$[-100 \leq x_1 \leq 100]$ $[-100 \leq x_2 \leq 100]$	Multimodal function with the minimum located in a periferic location surrounded by a flat area - the area of minimum is very small.	X [-75, 75] F=0

Table 5.9: Properties of test functions used for the SNTO-Bees Algorithm

SNTO-Bees Algorithm						
Function No.	**Test Function**	**Success (%)**	**Mean Number of Function Evaluation**	**Minimum position**	**Global minimum**	**Iteration Time (sec.)**
1	MCastellani TF 1	100	851	X[-100.0000, -29.8144]	-0.0013	10.9892
2	MCastellani TF 2	100	1081	X[-59.9991, -60.0013]	4.3418e-08	12.6516
3	MCastellani TF 3	100	1102	X[74.9310, 74.9296]	-1.3139e-10	14.2388
4	MCastellani TF 4	100	1294	X[79.4819, 79.4815]	-2.2795e-08	27.1370
5	MCastellani TF 5	100	1037	X[75.0276, 75.0286]	-5.6279e-05	16.6020
6**	MCastellani TF 6	92	1322	X[49.7812, 49.7916]	-0.0758	17.6929
7	MCastellani TF 7	100	1452	X[-0.0001, -0.0015]	0.002000	41.4334
8	MCastellani TF 8	100	1276	X[90.002, 0.0019]	-3.8026e-06	32.0084
9**	MCastellani TF 9	92	1503	X[0.01428, -0.01437]	9.0306e-12	72.9967
10	MCastellani TF 10	100	1839	X[-75.0008, -75.0002]	2.9676e-09	78.6792

Table 5.10: Performance of the SNTO-Bees Algorithm on MCastellani TF 1 through 10

As shown in Table 5.10 above, with the exception of MCastellani 6 and 9, the SNTO-Bees Algorithm found the global optimum of the test functions with 100% success using a small mean number of function evaluations obtained from 100 independent runs.

With MCastellani 6 and MCastellani 9, the SNTO-Bees Algorithm had 92% success from 100 runs. This is because the same *nt* points were used. I observed that when the number of *nt* points is increased together with a much smaller contraction ratio, the SNTO-Bees Algorithm had 100% success with MCastellani 6 and MCastellani 9.

For the calculation of the global minimum of MCastellani 1 – 10, a double variable type was used as default. This was done in order to overcome the problem of limited precision of numerical evaluation of the functions, for example, in MCastellani 3, X[75, 75] was used and the global optimum found was -1.3139e-10, and not 0.

5.4.6 Application to Mathematical Benchmark Problems

The SNTO-Bees Algorithm is again applied to a list of well-known mathematical benchmark functions that was previously used to test the PSO-Bees Algorithm and the *improved* Bees Algorithm in Chapters 3 and 4 respectively. The functions are presented in Table 5.11 while the results obtained from this implementation are presented in Table 5.12 which shows that the SNTO-Bees Algorithm produces better results.

No	Reference	Interval	Test Function	Global Optimum
1	De Jong	[-2.048, 2.048]	$\max F = (3905.93) - 100\,(x_1^2 - x_2^2) - (1 - x_1)^2$	X[1,1] F=3905.93
2	Goldstein & Price	[-2, 2]	$\min F = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $X[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	X[0,-1] F=3
3	Branin	[-5, 10]	$\min F = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e$ $a = 1, b = \frac{5.1}{4}\left(\frac{7}{22}\right)^2, c = \frac{5}{22}X7, d = 6, e = 10, f = \frac{1}{8}X\frac{7}{22}$	X[-22/7,12.275] X[22/7,2.275] X[66/7,2.475] F=0.3977272
4	Martin & Gaddy	[0, 10]	$\min F = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$	X[5,5] F=0
5	Rosenbrock -1	(a) [-1.2, 1.2] (b) [-10, 10]	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	X[1,1] F=0
6	Rosenbrock - 2	[-1.2, 1.2]	$\min F = \sum_{i=1}^{3} \{100(x_i^2 - x_{i+1})^2 + (1 - x_1)^2\}$	X[1,1,1,1] F=0
7	Hyper sphere model	[-5.12, 5.12]	$\min F = \sum_{i=1}^{6} x_i^2$	X[0,0,0,0,0,0] F=0
8	Griewangk	[-512, 512]	$\max F = \frac{1}{0.1 + \left(\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)}$	X[0,0,0,0,0,0,0,0,0,0] F=10
9	Ackley	[-5.12, 5.12]	$f(x) = 20 + e - 20\,e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}$	X [0, …, 0] F=0
10	Schwefel	[-500, 500]	$f(x) = 418.9829 \cdot n - \sum_{i=1}^{n} (-x_i \text{ Sin } (\sqrt{\lvert x_i \rvert}))$	X [1, …, 1] F=0

Table 5.11: Test Functions (Mathur *et al.* 2000)

func no	SIMPSA		NE SIMPSA		GA		ANT		Bees Algorithm		PSO Algorithm		PSO-Bees Algorithm		*Improved* Bees Algorithm		SNTO-Bees Algorithm	
	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals	Success %	mean no of func. evals
1	***	****	****	****	100	10160	100	6000	100	868	100	872	100	815	100	806	100	785
2	***	****	****	****	100	5662	100	5330	100	999	100	1008	100	879	100	851	100	812
3	***	****	****	****	100	7325	100	1936	100	1657	100	1594	100	1463	100	1387	100	1311
4	***	****	****	****	100	2844	100	1688	100	526	100	507	100	486	100	462	100	436
5a	100	10780	100	4508	100	10212	100	6842	100	631	100	609	100	594	100	573	100	553
5b	100	12500	100	5007	***	****	100	7505	100	2306	100	2281	100	1829	100	1794	100	1890
6	99	21177	94	3053	***	****	100	8471	100	28529	100	27736	100	21105	100	20729	100	20018
7	***	****	****	****	100	15468	100	22050	100	7113	100	6930	100	6794	100	6485	100	6153
8	***	****	****	****	100	200000	100	50000	100	1847	100	1851	100	1798	100	1671	100	1604
9	***	***	***	****	***	****	***	****	***	****	100	2247	100	1979	100	1829	100	1786
10	***	***	***	****	***	****	***	****	***	****	100	4583	100	3927	100	3150	100	2975

**** Data not available

Table 5.12: Results of test on other benchmark functions

The optimisation stopped when the difference between the maximum fitness obtained and the global optimum was less than 0.1% of the optimum value, or less than 0.001, whichever is smaller. In the case where the optimum was 0, the solution was accepted if it differed from the optimum by less than 0.001.

As shown in Table 5.11, the SNTO-Bees Algorithm performed better compared to the other global optimisation algorithms indicated by the smallest number of function evaluations.

The advantages and disadvantages of the proposed algorithm is presented in each chapter.

5.5 Summary

This chapter has presented the Sequential Number-Theoretic Optimisation - Bees (SNTO-Bees) Algorithm, a modification and improvement to the Bees Algorithm. The algorithm came into existence while trying to resolve the limitations of the original Bees Algorithm on problems with high dimensions. The technique of point generation in a multi dimensional capacity has been implemented in the original Bees Algorithm and applied to mechanical design optimisation problems, in particular, the design of welded beam (single objective and multi objective), the design of a coil spring and the design of a pressure vessel. Finally the algorithm was tested on multi-modal deceptive benchmark functions in addition to a number of well-known benchmark functions previously used in

Chapters 3 and 4 to benchmark the PSO-Bees Algorithm and the *improved* Bees Algorithm respectively.

The results obtained by the SNTO-Bees Algorithm was better compared to those produced by previously applied optimisers for the same mechanical design optimisation problems as well as the other renowned mathematical benchmark functions.

Chapter 6: Conclusion

Change your thoughts and you change your world.

This chapter summarises the main contributions of this research and the conclusions reached. It also provides suggestions for further works. This research has focused on enhancements to the Bees Algorithm (*improved* Bees Algorithm and SNTO-Bees Algorithm) and resolving the problem of premature convergence in the Particle Swarm Optimisation Algorithm (PSO-Bees Algorithm).

6.1 Contributions

The specific contributions are:

- The development of a PSO-Bees Algorithm by improving the ability of the PSO Algorithm to converge onto the global optimum of objective functions. This helped to solve the major problem of premature convergence known to exist in the PSO Algorithm by combining the fast convergence property of the PSO Algorithm and the inherent ability of the original Bees Algorithm to avoid being trapped in local optima.

- The development of a new *improved* Bees Algorithm. This is achieved by the introduction of a momentum into the original Bees Algorithm to guide and assist the search process (a balance between exploration and exploitation). This helped to eliminate the need for 'killing' Bees and the declining effect of global random search as the iteration progresses. The momentum has an analogous effect to the

velocity update equation in the PSO Algorithm (that is, improving the original Bees Algorithm with the PSO Algorithm).

- The development of a new Bees Algorithm called SNTO-Bees Algorithm with the ability to converge onto the local or global optimum depending on the nature of the objective functions. The algorithm performed significantly better handling test functions with high dimensionality.

- The introduction of a PSO-Bees Algorithm to train a Multi-Layer Perceptron (MLP) neural network for Control Chart Pattern Recognition and Wood Defect Classification problems. The algorithm performed better in the classification and recognition applications.

- The comparisons of the proposed algorithms. These show promising results and the proposed algorithms are rigorously competitive with other methods in terms of computational costs and the success of obtaining the global solutions. The proposed algorithms showed a superior performance in terms of the solution qualities against the compared methods.

6.2 Conclusions

The objectives stated in chapter 1 have all been achieved. This research has demonstrated the hypothesis that improved nature-inspired optimisation algorithms will result from hybridisation.

This thesis has presented three new optimisation algorithms: the PSO-Bees Algorithm, the *improved* Bees Algorithm and the SNTO-Bees Algorithm. Experimental results on training neural networks, benchmark test functions, multi-modal deceptive functions and mechanical design optimisation show that the proposed algorithms has remarkable robustness, producing a 100% success rate in all cases. The algorithms converged to the maximum or minimum without becoming trapped at local optima and generally outperformed other techniques that were compared with it in terms of speed of optimisation and accuracy of the results obtained. Thus, objectives 1-8 have been met.

Two different constrained mechanical design optimisation problems were solved using the *improved* Bees Algorithm. In each case, the algorithm converged to the optimum without becoming trapped at local optima. Again, the algorithm generally outperformed other optimisation techniques in terms of the accuracy of the results obtained. Thus, objective 8 has been met.

Three different constrained mechanical design optimisation problems were solved using the SNTO-Bees Algorithm. In each case, the algorithm converged to the optimum without becoming trapped at local optima and outperformed other optimisation techniques in terms of the accuracy of the results obtained. Thus, objectives 8 and 9 have been met.

Mathematical benchmark optimisation problems were solved using the PSO-Bees Algorithm. The algorithm converged to the optimum without becoming trapped at local

optima and outperformed other optimisation techniques in terms of the accuracy of the results obtained. Thus, objective 5 has been met.

Benchmark function optimisation problems were solved using the *improved* Bees Algorithm. The algorithm did not become trapped at local optima and outperformed other optimisation techniques in terms of the accuracy of the results obtained. Thus, objective 5 has been met.

Function optimisation problems and a number of deceptive multi-modal optimisation functions were solved using the SNTO-Bees Algorithm. The algorithm outperformed other optimisation techniques in terms of the accuracy of the results obtained. Thus, objectives 5 and 8 have been met.

The *improved* Bees Algorithm and the SNTO-Bees Algorithm were used as a multi-objective optimisation tool for complex optimisation problems. The tool was used to search for multiple Pareto optimal solutions in a mechanical engineering problem. Compared to two non-dominated Genetic Algorithms and the Bees Algorithm, the *improved* Bees Algorithm and the SNTO-Bees Algorithm were able to find more trade-off solutions. Thus, objective 8 has been met.

The PSO-Bees Algorithm required less tuning and search space sampling than the PSO Algorithm for the problems tested. Thus, objectives 4 and 5 have been met.

The *improved* Bees Algorithm and the SNTO-Bees Algorithm required less tuning and search space sampling than the original Bees Algorithm for the problems tested. Thus, objectives 5, 6 and 8 have been met.

Despite the high dimensionality of the control chart pattern recognition problem (each particle represented 2351 parameters that had to be determined), the PSO-Bees Algorithm succeeded in training more accurate classifiers than did the well-established BP algorithm. Likewise, for the wood defect classification problem (where each particle represented 1594 parameters that had to be determined), the PSO-Bees Algorithm trained classifiers were able to identify the defects more accurately than did classifiers trained using the original PSO Algorithm and the well-established back-propagation method. Experimental evidence demonstrates that the PSO-Bees Algorithm produced MLP networks with a lower total output error. Thus, objective 4 has been met.

Finally, the performances of all the three proposed optimisation algorithms were found to be better compared to their predecessors.

6.3 Further Work

This section suggests promising new directions for further research which can augment and enhance the proposed algorithms.

A major area of interest would be to make all the parameters of the PSO-Bees Algorithm adaptive. At the moment, the algorithm only incorporates adaptive neighbourhood search,

making all the parameters adaptive would enable the algorithm to be more effective and robust in handling dynamic multi-swarm / multi-modal / multi-objective optimisation problems or put simply, dynamically changing fitness landscape applications.

The proposed PSO-Bees Algorithm could focus on adjusting particle motion, making use of the Kalman Filter to update particle positions. This would enhance exploration without hurting the ability to converge rapidly to good solutions as proven by (Monson and Seppi 2004).

The ability of the PSO-Bees Algorithm can be enhanced to implement multi-swarms, where individual swarms work cooperatively together while exchanging vital information to solve optimisation problems.

In the application of the PSO-Bees Algorithm to training neural networks for the control chart pattern recognition and wood defect classification problems, the implementation of a better coding strategy can increase the classification and pattern recognition capabilities of the algorithm.

The proposed PSO-Bees Algorithm can be combined with other known effective global optimisers to improve the speed and accuracy in converging onto the global optima of objective functions without becoming trapped in local optima.

A reduction in the number of parameters of the PSO-Bees Algorithm without a corresponding negative influence on the performance of the algorithm would be ideal and welcomed.

The suggestions for further research on the PSO-Bees Algorithm are also applicable to the *improved* Bees Algorithm and the SNTO-Bees Algorithm.

There are now improved variants of the SNTO technique. Implementing the enhanced variants in the original Bees Algorithm and in the PSO Algorithm could be explored further.

Bibliography

Akhtar, S. and Tai, K. and Ray, T. 2002. A Socio-Behavioural Simulation Model for Engineering Design Optimisation. *Engineering Optimisation* 34(4), pp. 341 - 354.

Alcock, R. 1996. *Techniques for Automated Visual Inspection of Birch Wood Board.* University of Wales Cardiff, United Kingdom.

Aleksander, I. and Morton, H. 1990. *An Introduction to Neural Computing.* Chapman and Hall.

Altenberg, L. 1994. The Schema Theorem and Price's Theorem. *In Foundations of Genetic Algorithms 3*, pp. 23 – 49.

Angeline, P. J. 1998. Using Selection to Improve Particle Swarm Optimisation. *Proceedings of IEEE International Comference on Evolutionary Computation.* IEEE Press. pp. 84 - 89

Aragon, V. S. and Esquivel, S. C. 2004. An Evolutionary Algorithm to Track Changes of Optimum Value Locations in Dynamic Environments. *Journal of Computer Science & Technology* 4(3)(12).

Arora, J. S. 2004. *Introduction to Optimum Design.* New York: Elsevier.

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.* pp. 18 – 27

Bateson, W. 1909. Mendel's Principles of Heredity. Cambridge University Press.

Beaudoin, W. and Verel, S. and Collard, P. and Escazut, C. 2006. Deceptiveness and neutrality the ND family of fitness landscapes. *In GECCO'06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pp. 507 - 514.

Bergh, F. V. D. 2001. *An Analysis of Particle Swarm Optimisers.* University of Pretoria.

Bergh, F. v. d. and Engelbrecht, A. P. 2001. Effects of Swarm Size on Cooperative Particle Swarm Optimisers. *Proceedings of the Genetic and Evolutionary Computation Conference.* pp. 892 - 899

Besson, P. and Vesin, J. M. and Popovici, V. and Kunt, M. 2006. Differential Evolution Applied to a Multimodal Information Theoretic Optimisation Problem. *In Applications of Evolutionary Computing, Proceedings of the 8th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (evoIASP)*, pp. 505 – 509.

Beyer, H. 1994. Toward a Theory of Evolution Strategies: The (μ, λ) Theory. *Evolutionary Computation* 2(4), pp. 381 – 407.

Blackwell, T. 2007. Particle Swarm Optimisation in Dynamic Environments. *In Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 29 – 52.

Bonabeau, E. and Dorigo, M. and Theraulaz, G. 1999. *Swarm Intelligence: from Natural to Artificial Systems*. New York: Oxford University Press.

Bornberg-Bauer, E. and Chan, H. S. 1999. Modeling evolutionary landscapes: Mutational stability, topology, and superfunnels in sequence space. *Proceedings of the National Academy of Science of the United States of Americs (PNAS) - Biophysics 96.* Volume 19, pp. 10689 – 10694.

Branke, J. 1999. The Moving Peaks Benchmark.*Technical Report, Institute AIFB.* University of Karlsruhe, D-76128 Karlsruhe, Germany.

Branke, J. 2001. Evolutionary Optimisation in Dynamic Environments. *Genetic Algorithms and Evolutionary Computation.*

Branke, J. and Salihoglu, E. and Uyar, S. 2005. Towards an analysis of dynamic environments. *In GECCO'05: Proceedings of the 2005 conference on Genetic and Evolutionary Computation.* pp. 1433 – 1440

Brits, R. and Engelbrecht, A. P. and Bergh, F. v. d. 2002. A Niching Particle Swarm Optimiser. *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning.* pp. 692 - 696

Burke, E. and Gustafson, S. M. and Kendall, G. and Krasnogor, N. 2002a. Advanced Population Diversity Measures in Genetic Programming. *In Proceedings of the 7th International Conference on Parallel Problem Solving from Nature - PPSN VII*, p. 341.

Burke, E. K. and Gustafson, S. M. and Kendall, G. 2002b. Survey and Analysis of Diversity Measures in Genetic Programming. *In Proceedings of Genetic and Evolutionary Computation Conference.* pp. 716 – 723

Carlisle, A. and Dozier, G. 2000. Adapting Particle Swarm Optimisation to Dynamic Environments. *International Conference on Artificial Intelligence (ICAI)*, pp. 429 - 434.

Carlisle, A. J. and Dozier, G. 2002. Tracking Changing Extrema with Adaptive Particle Swarm Optimiser. *In Proceedings of the 5th Biannual World Automation Congress (WAC 2002)* Volume 13, pp. 265 – 270.

Castellani, M. 2008. *Optimisation Heuristics and Parameter Selection - Presentation.* Cardiff, South Wales: Manufacturing Engineering Centre, Cardiff University.

Černý, V. 1985. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimisation Theory and Applications* 45(1), pp. 41 - 51.

Chankong, V. and Haimes, Y. Y. 1983. *Multiobjective Decision Making Theory and Methodology*. North-Holland, Elsevier, Dover Publications, New York.

Cheng, C. S. 1995. A Multi-Layered Neural Network model for detecting changes in the process mean. *Computers and Industrial Engineering* 28(1), pp. 51 - 61.

Cheng, C. S. 1997. A Neural Network Approach for the Analysis of Control Chart Patterns. *International Journal of Production Research* 35(3), pp. 667 - 697.

Chinneck, J. W. 2000. *Practical Optimization: A Gentle Introduction.*

Clerc, M. 1999. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimisation. *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 1951 - 1957

Conners, R. W., Cho, T. C., Ng, T. C., Drayer, T. H. 1992. A Machine Vision System for Automatically Grading Hardwood Lumber. *Industrial Metrology 2*, pp. 317 - 341.

Conners, R. W., McMillin, C. W., Lin, K., Vasquez-Espinosa, R. E. 1983. Identifying and Locating Surface Defects in Woods. *IEEE Trans. on Pattern Analysis and Machine Intelligence* PAMI - 5/6 (Part of an Automated Lumber Processing System), pp. 573 - 583.

Davidor, Y. 1990. Epistasis Variance: A Viewpoint on GA-Hardness. *In Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pp. 23 – 35.

Davis, R. and King, J. 1977. An Overview of Production Systems. *Machine Intelligence* Volume 8.

Dawkins, R. 1987. The Evolution of Evolvability. *In ALIFE – Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. pp. 201 – 220

Deb, K. 1991. Optimal Design of a Welded Beam via Genetic Algorithm. *AIAA Journal* 29(11), pp. 2013 - 2015.

Deb, K. and Pratap, A. and Moitra, S. 2000. Mechanical Component Design for Multiple Objectives Using Elitist Non-Dominated Sorting GA. Kanpur, India: Indian Institute of Technology. p. 10

Deneubourg, J. and Goss, S. 1989. Collective Patterns and Decision-making. *Ethology, Ecology & Evolution* 1(4), pp. 295 - 311.

Deneubourg, J. and Pasteels, J. M. and Verhaeghe, J. C. 1983. Probabilistic Behaviour in Ants: A Strategy of Errors? *Journal of Theoretical Biology* 105(2), pp. 259 – 271.

Dietterich, T. 1995. Overfitting and undercomputing in machine learning. *ACM Computing Surveys (CSUR)* 27(3), pp. 326 - 327.

Digalakis, J. and Margaritis, K. 2004. Performance Comparison of Memetic Algorithms. *Journal of Applied Mathematics and Computation* Volume 158, pp. 237 - 252.

Dorigo, M. and Blum, C. 2005. Ant Colony Optimisation Theory: A Survey. *Theoretical Computer Science* 344(2-3), pp. 243 - 278.

Dorigo, M. and DiCaro, G. and Gambardella, L. 1998. Ant Algorithms for Discrete Optimisation.*Technical Report IRIDIA/98-10.* Universite Libre de Bruxelles, Belgium.

Dorigo, M. and Maniezzo, V. and Colorni, A. 1996. The Ant System: Optimisation by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics* 26(1), pp. 29 - 41.

Eberhart, R. and Shi, Y. 2001. Particle Swarm Optimisation Developments, Applications and Resources. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 81 - 86.

Eberhart, R. and Shi, Y. and Kennedy, J. 2001. *Swarm Intelligence.* San Francisco: Morgan Kaufmann.

Eberhart, R. C. and Kennedy, J. 1995. A New Optimiser using Particle Swarm Theory. *Proceedings of Sixth International Symposium on Micromachine and Human Science.* pp. 39 - 43

Eberhart, R. C. and Shi, Y. 2000. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimisation. *Proceedings of the Congress on Evolutionary Computation* Volume 1, pp. 84 - 89.

Eberhart, R. C. and Simpson, P. K. and Dobbins, R. W. 1996. *Computational Intelligence PC Tools* Academic Press Professional.

Eiben, A. E. and Schippers, C. A. 1998. On Evolutionary Exploration and Exploitation. *Fundamenta Informaticae* 35(1 - 4), pp. 35–50.

Engelbrecht, A. P. 2005. *Fundamentals of Computational Swarm Intelligence.* John Wiley and Sons Ltd, p. 672 pages.

Engelbrecht, A. P. 2006. Particle Swarm Optimisation: Where does it belong? *In Proceeding of the IEEE Swarm Intelligence Symposium*, pp. 48 - 54.

Eshelman, L. J. and Schaffer, J. D. 1991. Preventing Premature Convergence in Genetic Algorithms by Preventing Incest. *In Proceedings of the 4th International Conference on Genetic Algorithms (ICGA)*. pp. 115–122

Estévez, P. A. and Pérez, C. A. and Caballero, R. E. and Buhler, G. and Goles, E. 1998. Classification of Defects on Wood Boards Based on Neural Networks and Genetic Selection of Features. *Proceeding of 4th International Conference on Information Systems, Aealysis and Synthesis, ISAS'98* Volume 1, pp. 624 - 629.

Fang, K. T. and Hickernell, F. J. and Winker, P. 1996. Some Global Optimisation Algorithms in Statistics. *Operations Research and Its Applications, Lecture Notes in Operations Research* Volume 2, pp. 14 - 24.

Fang, K. T. and Wang, Y. 1990. A Sequential Algorithm for Optimisation and Its Application to Regression Analysis. *in Lecture Notes in Contemporary Mathematics*, pp. 17 - 28.

Fang, K. T. and Wang, Y. 1994. *Number-Theoretic Methods in Statistics*. Chapman & Hall.

Fang, K. T. and Wang, Y. and Bentler, P. M. 1994. Some Applications of Number-Theoretic Methods in Statistics. *Statistical Science* 9(3), pp. 416 - 428.

Fausett, L. V. 1994. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. Prentice Hall.

Felix, T. and Chan, S. and Tiwari, M. K. 2007. *Swarm Intelligence: Focus on Ant and Particle Swarm Optimisation*. I-TECH Education and Publishing, Vienna, Austria.

Fisher, R. A. 1918. The correlations between relatives on the supposition of Mendelian inheritance. *Philosophical Transactions of the Royal Society of Edinburgh* 52, pp. 399 - 433.

Frisch, K. V. 1976. *Bees: Their Vision, Chemical Senses and Language*. Revised Edition ed. Ithaca, N.Y.: Cornell University Press.

Fudenberg, D. and Tirole, J. 1991. *Game Theory*. The MIT Press.

Galperin, E. A. 1997. Pareto analysis vis-à-vis balance space approach in multi-objective global optimisation. *Journal of Optimisation Theory and Applications* 93(3), pp. 533 - 545.

Gan, F. and Xu, Q. and Zhang, L. and Liang, Y. 2001. An Improved Optimisation Strategy and Its Application to Clustering Analysis. *Analytical Sciences, The Japan Society for Analytical Chemistry* Volume 17, pp. 869 - 873.

Ghanbarzadeh, A. 2007. *THE BEES ALGORITHM - A Novel Optimisation Tool.* Cardiff University.

Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence - Computers and Operations Research.Elsevier Science Ltd. Oxford, UK, UK. pp. 533 - 549

Glover, F. 1989. Tabu Search, Part I - ORSA Journal on Computing 1.pp. 190 - 206

Glover, F. 1990. Tabu Search, Part II - ORSA Journal on Computing 2 pp. 4 - 32

Gobb, H. G. and Grefenstette, J. J. 1993. Genetic Algorithms for Tracking Changing Environments. *In Proceedings of 5th ICGA*, pp. 523 – 529.

Goss, S. and Beckers, R. and Deneubourg, J. and Aron, S. and Pasteels, J. M. 1990. How Trail Laying and Trail following can solve Foraging Problems for Ant Colonies.*In Behavioural Mechanisms of Food Selection, NATO ASI Series, G 20.* Springer-Verlag, Berlin. pp. 661 - 678

Grassé, P. 1959. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation des termites constructeurs. *Insectes Sociaux.* Paris: pp. 41 - 48

Greig, D. M. 1980. *Optimisation, Chapters 3 and 4.* New York: Longman Inc.

Guntsch, M. and Middendorf, M. 2001. Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. *In Proceedings of the EvoWorkshops on Applications of Evolutionary Computing*, pp. 213 - 222.

Gurin, L. S. and Rastrigin, L. A. 1965. Convergence of the Random Search Method in the presence of noise. *Automation and Remote Control* 26, pp. 1505 - 1511.

Hansen, P. 1986. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. *Presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.*

Haug, E. J. and Arora, J. S. 1979. *Applied Optimal Design.* New York: Wiley Interscience.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation*. 2nd ed. Upper Saddle River, N. J. Prentice Hall.

Haykin, S. and Bhattacharya, T. K. 1992. Adaptive Radar Detection Using Supervised Learning Networks.*Computational Neuroscience Symposium, Indiana University*. Purdue University at Indianapolis: pp. 35 - 51

Hendtlass, T. and Randall, M. 2001. A Survey of Ant Colony and Particle Swarm Meta-Heuristics and their Application to Discrete Optimisation Problems. *Proceedings of the Inaugural Workshop on Artificial Life*. pp. 15 - 25

Holland, J. H. and Burks, A. W. 1985. Adaptive Computing System Capable of Learning and Discovery. *Genetic Algorithm and Genetic Programming System 382/155, Learning Systems 706/62*.

Holland, J. H. and Reitman, J. S. 1977. Cognitive Systems Based on Adaptive Algorithms. *ACM SIGART Bulletin* 63(49).

Horst, R. and Tuy, H. 1990. *Global Optimisation – Deterministic Approaches*. 2nd ed. Springer, Berlin.

Huynen, M. A. 1996. Exploring Phenotype Space Through Neutral Evolution. *Journal of Molecular Evolution* 43(3), pp. 165 – 169.

Huynen, M. A. and Stadler, P. F. and Fontana, W. 1996. Smoothness within Ruggedness: The role of Neutrality in Adaptation. *Proceedings of the National Academy of Science of the United States of Americs (PNAS) - Evolution* 93, pp. 397 – 401.

Igel, C. and Toussaint, M. 2003. On classes of functions for which No Free Lunch results hold. *Information Processing Letters* 86(6), pp. 317 - 321.

Jacob, D. A. and Luke, S. R. 1993. Training Artificial Neural Networks for Statistical Process Control. *The Tenth Biennial University Government Industry Microelectronics Symposium IEEE*. Piscataway, USA: pp. 235 - 239

Jansen, T. and Wegener, I. 2007. A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-boolean functions of unitation. *Theoretical Computer Science* 386(1-2), pp. 73 - 93.

Kannan, B. K. and Kramer, S. N. 1994. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimisation and Its Applications to Mechanical Design. *Journal of Mechanical Design. Transactions of the ASME* 116, pp. 318 – 320.

Kauffman, S. A. 1993. *The Origins of Order: Self-Organisation and Selection in Evolution*. Oxford University Press.

Kennedy, J. and Eberhart, R. C. 1995a. Particle Swarm Optimisation. *In Proceedings of IEEE International Conference on Neural Networks* Volume 4, pp. 1942 – 1948.

Kennedy, J. and Eberhart, R. C. 1995b. Particle Swarm Optimization. *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE Press. pp. 1942 - 1948

Kennedy, J. and Eberhart, R. C. 1997. A Discrete Binary Version of the Particle Swarm Algorithm. *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*. pp. 4104 - 4109

Kennedy, J. and Spears, W. 1998. Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator. *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 78 - 83

Kirkpatrick, S. and Gelatt, C. D. and Vecchi, M. P. 1983. Optimisation by Simulated Annealing. *Science* 220(4598), pp. 671 - 680.

Kirschner, M. and Gerhart, J. 1998. Evolvability. *Proceedings of the National Academy of Science of the USA (PNAS)* 95(15), pp. 8420 – 8427.

Kohonen, T. 1988. The 'Neural' Phonetic Typewriter. *IEEE Computer Society* 27(3), pp. 11 - 22.

Koivo, A. J. and Kim, C. W. 1986. Classification of Surface Defects on Wood Boards. *IEEE International Conference on Systems, Man and Cybernetics, Atlanta, GA*, pp. 1431 - 1436.

Koivo, A. J., Kim, C. W. 1994. Hierarchical Classification of Surface Defects on Dusty Wood Boards. *Pattern Recognition Letters* Vol. 15, pp. 713 - 721.

Köppen, M. and Wolpert, D. H. and Macready, W. G. 2001. Remarks on a recent paper on the "No Free Lunch" Theorems. *IEEE Transactions on Evolutionary Computation* 5(3), pp. 295 - 296.

Korošec, P. and Šilc, J. 2006. Real-Parameter Optimisation Using Stigmergy. *In Proceedings of the Second International Conference on Bioinspired Optimisation Methods and their Application (BIOMA2006)*, pp. 73 - 84.

Krasnogor, N. and Smith, J. 2005. A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues. *IEEE Transactions on Evolutionary Computation* 9(5), pp. 474 - 488.

Lagraias, J. C. and Reeds, J. A. and Wright, M. H. and Wright, P. E. 1998. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimisation (SIOPT)* 9(1), pp. 112 - 147.

Lampinen, J. and Smolander, S. and Silven, O. and Kauppinen, H. 1994. Wood Defect Recognition: A Comparative Study. *Workshop on Machine Vision for Advanced Production, Oulu, Finland.*

Lawrence, S. and Giles, C. L. 2000. Overfitting and Neural Networks Conjugate Gradient and Backpropagation. *In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*. IEEE Computer Society. pp. 1114 – 1119

Leite, J. P. B. and Topping, B. H. V. 1998. Improved Genetic Operators for Structural Engineering Optimisation. *Advances in Engineering Software* 29(7-9), pp. 529 - 562.

Lewis, R. M. and Torczon, V. J. and Trosset, M. W. 2000. Direct Search Methods: Then and Now.*Technical Report NASA/CR-2000-210125 and ICASE Report No. 2000-26.* Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, USA.

Liepins, G. E. and Vose, M. D. 1991. Deceptiveness and Genetic Algorithm Dynamics. *In Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA).* pp. 36–50

Ling, C. X. 1995. Overfitting and generalisation in learning discrete patterns. *Neurocomputing* 8(3), pp. 341 - 347.

Liu, P. and Lau, F. and Lewis, M. J. and Wang, C. L. 2002. A New Asynchronous Parallel Evolutionary Algorithm for Function Optimisation. *In Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, pp. 401 - 410.

Luger, G. F. 2002. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*. Fourth ed. Harlow, England: Addison-Wesley.

Lush, J. L. 1935. Progeny Test and Individual Performance as indicators of an animal's breeding value. *Journal of Dairy Science* 18(1), pp. 1 - 19.

Manderick, B. and Moyson, F. 1988. The Collective Behaviour of Ants: an Example of Self-Organisation in Massive Parallelism.*In Proceedings of AAAI Spring Symposium on Parallel Models of Intelligence*. Stanford, California.

Mathur, M. and Karale, S. B. and Priye, S. and Jayaraman, V. K. and Kulkarni, B. D. 2000. Ant Colony Approach to Continuous Function Optimization. *Ind. Eng. Chem. Res.* 39(10), pp. 3814-3822.

Matyas, J. 1965. Random Optimisation. *Automation and Remote Control (AC)* 26(2), pp. 244 – 251.

McKinnon, K. I. M. 1999. Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimisation* 9(1), pp. 148 - 158.

Mendes, R. and Mohais, A. S. 2005. DynDE: A Differential Evolution for Dynamic Optimisation Problems. *In Proceedings of 2005 IEEE Congress on Evolutionary Computation* volume 3, pp. 2808 – 2815.

Metropolis, N. and Rosenbluth, A. W. and Rosenbluth, M. N. and Teller, A. H. and Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* 21(6), pp. 1087 - 1092.

Mezura-Montes, E. and Coello, C. A. C. and Landa-Becerra, R. 2003. Engineering Optimization Using a Simple Evolutionary Algorithm. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*. IEEE Computer Society.

Mezura-Montes, E. and Velàzquez-Reyes, J. and Coello, C. A. C. 2006. A Comparative Study of Differential Evolution Variants for Global Optimisation. *In GECCO'06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pp. 485 – 492.

Monson, C. K. and Seppi, K. D. 2004. The Kalman Swarm - A New Approach to Particle Motion in Swarm Optimisation. *GECCO* Volume 1, pp. 140 - 150.

Mori, N. and Imanishi, S. and Kita, H. and Nishikawa, Y. 1997. Adaptation to Changing Environments by Means of the Memory Based Thermodynamical Genetic Algorithm. *In Proceedings of 5th ICGA*, pp. 299 – 306.

Moriarty, D. E. and Schultz, A. C. and Grefenstette, J. J. 1999. Evolutionary Algorithms for Reinforcement Learning. *Journal of Artificial Intelligence Research* 11, pp. 241 - 276.

Morrison, R. W. 2004. Designing Evolutionary Algorithms for Dynamic Environments. *Natural Computing*. Springer, Berlin.

Morrison, R. W. and DeJong, K. A. 1999. A Test Problem Generator for Non-Stationary Environments. *In Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99)* Volume 3, pp. 2047–2053.

Moscato, P. 1989. On Evolution, Search, Optimisation, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Technical Report C3P 826, Caltech Concurrent Computation Program 158 - 79*. California Institute of Technology, Pasadena, CA 91125, USA.

Muttil, N. and Liong, S. 2004. Superior Exploration–Exploitation Balance in Shuffled Complex Evolution. *Journal of Hydraulic Engineering* 130(12), pp. 1202–1205.

Naudts, B. and Suys, D. and Verschoren, A. 2000. Generalised Royal Road Functions and their Epistasis. *Computers and Artificial Intelligence* 19(4).

Naudts, B. and Verschoren, A. 1996. Epistasis On Finite And Infinite Spaces. *In Proceedings of the 8th International Conference on Systems Research, Informatics and Cybernetics*, pp. 19 - 23.

Naudts, B. and Verschoren, A. 1999. Epistasis and Deceptivity. *Bulletin of the Belgian Mathematical Society.* 6(1), pp. 147 – 154.

Nelder, J. A. and Mead, R. A. 1965. A Simplex Method for function Minimisation. *Computer Journal* Volume 7, pp. 308 - 313.

Niederreiter, H. and Peart, P. 1986. Localisation of Search in Quasi-Monte Carlo Methods for Global Optimisation. *SIAM Journal on Science Statistics Computation* Volume 7, pp. 660 - 664.

Nigrin, A. 1993. Neural Networks for Pattern Recognition. *MIT Press*. p. 11

Nimwegen, E. v. and Crutchfield, J. P. and Huynen, M. 1999. Neutral evolution of mutational robustness. *Proceedings of the National Academy of Science of the United States of Americs (PNAS) - Evolution* 96(17), pp. 9716 – 9720.

Oei, C. K. and Goldberg, D. E. and Chang., S. 1991. Tournament selection, niching, and the preservation of diversity. In: (IlliGAL), I.G.A.L. ed. Illinois: Department of Computer Science, Department of General Engineering, University of Illinois at Urbana-Champaign.

Olsson, D. M. and Nelson, L. S. 1975. The Nelder-Mead Simplex Procedure for Function Minimisation. *Technometrics* 17(1), pp. 45 – 51.

Omran, M. and Salman, A. and Engelbrecht, A. P. 2002. Image Classification using Particle Swarm Optimisation. *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*. pp. 370 - 374

Packianather, M. S. and Drake, P. R. 2005. Comparison of Neural and Minimum Distance Classifiers in Wood Veneer Defect Identification. *IMechE Part B, J. of Engineering Manufacture* 219, pp. 831 - 844.

Packianather, M. S. and Drake, P. R. 2006. Post-processing Multilayered Feed-forward Neural Network Outputs for Identifying Wood Veneer Defects. In: Teti, R. ed. *5th CIRP*

International Seminar on Intelligent Computation in Manufacturing Engineering (CIRP ICME '06). Ischia, Italy: pp. 961 - 969

Pham, D. T. and Alcock, R. J. 1996. Automatic Detection of Defects on Birch Wood Boards. *Journal of Process Mechanical Engineering* Vol. 210 (Proc. I Mech E, Part E), pp. 45 - 52.

Pham, D. T. and Alcock, R. J. 1998a. Automated Grading and Defect Detection: A Review. *Forest Products Journal.* Volume 48, No. 4, pp. 34 - 42.

Pham, D. T. and Alcock, R. J. 1998b. Recent Developments in Automated Visual Inspection of Wood Boards. In: Tzafestas, S.G. ed. *In Advances in Manufacturing - Decision, Control and Information Technology.* Springer Verlag, Berlin and London, pp. 79 - 87.

Pham, D. T. and Castellani, M. and Sholedolu, M. and Ghanbarzadeh, A. 2008. The Bees Algorithm and Mechanical Design Optimisation. *ICINCO 2008 - International Conference on Informatics in Control, Automation and Robotics.*

Pham, D. T. and Ghanbarzadeh, A. 2006. Mechanical Components Optimal Design using the Bees Algorithm. Manufacturing Engineering Centre, Cardiff University.

Pham, D. T. and Ghanbarzadeh, A. 2007. Multi-Objective Optimisation using the Bees Algorithm.*Proceedings of 3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007).* Taylor and Francis. pp. 529 - 533

Pham, D. T. and Ghanbarzadeh, A. and Koc, E. and Otri, S. and Rahim, S. and Zaidi, M. 2005. Technical Note: Bees Algorithm.Cardiff: Manufacturing Engineering Centre, Cardiff University.

Pham, D. T. and Ghanbarzadeh, A. and Koc, E. and Otri, S. and Rahim, S. and Zaidi, M. 2006a. The Bees Algorithm, A Novel Tool for Complex Optimisation Problems. *Proc 2nd Int Virtual Conf on Intelligent Production Machines and Systems (IPROMS 2006).* Oxford: Elsevier.

Pham, D. T. and Liu, X. 1995. *Neural Networks for Identification, Prediction and Control.* London: Springer.

Pham, D. T. and Oztemel, E. 1992. Control Chart Pattern Recognition Using Neural Networks. *Journal of Systems Engineering*, pp. 256-262.

Pham, D. T. and Oztemel, E. 1996. *Intelligent Quality Systems.* Springer-Verlag, London, UK, p. 201.

Pham, D. T. and Soroka, A. J. and Ghandbarzadeh, A. and Koc, E. and Otri, S. and Packianather, M. 2006b. Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm. *INDIN'06 IEEE International Conference on Industrial Informatics*. Singapore: pp. 1346 - 1351

Phillips, P. C. 1998. The Language of Gene Interaction. *Genetics* 149(3), pp. 1167 - 1171.

Pitts, W. and McCulloch, W. S. 1943. A Logical Calculus of the Ideas Imminent in Nervous Activity. *Bulletin of Mathematical Biophysics* Volume 5, pp. 115 - 133.

Ragsdell, K. M. and Phillips, D. T. 1976. Optimal Design of a Class of Welded Structures using Geometeric Programming. *ASME Journal of Engineering for Industry* 98, pp. 1021 - 1025.

Rastrigin, L. A. 1963. The Convergence of the Random Search Method in the Extremal Control of Many-Parameter System. *Automation and Remote Control* 24, pp. 1337 - 1342.

Ratnaweera, A. and Halgamuge, S. and Watson, H. 2003. Particle Swarm Optimisation with Self-Adaptive Acceleration Coefficients. *Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery*. pp. 264 - 268

Ratnieks, F. L. W. 2008. How honey bee colonies track rewarding food patches. *Laboratory of Apiculture & Social Insects, Department of Animal & Plant Sciences, University of Sheffield.*

Rechenberg, I. 1994. Werkstatt Bionik und Evolutionstechnik. *Evolutionsstrategie '94*. Frommann Holzboog, Stuttgart.

Rekliatis, G. V. and Ravindrab, A. and Ragsdell, K. M. 1983. *Engineering Optimisation Methods and Applications*. New York: Wiley.

Reynolds, R. G. 1999. Cultural Algorithms: Theory and Application. In: Corne, D. *et al.* eds. *New Ideas in Optimisation*. McGraw-Hill, p. 367.

Richter, H. 2004. Behavior of Evolutionary Algorithms in Chaotically Changing Fitness Landscapes. *In Proceedings of 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pp. 111 - 120.

Ronald, S. 1995. Preventing diversity loss in a routing genetic algorithm with hash tagging. .Complexity International.

Ronald, S. 1996. *Genetic Algorithms and Permutation-Encoded Problems: Diversity Preservation and a Study of Multimodality.* University Of South Australia.

Rudnick, W. M. 1992. *Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks.* Oregon Graduate Institute of Science & Technology.

Rudolph, G. 1997. Convergence Properties of Evolutionary Algorithms. In: Kovac, D. ed. *Forschungsergebnisse zur Informatik.* Vol. 35. Hamburg, Germany.

.

Rudolph, G. 1999. Self-adaptation and global convergence: A counter-example. *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC99.* Vol. 1. pp. 646–651.

Rudolph, G. 2001. Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation* 5(4), pp. 410–414.

Russell, S. J. and Norvig, P. 2002. Artificial Intelligence: A Modern Approach. *Prentice Hall, Second Edition.*

Rychtyckyj, N. and Reynolds, R. G. 1999. Using Cultural Algorithms to Improve Performance in Semantic Networks. *In Proceeding IEEE Congress on Evolutionary Computation* Volume 3, pp. 1651 - 1656.

Salerno, J. 1997. Using the Particle Swarm Optimisation Technique to Train a Recurrent Neural Model. *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence.* IEEE Press. pp. 45 - 49

Sarle, W. 1997. What is overfitting and how can I avoid it? *Usenet FAQs: comp.ai.neural-nets FAQ, 3: Generalisation* 3.

Sarle, W. S. 1995. Stopped Training and Other Remedies for Overfitting.*In Proceedings of the 27th Symposium on the Interface: Computing Science and Statistics.* pp. 352 – 360

Schaffer, J. D. and Eshelman, L. J. and Offutt, D. 1990. Spurious Correlations and Premature Convergence in Genetic Algorithms. *In Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA).* pp. 102–112

Schalkoff, R. J. 1997. *Artificial Neural Networks.* McGraw-Hill Series in Computer Science.

Schoonderwoerd, R. and Holland, O. E. and Bruten, J. L. and Rothkrantz, L. J. M. 1996. Ant-Based Load Balancing in Telecommunications Networks. *Adaptive Behaviour* 5(2), pp. 169 - 207.

Schumer, M. A. 1965. Optimisation by Adaptive Random Search. *PhD thesis, Princeton University, NJ.*

Schumer, M. A. and Steiglitz, K. 1968. Adaptive Step Size Random Search. *IEEE Transactions on Automatic Control (AC)* 13(3), pp. 270 - 276.

Schutte, J. F. and Groenwold, A. A. 2003. Sizing Design of Truss Structures using Particle Swarms. *Structural and Multidisciplinary Optimisation* 24(4), pp. 261 - 269.

Seeley, T. D. 1996. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Cambridge, Massachusetts: Harvard University Press.

Seeley, T. D. and Visscher, P. K. and Passino, K. M. 2006. Group Decision Making In Honey Bee Swarms. *Americal Scientist* Volume 94, pp. 220 - 229.

Shekel, J. 1971. Test Functions for Multimodal Search Techniques. *In Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems.* Princeton, NJ, USA:

Shi, Y. and Eberhart, R. C. 1998a. A Modified Particle Swarm Optimiser. *Proceedings of the IEEE Congress on Evolutionary Computation.* IEEE Press. pp. 69 - 73

Shi, Y. and Eberhart, R. C. 1998b. Parameter Selection in Particle Swarm Optimisation.*Proceedings of the Seventh Annual Conference on Evolutionary Programming.* pp. 591 - 600

Shi, Y. and Eberhart, R. C. 2001. Fuzzy Adaptive Particle Swarm Optimisation.*Proceedings of the IEEE Congress on Evolutionary Computation.* IEEE Press. pp. 101 - 106

Shigley, J. E. 1977. *Mechanical Engineering Design.* New York: McGraw-Hill.

Shipman, R. and Shackleton, M. and Ebner, M. and Watson, R. 2000. Neutral Search Spaces for Artificial Evolution: A lesson from life. *Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life, Bradford Books, Complex Adaptive Systems.*

Siddall, J. N. 1972. *Analytical Decision-making in Engineering Design.* New Jersey: Prentice-Hall.

Singh, G. and Deb, K. 2006. Comparison of multi-modal optimisation algorithms based on evolutionary algorithms. *In Proceedings of the 8th Annual Conference on Genetic and evolutionary computation (GECCO'06).* pp. 1305–1312

Smith, R. E. 1992. A Report on The First International Workshop on Learning Classifier Systems (IWLCS-92).

Smith, S. S. 2004. Using multiple genetic operators to reduce premature convergence in genetic assembly planning. *Computers in Industry* 54(1), pp. 35 – 49.

Smith, T. and Husbands, P. and Layzell, P. and O'Shea, M. 2002. Fitness Landscapes and Evolvability. *Evolutionary Computation* 10(1), pp. 1 - 34.

Solis, F. and Wets, R. 1981. *Minimisation by Random Search Techniques*. Mathematics of Operations Research, p. 672 pages.

Spotts, M. F. 1971. *Design of Machine Elements*. 4th ed. Englewood Cliffs; [Hemel Hempstead] : Prentice-Hall.

Steuer, R. E. 1989. *Multiple Criteria Optimisation: Theory, Computation and Application*. Krieger Pub Co, reprint edition.

Stewart, T. 2001. Extrema Selection: Accelerated Evolution on Neutral Networks. *In Congress on Evolutionary Computation. Volume 1.*

Stickland, T. R. and Tofts, C. M. N. and Franks, N. R. 1992. A Path Choice Algorithm for Ants. *Naturwissenschaften* 79(12), pp. 567 - 572.

Storn, R. and Price, K. 1995. Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimisation over Continuous Spaces. *Technical Report TR-95-012*. International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704, Berkeley, CA.

Suganthan, P. N. 1999. Particle Swarm Optimiser with Neighbourhood Operator. *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 1958 - 1962

Taguchi, G. 1986. Introduction to Quality Engineering: Designing Quality into Products and Processes. *Asian Productivity Organisation / American Supplier Institute Inc. / Quality Resources / Productivity Press Inc.,* .

Tetko, I. V. and Livingstone, D. J. and Luik, A. I. 1995. Neural network studies, 1. Comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences* 35(5), pp. 826 - 833.

Théraulaz, G. and Bonabeau, E. 1995. Coordination in Distributed Building. *Science* 269(5224), pp. 686 - 668.

Thierens, D. and Goldberg, D. E. and Pereira, A. G. 1998. Domino convergence, drift and the temporal-salience structure of problems. *In Proceedings of the International Conference on Evolutionary Computation*. pp. 535–540

Velasco, T. and Rowe, R. 1993. Back Propagation Artificial Neural Networks for the Analysis of Quality Control Charts. *Computers and Industrial Engineering* 25(1-4), pp. 397 - 400.

Vesterstrom, J. S. and Riget, J. and Krink, T. 2002. Division of Labour in Particle Swarm Optimisation. *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 1570 - 1575

Wagner, A. 2005. Robustness, Evolvability, and Neutrality. *FEBS Lett* 579(8), pp. 1772 – 1778.

Weise, T. 2008. *Global Optimisation Algorithms - Theory and Application*. Second ed. Thomas Weise, Online ebook, p. 728.

Weise, T. 2009. Evolving Distributed Algorithms with Genetic Programming.*Distributed Systems Group*. Kassel: University of Kassel.

Werfel, J. and Nagpal, R. 2006. Extended Stigmergy in Collective Construction. *IEEE Intelligent Systems* 21(2), pp. 20 - 28.

Werra, D. d. and Hertz, A. 1989. Tabu Search Techniques: A tutorial and an application to neural networks - OR Spektrum. pp. 131 - 141.

Wilke, C. O. 2001. Adaptive Evolution on Neutral Networks. *Bulletin of Mathematical Biology* 63(4), pp. 715 - 730.

Wilke, D. N. and Kok, S. and Groenwold, A. A. 2007. Comparison of linear and classical velocity update rules in particle swarm optimisation: notes on diversity. *International Journal for Numerical Methods in Engineering* 70(8), pp. 962 – 984.

Williams, G. C. 1957. Pleiotropy, Natural Selection, and the Evolution of Senescence. *Evolution* 11(4), pp. 398 – 411.

Wolpert, D. H. and Macready, W. G. 1997. No Free Lunch Theorems for Optimisation. *IEEE Transactions on Evolutionary Computation* 1(1), pp. 67 - 82.

Zheng, Y. and Ma, L. and Zhang, L. and Qian, J. 2003. Emperical Study of Particle Swarm Optimiser with Increasing Inertia Weight. *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 221 - 226

Zielinski, K. and Laur, R. 2007. Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimisation Algorithm. *Informatica* 31, pp. 51 - 59.

Zitzler, E. and Deb, K. and Thiele, L. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation. *Evolutionary Computation* 8(2), pp. 173 - 195.

Zurada, J. M. and Malinowski, A. and Usui, S. 1997. Perturbation Method for Deleting Redundant Inputs of Perceptron Networks. *Neurocomputing*. Vol. 14. Elsevier, pp. 177 - 193.

Appendix A

Glossary

ACO: Ant Colony Optimisation is an optimisation algorithm inspired by the research on real ants and simulation experiments for problems that can be reduced to finding optimal paths in graphs (based on the metaphor of ants seeking for food).

APSO: Adaptive PSO is a variant of the PSO Algorithm with additional adaptive adjustments to the parameters of the PSO algorithm.

BA: Bees Algorithm is a new optimisation algorithm developed by the researchers at the Manufacturing Engineering Centre (MEC) of Cardiff University after observing the "waggle dance" of bees foraging for nectar.

BF: Benchmark Functions are mathematical problems used to demonstrate the utility of global optimisation algorithms. These problems usually have no direct real-world application but are well understood, widely researched and are used to measure speed / ability of the optimiser, derive theoretical results just to mention a few.

CCPR: Control Charts are a graphical display of a quality characteristic that has been measured from a sample versus the sample number or time. The chart contains a centre line (CL) that represents the average value and the upper (UCL) and lower (LCL) lines allow variation limits of the quality characteristic under consideration.

Pattern Recognition is the process of extracting information from an unknown data stream or signal and assigning it to one of the prescribed classes or categories.

CLPSO: Comprehensive Learning PSO is a variant of the PSO Algorithm in which the conventional equation for the velocity update is modified to include a learning probability. The algorithm uses a different value for each particle to give them different levels of exploration and exploitation abilities.

Converged: Convergence is a term loosely used to indicate an algorithm has reached the point where it does not appear to make any further positive progress.

CPSO: Cooperative PSO is a variant of the original PSO algorithm that employs cooperative behavior in order to significantly improve the performance of the original PSO algorithm. It uses multiple swarms to optimise different components of the solution vector cooperatively.

DC: Domino Convergence occurs when the solution candidates have features which contribute to significantly different degrees to the total fitness.

DCFL: Dynamically Changing Fitness Landscape is used to describe a non stationary fitness landscape. An optimum in iteration t will no longer be an optimum in iteration $t + 1$.

DE, DES: Differential Evolution is a method for mathematical optimisation of multidimensional functions belonging to the group of evolution strategies. It has

proven to be a very reliable optimisation strategy for many different tasks where parameters are encoded in real vectors.

Deceptiveness: Deceptiveness is one of the major causes of problems in optimisation upsetting features of the fitness landscapes. The gradient deceptive objective function leads the optimiser away from the optima.

DEPSO: Differential Evolution PSO is a variant of the PSO Algorithm that combines and alternates the original PSO Algorithm and the DE operator.

Diversity: Diversity preservation is a major concern in optimisation because the loss of it can lead to premature convergence to a local optimum. Also, exploitation and exploration are directly linked to diversity: exploration increases diversity whereas exploitation works against it

DNPSO: Dynamic Neighbourhood PSO is a variant of the PSO Algorithm. The dynamic neighbourhood method for solving multi-objective optimisation problems modifies the PSO algorithm to locate the Pareto front.

DPSO: Dissipative PSO, a variant of the PSO Algorithm introduces negative entropy to stimulate the model in PSO, creating a dissipative structure that prevents premature stagnation.

DS: Downhill Simplex method or the Nelder-Mead method or amoeba method is a commonly used nonlinear optimisation algorithm; a numerical method for minimising an objective function in a many dimensional space of n-dimensional real

vectors.

EA: Evolutionary Algorithms are generic, population-based meta-heuristic optimisation algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, recombination, migration, locality, neighbourhood and survival of the fittest.

EC: Evolutionary Computation is a subfield of artificial intelligence that involves combinatorial optimisation problems. Evolutionary computation uses iterative progress, such as growth or development in a population. The population is then selected in a guided random search using parallel processing to achieve the desired end.

EP: Evolutionary Programming are poles apart compared to the other major types of evolutionary algorithms though there is a semantic difference: while single individuals of a species are the biological metaphor for solution candidates in other evolutionary algorithms, in evolutionary programming, a solution candidate is thought of as a species.

Epistasis: Epistasis in biology is defined as a form of interaction between different genes. The term meant that one gene suppresses the phenotypical expression of another gene. In the context of statistical genetics, *epistasis* was originally called "epistacy". The interaction between genes is *epistatic* if the effect on the fitness from altering one gene depends on the allelic state of other genes.

EPSO: Hybrid of Evolutionary Programming and PSO Algorithm is a variant of the

PSO Algorithm that incorporates a selection procedure into the original PSO algorithm, as well as self-adapting properties for its parameters.

ES: Evolution Strategies is a heuristic optimisation technique based on the ideas of *adaptation* and *evolution*, a special form of evolutionary algorithms.

Evolvability: Evolvability is a metaphor in global optimisation borrowed from biological systems. A biological system is evolvable if it is able to generate heritable, selectable phenotypic variations. Such properties can then be evolved and changed by natural selection. In its second sense, a system is evolvable if it can acquire new characteristics via genetic change that help the organism(s) to survive and to reproduce.

Exploitation: Exploitation in terms of optimisation means trying to improve the currently known solution(s) by performing small changes which lead to new individuals very close to them.

Exploration: Exploration in terms of optimisation means finding new points in areas of the search space which has not yet been investigated.

GA: Genetic Algorithm is an optimisation algorithm that view learning as a competition among a population of evolving candidate problem solutions. A 'fitness' function evaluates each solution to decide whether it will contribute to the next generation of solutions.

GAPSO: Hybrid of Genetic Algorithm and PSO is a variant of the PSO Algorithm

combines the advantages of swarm intelligence and a natural selection mechanism, the GA, in order to increase the number of highly evaluated agents, while also decreasing the number of lowly evaluated agents at each iteration step

GO: Global Optimisation Algorithm is an optimisation algorithm that locates the global maximum (or minimum) of the objective function through out the problem search space.

GPSO: Gaussian PSO is a variant of the PSO Algorithm. The Gaussian function is introduced to guide the movements of the particles. In this variant, the inertia weight constant is no longer needed and the acceleration coefficient constant is replaced by random numbers with Gaussian distributions.

HC: Hill climbing is an optimisation technique belonging to the family of local search and it is quite easy to implement. The hill climbing is a simple search optimisation algorithm for single objective functions f. In principle, hill climbing algorithms perform a loop in which the currently known best solution individual p^* is used to produce one offspring p_{new}. If this new individual is better than its parent, it replaces it and the cycle starts all over again and it is similar to an evolutionary algorithm with a population size of 1

HPSO: Hybrid PSO is a term loosely used to refer to the incorporation of other methods that have already been tested in other evolutionary computation techniques. The growth and improvement of the particle swarm algorithm is credited to the incorporations; some of which includes selection, mutation and crossover as well as

the differential evolution (DE) into the PSO Algorithm.

LCS: Learning Classifier Systems is a patented new class of cognitive systems that are a special case of production systems with close links to reinforcement learning and genetic algorithms consisting of four major parts: a set of interacting productions, called classifiers; a performance algorithm that directs the action of the system in the environment; a simple learning algorithm that keeps track on each classifier's success in bringing about rewards; and a more complex algorithm, called the genetic algorithm that modifies the set of classifiers so that variants of good classifiers persist while new potentially better ones are created in an efficient manner.

LO: Local Optimisation Algorithm is an optimisation algorithm that locates the maximum (or minimum) of a region $B \subset S$. The local minimum is not always the minimum of the search space S, it is merely the minimum of the region B, where B is defined to contain a single minimum.

MA: Memetic Algorithms are a family of optimisation methods that simulates cultural evolution rather than the biological one. Memetic Algorithms represents one of the recent growing areas of research in evolutionary computation. The term MA is widely used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search.

MLP: Multi-Layer Perceptron is a network composed of more than one layer of neurons, with some or all of the outputs of each layer connected to one or more of the inputs of another layer. The first layer is called the input layer, the last one is the output layer,

and in between there may be one or more hidden layers.

MOPSO: Multi-Objective Particle Swarm Optimisation is a variant of the PSO Algorithm developed for multi-objective optimisation problems consist of several objectives that need to be achieved simultaneously based on the Pareto optimality concept.

MSE: Mean Square Error is a metric used to compute, amongst other use, the difference between the output of a Neural Network and the desired output value that is specified in the data set.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (O_i^{actual} - O_i^{desired})^2$$

Neutrality: Neutrality is a word loosely used to describe the outcome of a search operation to a solution candidate if it yields no change in the objective values.

NFL: No Free Lunch is a theorem which helps to validate the notion that there is no optimisation algorithm that can outperform all others on all problems. There is a variety of optimisation methods specialised in solving different types of problems as well as algorithms that deliver good results for a many different problem classes, but are outperformed by highly specialised methods in each of them.

NN: Neural Network is a configurable mapping between an input space and an output space and these networks can represent an arbitrary mapping through adjustment of weights.

Noise: Noise is unwanted or unnecessary information corrupting or affecting the quality of data. There are two types of noise in optimisation: There is noise in the training data that is used as basis for learning which cause overfitting. This noise results because no measurement is 100% accurate and noise always exists when we try to fit a model to measured data. The second form of noise subsumes the perturbations that are likely to occur in the subsequent process – reason why the best robust solutions and not just the globally optimal ones are needed.

NS: Neighbourhood Size in this research opus refers to the area around the selected promising candidates designated for neighbourhood search.

OC: Optimality Conditions refers the solutions found by optimisation algorithms usually classified by its quality. The two main types of optimality conditions are local optima or global optima.

OF: Objective Function refers to the function that is optimised during the optimisation process, to compute either the set of parameters yielding the maximum (or the minimum) function value.

OPC: Optimisation Problem Classification is used to identify the various characteristics used to classify optimisation problems.

Overfitting: Overfitting is the emergence of an overly-complicated model (solution candidate) in an optimisation process resulting from the effort to provide the best possible results for as much of the available training data as possible.

Oversimplification: Oversimplification (also called overgeneralisations) is the opposite of overfitting.

PC: Premature Convergence - An optimisation process has prematurely converged to a local optimum if it is no longer able to explore other parts of the search space than the currently examined area and there exists another region that contains a solution superior to the currently exploited one.

PM: Performance Measure is a term used to measure and assess the performance of optimisation algorithms on six fronts: accuracy, reliability, robustness, efficiency, diversity and coherence. They represent a useful tool / means for checking the effectiveness / efficiency of the optimisation algorithm.

PSO: Particle Swarm Optimisation (PSO) Algorithm is a population-based stochastic optimisation technique developed by Eberhart and Kennedy and inspired by the social behaviour of bird flocking or fish schooling. PSO has its roots in artificial life and social psychology, as well as in engineering and computer science. It utilises a "population" of particles that fly through the problem hyperspace with given velocities.

PSO-Bees: PSO-Bees Algorithm is a variant of the PSO Algorithm that combines the fast convergence property of the PSO Algorithm and the inherent ability of the Bees Algorithm to avoid been trapped in local optima.

PSOPC: PSO with Passive Congregation is a variant of the PSO Algorithm using passive congregation, a mechanism that allows animals to aggregate into groups; employed

as a possible alternative to prevent the PSO algorithm from being trapped in local optima and to improve its accuracy and convergence speed.

Redundancy: Redundancy in the context of global optimisation is a feature of the genotype-phenotype mapping and means that multiple genotypes map to the same phenotype (the genotype-phenotype mapping is not injective).

SA: Simulated annealing is a generic probabilistic meta-algorithm for global optimisation problems locating good approximation to the global optimum of a given function in a large search space.

SNTO: Sequential Number-Theoretic Optimisation Algorithm is a recently new global optimization method popularly used in statistics with initial studies conducted to introduce this method into chemistry. SNTO is attractive due to its simplicity, ease of implementation and effective optimisation performance.

SNTO-Bees Algorithm: A novel Bees Algorithm combining the SNTO technique and the Bees Algorithm.

SPSO: Stretching PSO is a variant of the PSO Algorithm that is oriented towards solving the problem of finding all global minima. This PSO variant employs the deflection and stretching techniques, as well as a repulsion technique. The first two techniques (deflection and stretching) apply the concept of transforming the objective function by incorporating the already found minimum points. The latter (repulsion technique) adds the ability to guarantee that all particles will not move toward the already found minima.

TF: Test Functions are benchmark problems used to demonstrate the utility of global optimisation algorithms. These problems usually have no direct real-world application.

TS: Tabu Search is a mathematical optimisation method, belonging to the class of local search techniques. It enhances the performance of the local search method by using memory structures: once a potential solution has been determined, it is marked as "taboo" ("tabu" being a different spelling of the same word) so that the algorithm does not visit that solution repeatedly.

T-TEST: T-Tests are tests for statistical significance used with interval and ratio level data. T-tests are often employed in several different types of statistical tests:

- to test whether there are differences between two groups on the same variable, based on the mean (average) value of that variable for each group;
- to test whether a group's mean (average) value is greater or less than some standard;
- to test whether the same group has different mean (average) scores on different variables.

VEPSO: Vector Evaluated Particle Swarm Optimisation algorithm, a variant of the PSO Algorithm is based on the concept of the vector evaluated genetic algorithm (VEGA). In the VEPSO algorithm, two or more swarms are used in order to search the problem hyperspace. Each swarm is evaluated according to one of the objective functions and the information is exchanged between them. The knowledge coming from other swarms is used to guide each particle's trajectory towards the Pareto optimal points.

WDC: Wood Defect Classification refers to the extraction of features from different wood images containing known defect types or no defects and distinguishing between the features of those images.

Appendix B

Definition of Symbols

This appendix provides the list of all the symbols used in this research opus; presented on chapter basis.

Chapter 2 and Appendix G

$f(x_0) \leq f(x)$	Minimisation expression
$f(x_0) \geq f(x)$	Maximisation expression
$f(x)$	Objective function
$\|\| x - x^* \|\|$	Norm of $(x - x^*)$
R^n	Euclidean space
A	Feasible solution
x^*	Local minimum
R	Real domain
Z	Integer domain
x^*_B	Local minimiser
S	Search space
B	Proper subset of S
$S = R^n$	Unconstrained problem
z_0	Starting point
$B \subseteq S$	Set of feasible points in the neighbourhood of x^*_B

g_1, g_2	genotype
gpm	genotype-phenotype mapping
(x_i, y_i)	training data samples
A_t	set of training data
A_c	set of test cases
t	iteration at time t
G	Search space
g	Element of G
X	Problem space
$N(x[i], \sigma_i^2)$	Normal distribution
p^*	Best known solution individual
p_{new}	Offspring
$(G \subseteq R^n)$	Many dimensional space of n-dimensional real vectors
x_1 and x_2	Solution candidates
w	Weight
(n)	Number of scout bees
(m)	Number of sites selected out of n visited sites
(e)	Number of best sites out of m selected sites
(nep)	Number of bees recruited for best e sites
$(m\text{-}e)$	Number of bees recruited for the other selected sites (nsp)
(ngh)	Initial size of patches
(V_{n+1})	Particle velocity in iteration $(n+1)$
V_n	Particle velocity in iteration n

P_n	Particle position (solution) in iteration n
(P_{n+1})	Particle position (solution) in iteration $(n+1)$
p_i and $Pbest_n$	"Personal" best position in iteration n
p_g^* and $Gbest_n$	"Global" best position in iteration n
$rand_1$ and $rand_2$	Random numbers between 0 and 1
$c1$ and $c2$	Weighting factors
w	Inertia weight in the PSO Algorithm
◌	Representing *Pbest* in the diagram of the operations of the PSO Algorithm. Figure 2.6
○	Representing *Gbest* in the diagram of the operations of the PSO Algorithm. Figure 2.6
⟶	Arrow showing the direction and flight of particles in the diagram of operations of the PSO Algorithm. Figure 2.6
♦ ♠ * ♣ ● ■	List of particles in the diagram of the operations of the PSO Algorithm. Figure 2.6
$child_{1,2}(x)$, $Child_{1,2}(v)$	Offspring of breeding process
S_T	Selection rate
CR	Crossover value
k	random integer value within [1, n]
δ_2	Difference vector
Δ	Difference between two elements randomly chosen in the p_{best} set
$x(0)$	Particle position
$v(0)$	Initial velocity

$f(x_i(t))$	Fitness function value for particle i at iteration t
F_1 and F_2	Groups into which multiple objectives are divided
Index j	Swarm number ($j = 1, 2 \ldots M$)
Index i	Particle number ($i = 1, 2 \ldots N$)
$\chi^{[j]}$	Constriction factor of swarm j
$\varphi^{[j]}$	Inertia weight of swarm j
$p_i^{[j]}$	Best position found by particle in swarm j
$p_g^{[s]}$	Best position found for any particle in swarm s
$\lvert p_i(t-1) - p_g(t-1) \rvert$	Distance between global and local best
Grand(y)	Zero-mean Gaussian random number with standard deviation of y
Rand(θ)	Random vector with magnitude of one with angle uniformly distributed from zero to 2π
abs[N(0,1)]	Gaussian probability distribution
Rand(l_d, u_d)	Random number with predefined lower and upper limits
$\bar{x}$	Detected local minimum
sgn(y)	Triple valued sign function
Pc	Learning probability
ps	Population size

Chapter 3

wV_n	Momentum
P_{rand}	Random particle

P_{neigh}	Neighbourhood particle
$P_{Selected\ Candidate}$	Promising selected particle including the Gbest
◌	Representing *Pbest* in the diagram of the operations of the PSO-Bees Algorithm. Figure 3.2
◎	Representing G*best* in the diagram of the operations of the PSO-Bees Algorithm. Figure 3.2
◌	Representing neighbourhood size in the diagram of the operations of the PSO-Bees Algorithm. Figure 3.2
♦ ♠ * ♣ ● ■ ♥ ◘	List of particles in the diagram of the operations of the PSO-Bees Algorithm. Figure 3.2
φ	Summation of c1 and c2
x*	Theoretical optimum
Φ	Average of the performance criterion over a number of simulation runs
σ_{Φ}	Variance in the performance criterion
$S(t)$	Coherence
$e_s(t)$	Speed of swarm centre
$\ddot{e}(t)$	Average particle speed
s	Number of particles
O_i^{actual}	Actual output
$O_i^{desired}$	Desired output
N	Total number of training patterns
UCL	Upper control limit

LCL	Upper control limit
CL	Centre line
$\bar{y}(t)$	Scaled pattern value
y_{min}	Minimum allowed value
y_{max}	Maximum allowed value
μ	Mean value of the process variable being monitored (taken as 80 in this work)
σ	Standard deviation of the process (taken as 5)
a	Amplitude of cyclic variations (taken as 15 or less)
g	Magnitude of the gradient of the trend (taken as being in the range 0.2 to 0.5)
k	Parameter determining the shift position (= 0 before the shift position; = 1 at the shift position and thereafter)
r	Normally distributed random number (between – 3 and +3)
s	Magnitude of the shift (taken as being in the range 7.5 to 20)
t	Discrete time at which the pattern is sampled (taken as being within the range 0 to 59)
T	Period of a cycle (taken as being in the range 4 to 12 sampling intervals)
$y(t)$	Sample value at time t
$X = \mathbb{R}^n$	Real vectors

Chapters 4 & 5

$f(x)$	objective function
g	gravitational constant
g_i	constraint i
G	shear modulus
h	weld thickness
l	weld length
L	fixed distance from load to support
m	number of sites selected
M	mass
N	number active coils
nep	number of bees recruited for the best e sites
ngh	initial size of each patch
nsp	number of bees recruited for the other (m-e) selected sites
P	applied axial load
P_c	bar buckling load
Q	number of inactive coils
t	beam thickness
x	a scalar or a vector
x_{ie}	position of an elite bee in the i^{th} dimension
γ	weight density
δ	beam end deflection
Δ	minimum spring deflection

ρ	mass density
σ	maximum normal stress in beam
σ_d	allowable normal stress for beam material
τ	maximum shear stress in weld
τ'	primary stress
τ''	secondary stress
τ_d	allowable shear stress
ω	frequency of surge waves
ω_0	lower limit on surge wave frequency
b	beam width
c_1	Unit volume of weld material cost
c_2	Unit volume of bar stock cost
d	wire diameter
D	mean coil diameter
D_0	limit on outer diameter of the coil
e	number of top-rated (elite) sites
F	load
f	Cost function including setup cost

Appendix C

Abbreviations

AC:	Acceleration Coefficient
ACO:	Ant Colony Optimisation
APSO:	Adaptive PSO
AVIS:	Automated Visual Inspection System
BA:	Bees Algorithm
BF:	Benchmark Function
CC:	Constriction Coefficient
CCPR:	Control Chart Pattern Recognition
CLPSO:	Comprehensive Learning Particle Swarm Optimisation
CPSO:	Cooperative Particle Swarm Optimisation
DC:	Domino Convergence
DCFL:	Dynamically Changing Fitness Landscape
DE:	Differential Evolution

DEPSO:	Differential Evolution Particle Swarm Optimisation
DNPSO:	Dynamic Neighbourhood Particle Swarm Optimisation
DPSO:	Dissipative Particle Swarm Optimisation
DS:	Downhill Simplex
EA:	Evolutionary Algorithm
EARL:	Evolutionary Algorithms for Reinforcement Learning
ERL:	Evolutionary reinforcement learning
EC:	Evolutionary Computation
EP:	Evolutionary Programming
EPSO:	Evolutionary Particle Swarm Optimisation
ES:	Evolutionary Strategy
GA:	Genetic Algorithm
GAPSO:	Genetic Algorithm Particle Swarm Optimisation
GO:	Global Optimisation
GPSO:	Gaussian Particle Swarm Optimisation
HC:	Hill Climbing

HPSO:	Hybrid Particle Swarm Optimisation
IW:	Inertia Weight
LCS:	Learning Classifier System
LO:	Local Optimisation
MA:	Memetic Algorithm
MDC:	Minimum Distance Classifier
MLP:	Multi-Layer Perceptron
MOPSO:	Multi-Objective Particle Swarm Optimisation
MSE:	Mean Square Error
NFL:	No Free Lunch theorem
NN:	Neural Network
NS:	Neighbourhood Size
OC:	Optimality Condition
OF:	Objective Function
OPC:	Optimisation Problem Classification
PC:	Premature Convergence

PM:	Performance Measures
PSO:	Particle Swarm Optimisation
PSO-Bees:	Particle Swarm Optimisation-Bees Algorithm
PSOPC:	Particle Swarm Optimisation with Passive Congregation
RO:	Random Optimisation
SA:	Simulated Annealing
SIMPSA:	Deterministic Simplex method
SNTO:	Sequential Number-Theoretic Optimisation Algorithm
SNTO-Bees:	Sequential Number-Theoretic Optimisation (SNTO)-Bees Algorithm
SPC:	Statistical Process Control
SPSO:	Stretching Particle Swarm Optimisation
SPPSO:	Small Population Particle Swarm Optimisation
SZ:	Swarm Size
TF:	Test Function
TS:	Tabu Search

VC:	Velocity Clamping
VEPSO:	Vector Evaluated Particle Swarm Optimisation
WDC:	Wood Defect Classification

Appendix D

PSO Neighbourhood Topologies

This appendix provides a listing of PSO neighbourhood topologies (Engelbrecht 2005).

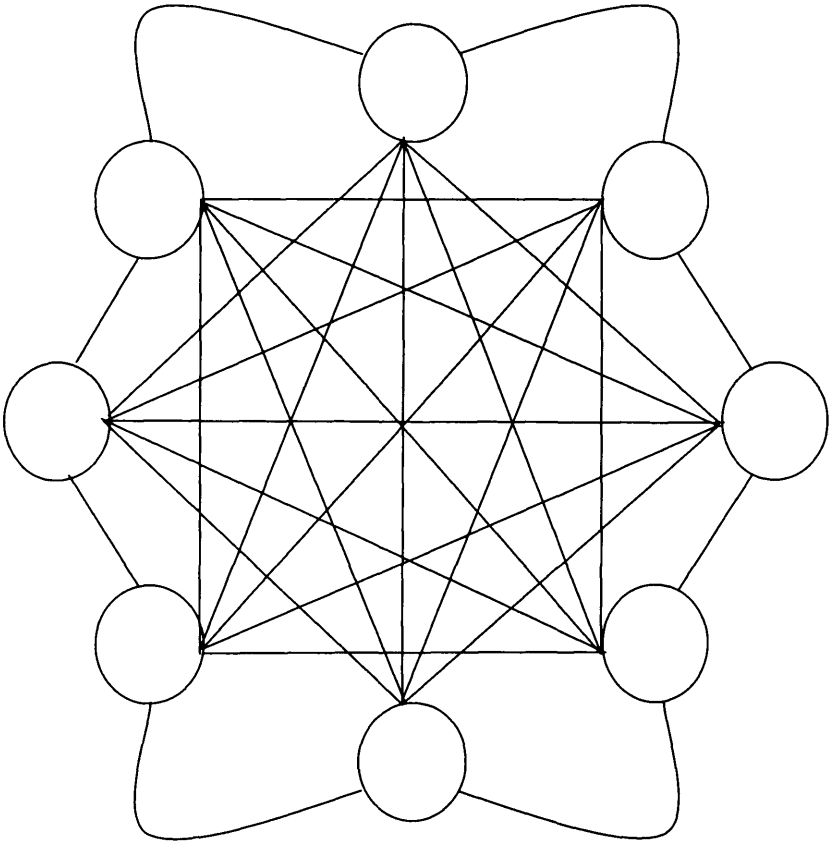


Figure D1: Graphical representation of the Star neighbourhood topology

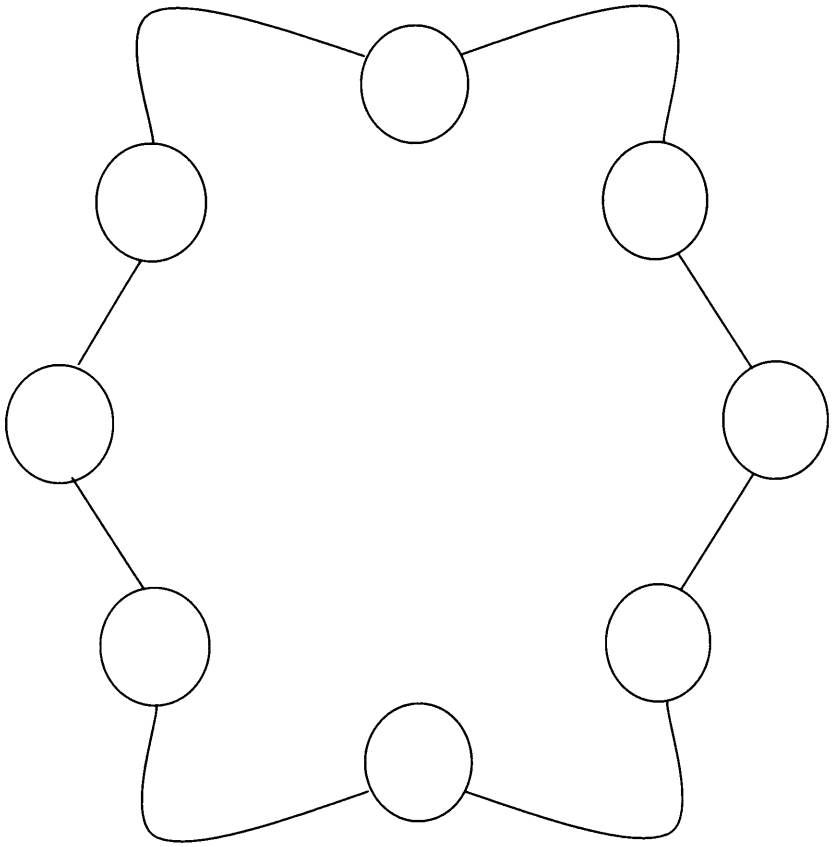


Figure D2: Graphical representation of the ring neighbourhood topology

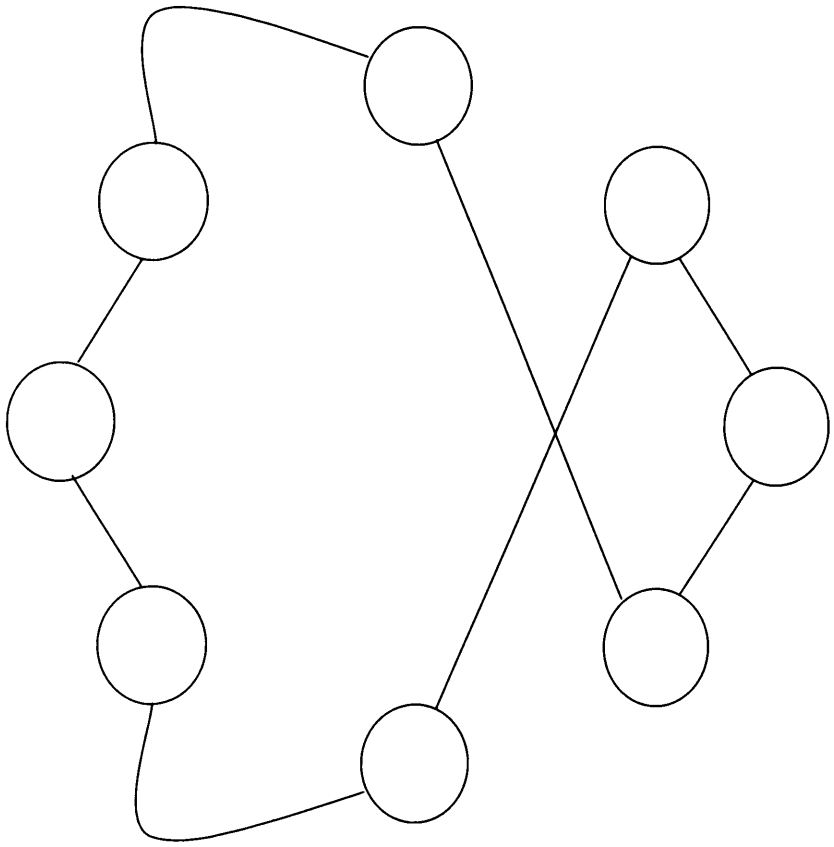


Figure D3: Graphical representation of the randomised ring neighbourhood topology

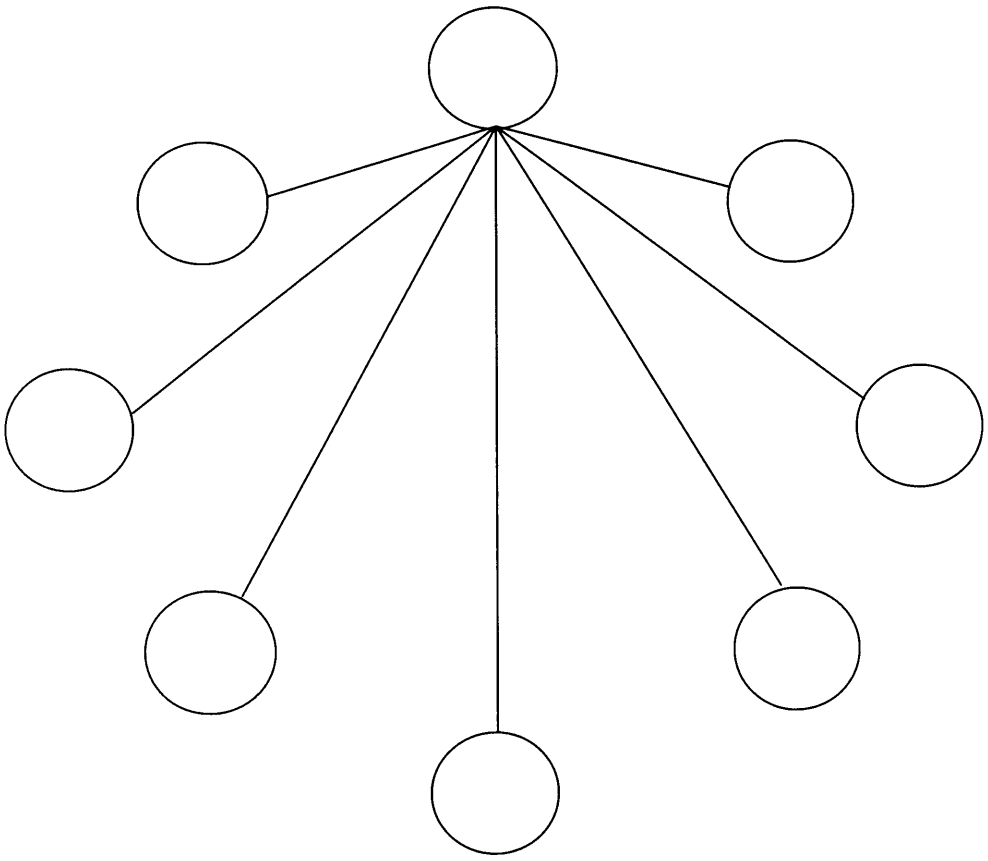


Figure D4: Graphical representation of the Wheel neighbourhood topology

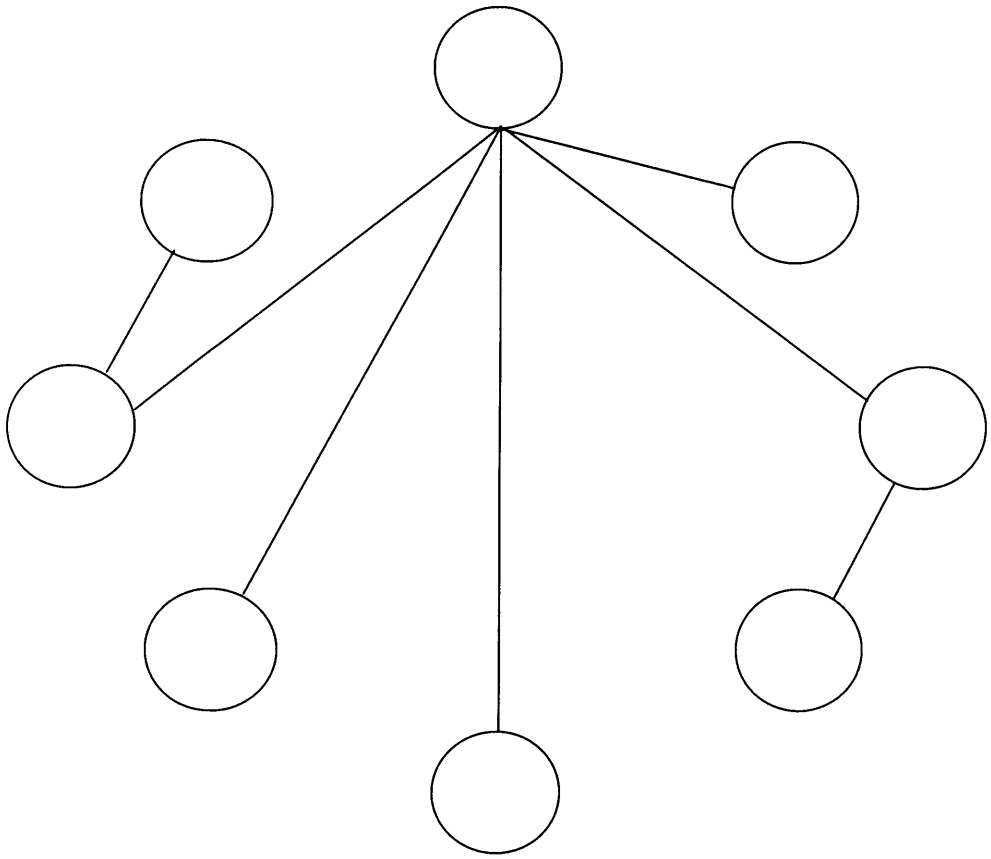


Figure D5: Graphical representation of the randomised wheel topology

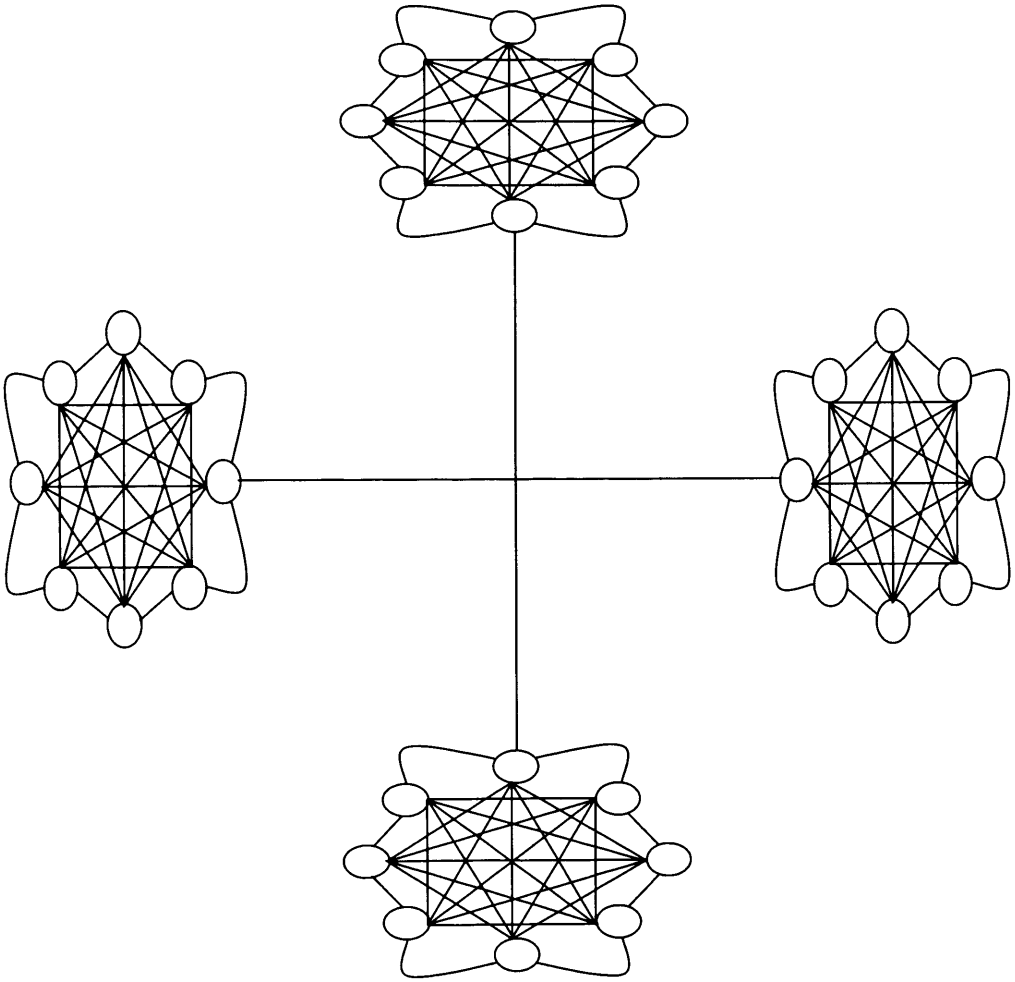


Figure D6: Graphical representation of the Four Clusters topology

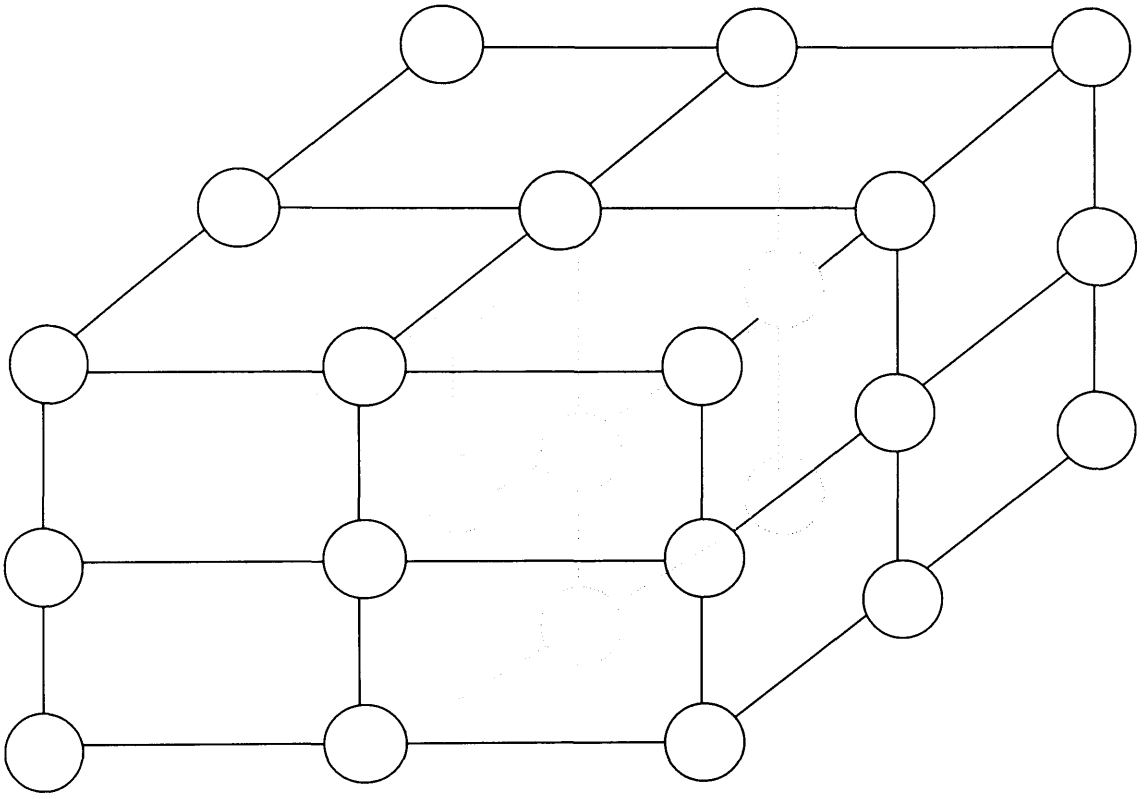


Figure D7: Graphical representation of the Von Neumann topology

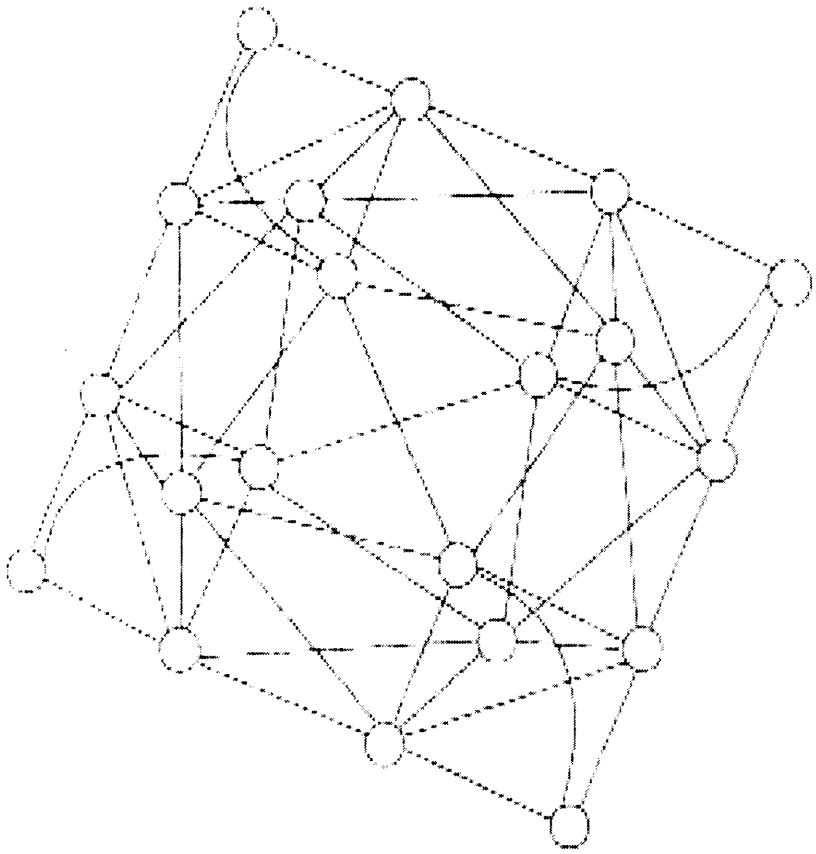


Figure D8: Graphical representation of the pyramid topology

Appendix E

Function Landscape

This appendix presents three dimensional plots of all the mathematical benchmark test functions used in Chapters 4, 5 and 6 to test the Hybrid PSO-Bees Algorithm, the improved Bees Algorithm and the SNTO-Bees Algorithm respectively. These functions were plotted based on their respective definitions in the mentioned chapters.

1. **De Jong's function** also known as sphere model is a continuous, convex and unimodal function.

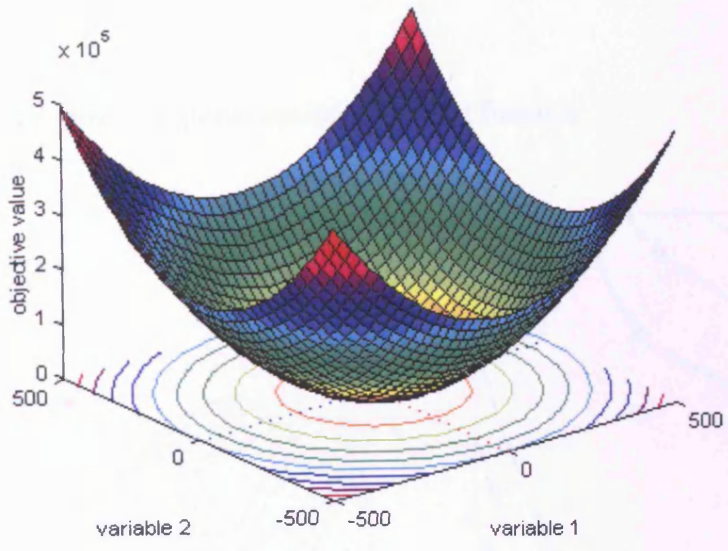


Figure E1: Visualisation of De Jong's function

2. **The Goldstein-Price function** is a global optimisation test function

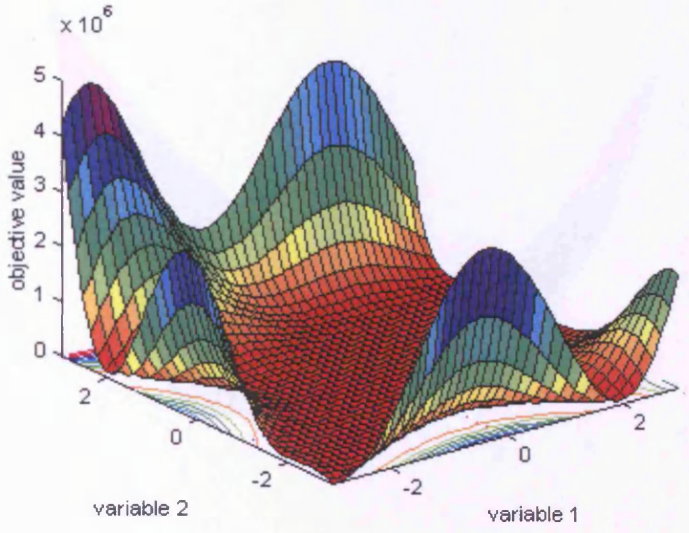


Figure E2: Visualisation of Goldstein-Price function

3. **The Branin function** is a global optimisation test function.

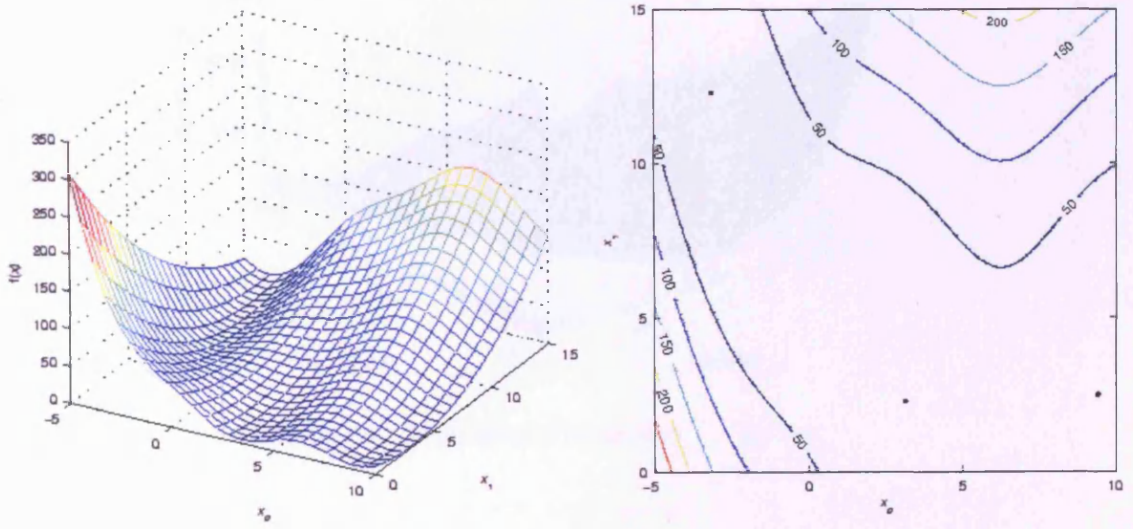


Figure E3: Visualisation of Branin function

4. Martin & Gaddy

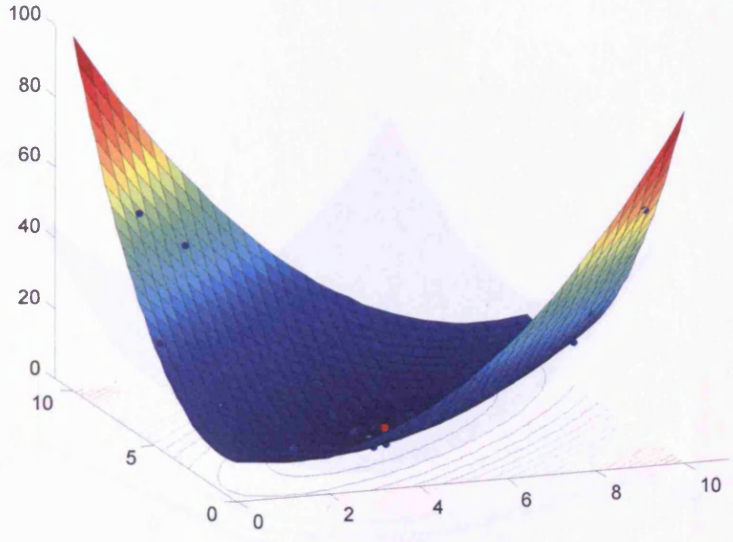


Figure E4: Visualisation of Martin & Gaddy function

5. Rosenbrock - 1

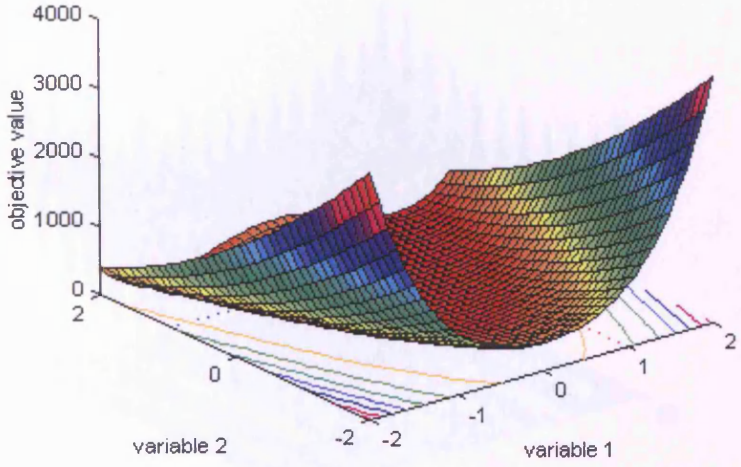


Figure E5: Visualisation of Rosenbrock – 1 function

6. **Griewangk function:** (E6a) full definition area from –500 to 500, (E6b): inner area of the function from –50 to 50, (E6c): area from –8 to 8 around the optimum at [0, 0]

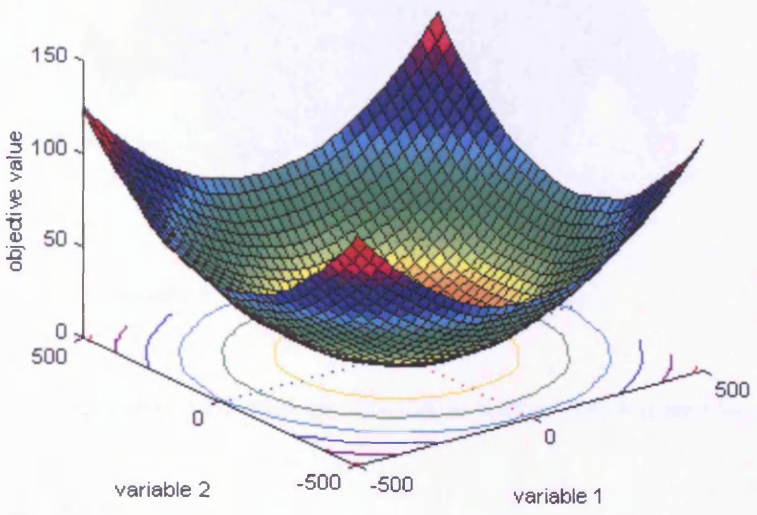


Figure E6a: Visualisation of Griewangk function (full definition area -500 to 500)

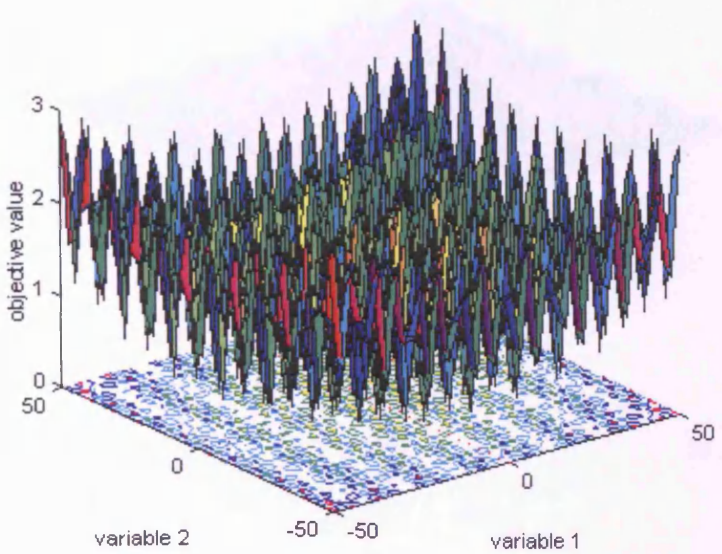


Figure E6b: Visualisation of Griewangk function (inner area -50 to 50)

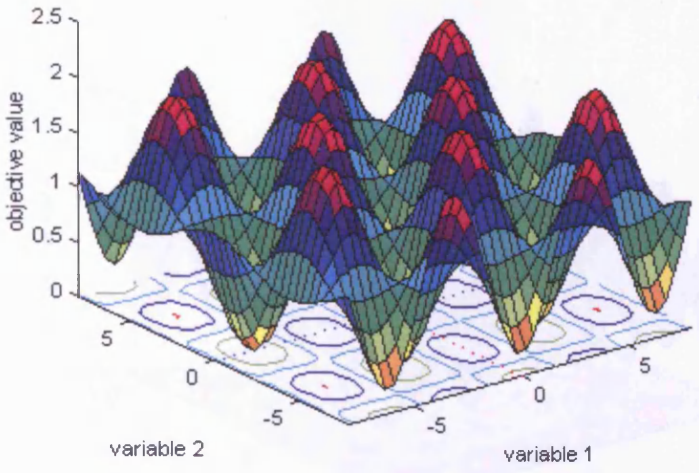


Figure E6c: Visualisation of Griewangk function (area from –8 to 8 around the optimum at [0, 0])

7. Ackley function.

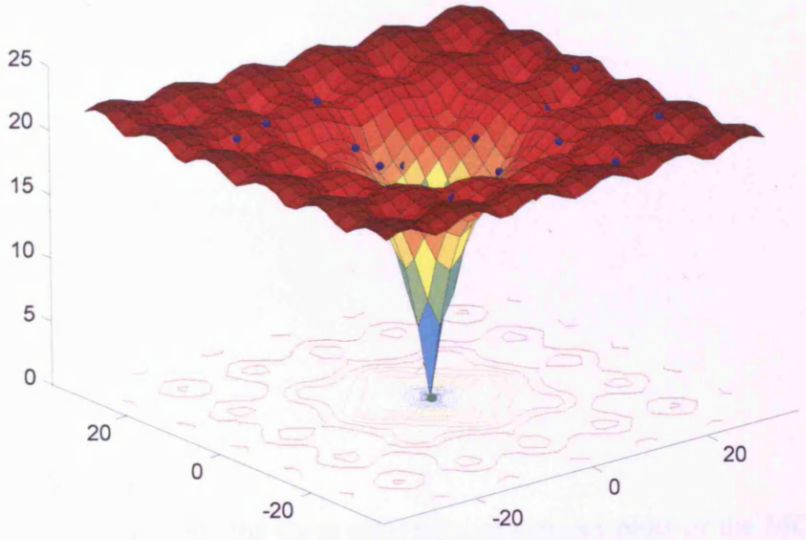


Figure E7: Visualisation of Ackley function

8. Schwefel function.

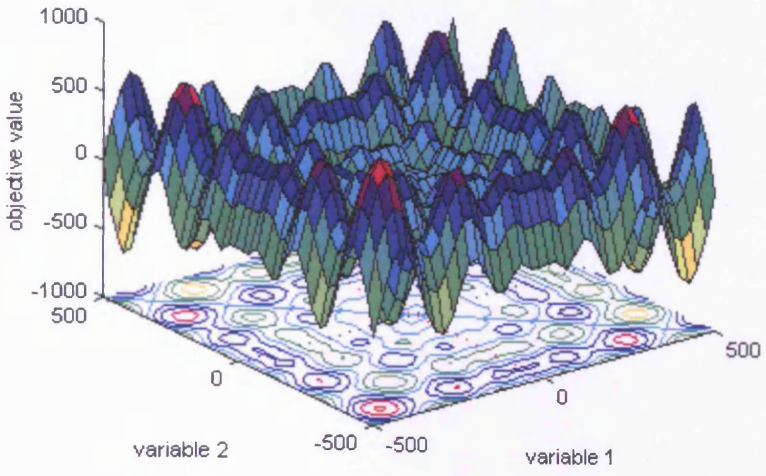


Figure E8: Visualisation of Schwefel function

The following pages presents the three dimensional contour plots of the MCastellani Test Functions 1 through 10 used in Chapter 5 to assess the SNTO-Bees Algorithm.

9. MCastellani TF 1

$$f(x_1, x_2) = \begin{cases} (x_1 \geq 0 \;\&\; x_2 \geq 0) \Rightarrow 1.0 - 0.75 \cdot e^{-\left(\frac{(x_2 - 50)^2}{50}\right)} \\ (x_1 < 0 \;\&\; x_2 < 0) \Rightarrow 1.0 - 1.0 \cdot \left(\frac{x_2 + 100}{70}\right) \cdot e^{-\left(\frac{(x_1 + x_2 + 130)^2}{25}\right)} \end{cases}$$

$$-100 \leq x_1 \leq 100$$
$$-100 \leq x_2 \leq 100$$
$$f(-100, -30) = 0$$

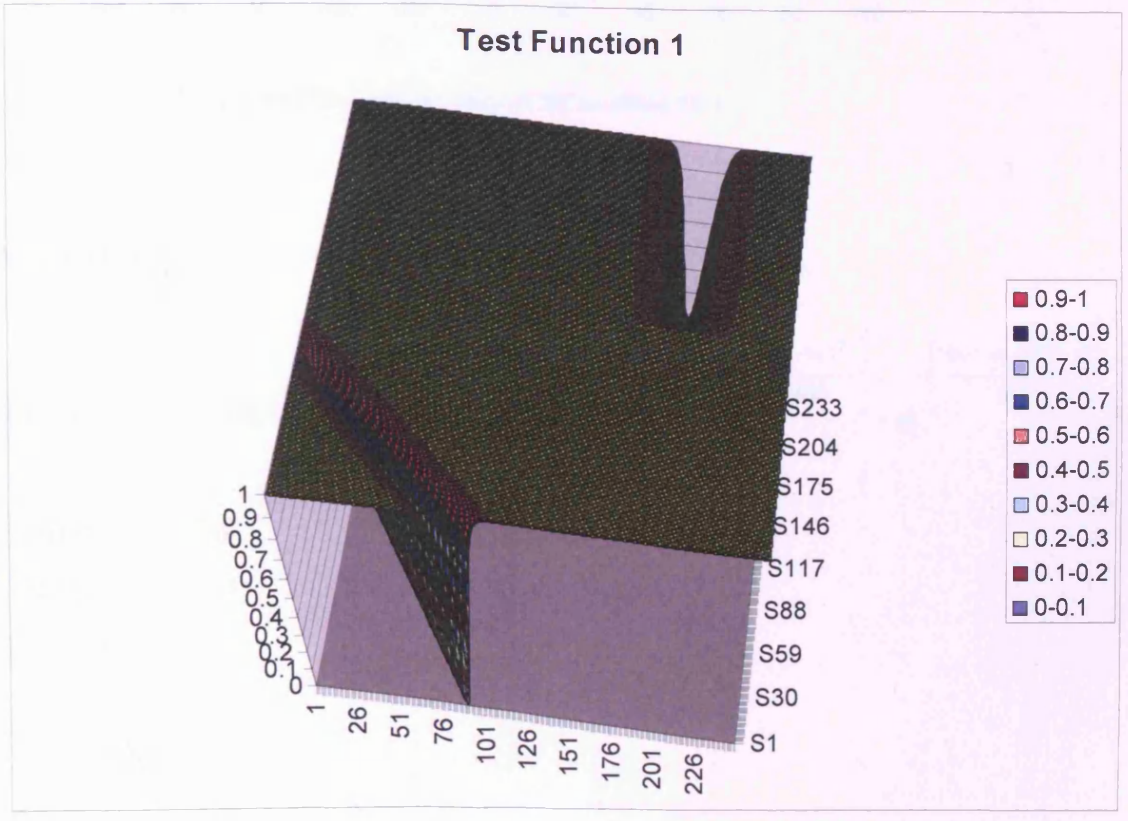


Figure E9a: Visualisation of MCastellani TF 1

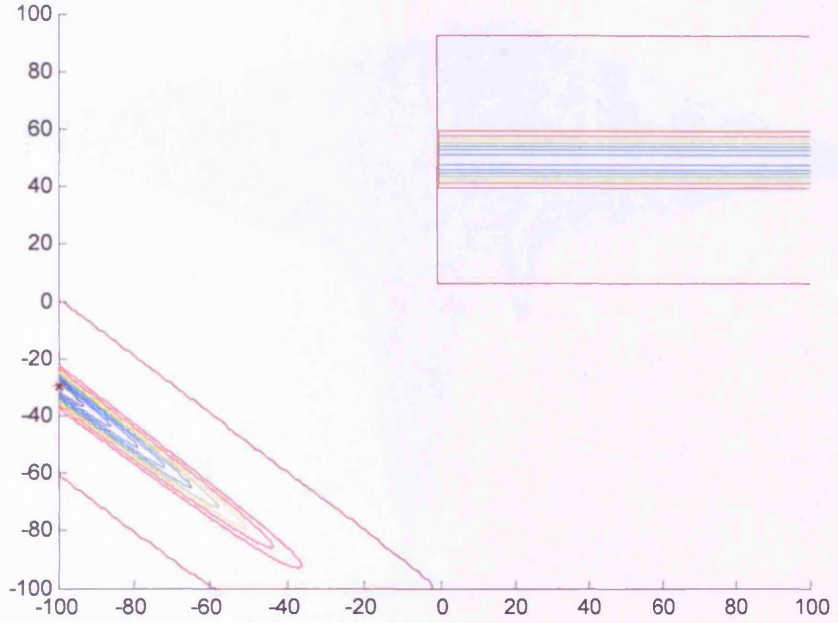


Figure E9b: Contour plots of MCastellani TF 1

10. MCastellani TF 2

$$f(x_1, x_2) = 1.0 - 0.75 \cdot e^{-\left(\frac{\left((x_1-60)^2+(x_2-40)^2\right)}{100}\right)} - 1.0 \cdot e^{-\left(\frac{(x_1-x_2)^2}{400}\right)} \cdot e^{-\left(\frac{(x_1+x_2+120)^2}{50}\right)}$$

$$-100 \le x_1 \le 100$$
$$-100 \le x_2 \le 100$$
$$f(-60,-60) = 0$$

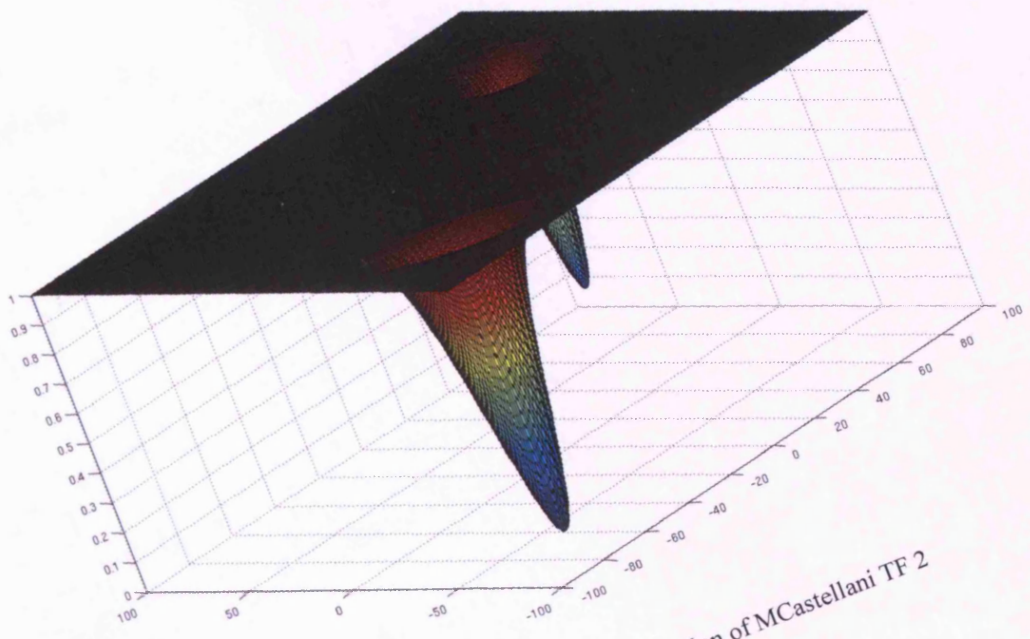


Figure E10a: Visualisation of MCastellani TF 2

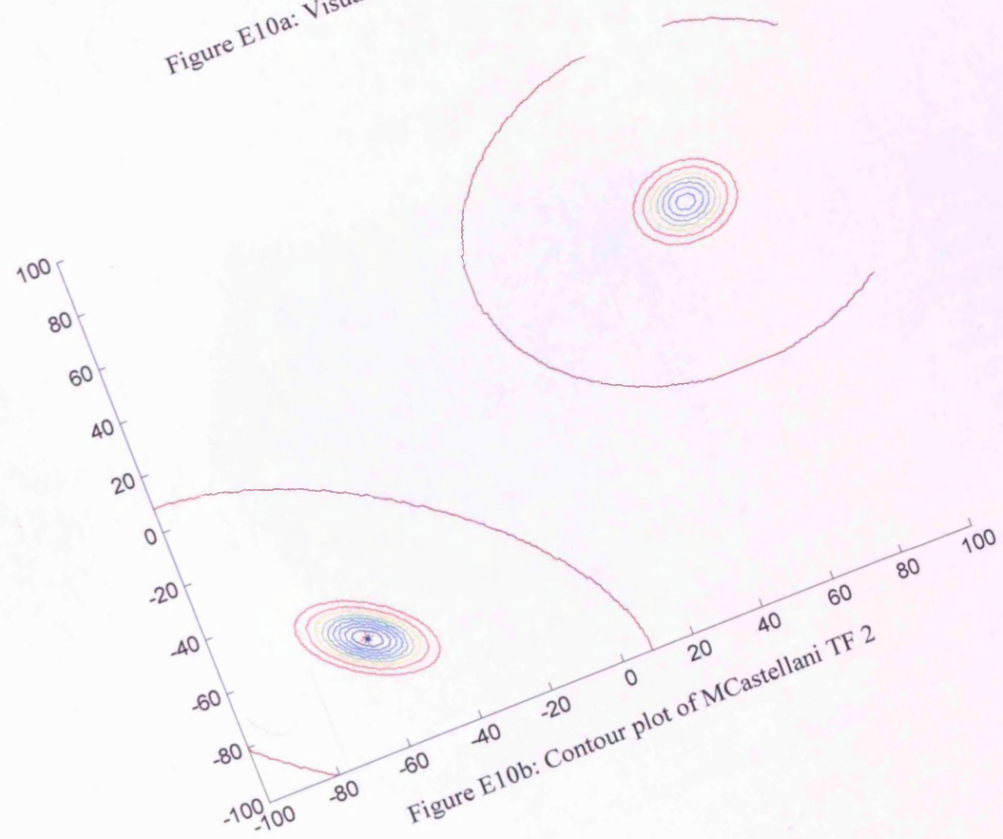


Figure E10b: Contour plot of MCastellani TF 2

11. MCastellani TF 3

$$f(x_1, x_2) = 1.0 +$$

$$- 1.0 \cdot e^{-\left(\frac{(x_1 - 75)^2 + (x_2 - 75)^2}{100}\right)^3}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1 + 60)^2 + (x_2 + 60)^2}{500}\right)}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1 + 60)^2 + (x_2 - 60)^2}{500}\right)}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1 - 60)^2 + (x_2 + 60)^2}{500}\right)}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1 + 0)^2 + (x_2 + 0)^2}{500}\right)}$$

$$-100 \leq x_1 \leq 100$$

$$-100 \leq x_2 \leq 100$$

$$f(75, 75) = 0$$

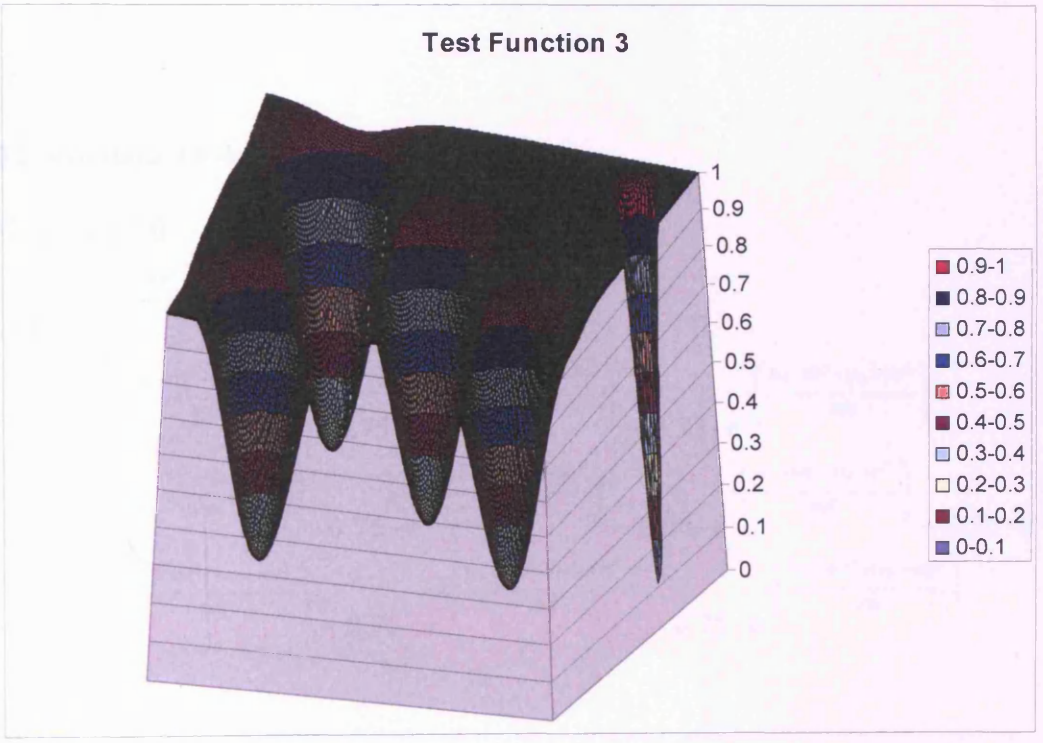


Figure E11a: Visualisation of MCastellani TF 3

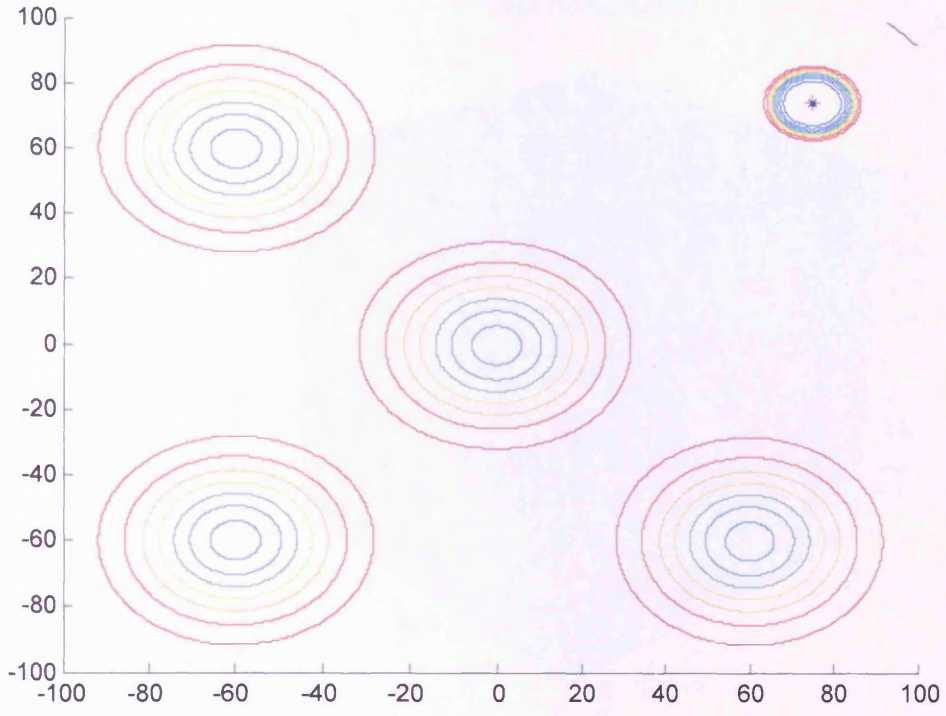


Figure E11b: Contour plot of MCastellani TF 3

12. MCastellani TF 4

$$f(x_1, x_2) = 1.0$$

$$- 1.0 \cdot e^{-\left(\frac{(x_1-80)^2+(x_2-80)^2}{100}\right)^4}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1+50)^2+(x_2+50)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1+50)^2+(x_2-50)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1-50)^2+(x_2+50)^2}{400}\right)}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1+0)^2+(x_2+50)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1+0)^2+(x_2-50)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1+50)^2+(x_2+0)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1-50)^2+(x_2+0)^2}{400}\right)}$$

$$- 0.75 \cdot e^{-\left(\frac{(x_1+100)^2+(x_2+100)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1-100)^2+(x_2+100)^2}{400}\right)} - 0.75 \cdot e^{-\left(\frac{(x_1+100)^2+(x_2-100)^2}{400}\right)}$$

$$-100 \le x_1 \le 100$$

$$-100 \le x_2 \le 100$$

$$f(80,80) = 0$$

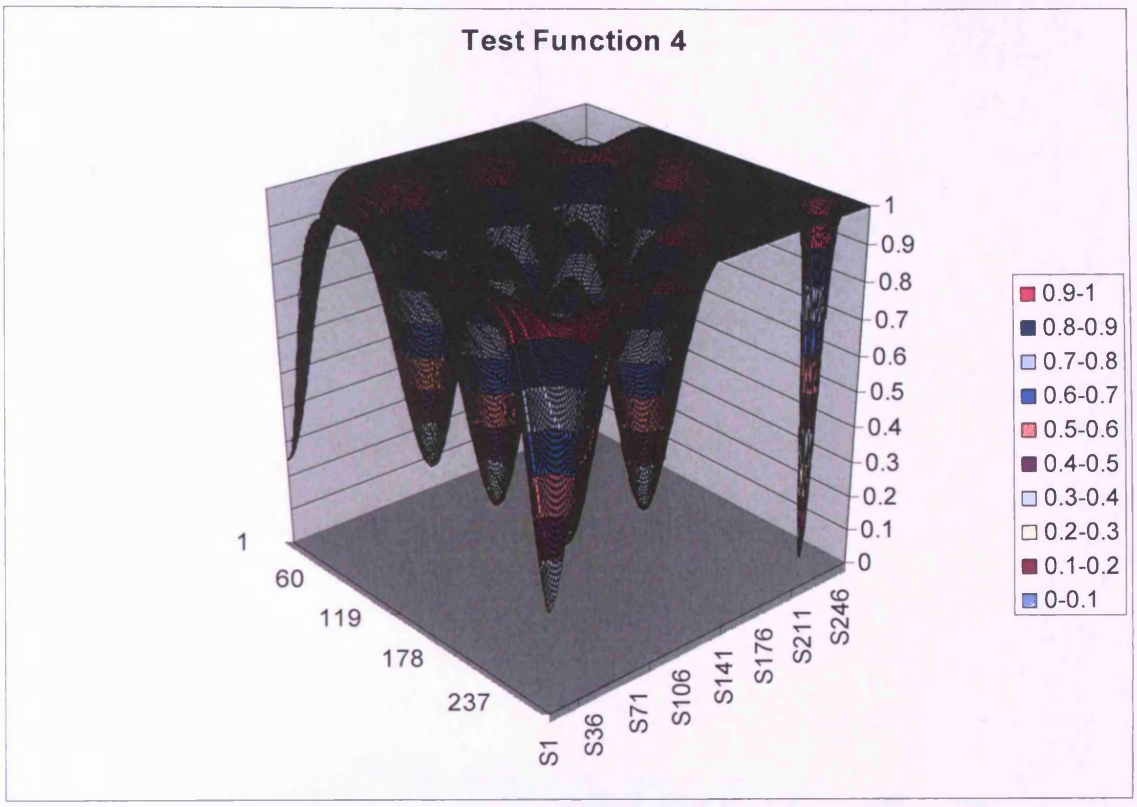


Figure E12a: Visualisation of MCastellani TF 4

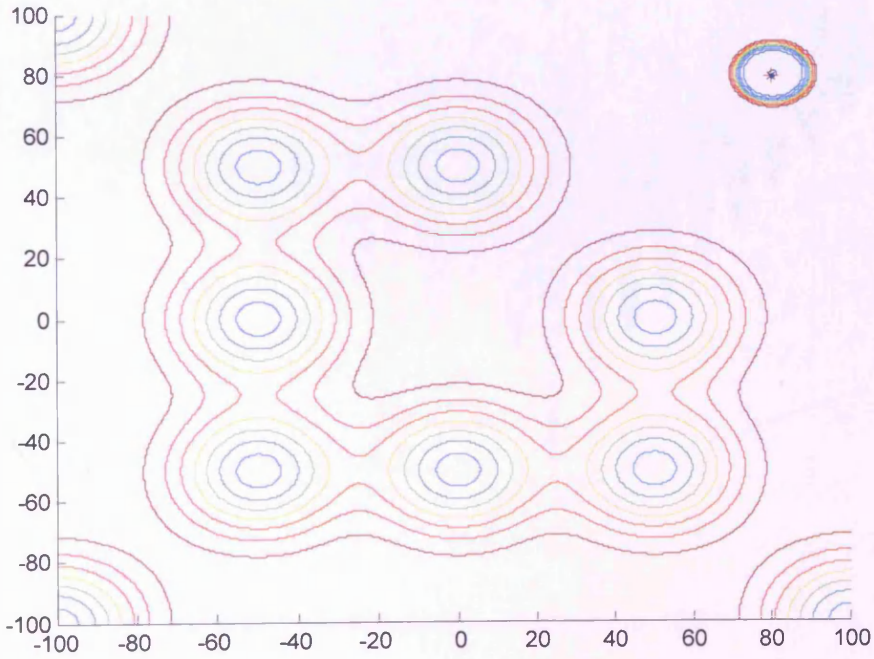


Figure E12b: Contour plot of MCastellani TF 4

13. MCastellani TF 5

$$f(x_1, x_2) = 1.0 +$$
$$-\left(0.4 + 0.4 \cdot \cos\left(2 \cdot \pi \cdot \frac{x_1 - 75}{25}\right)\right) \cdot \frac{x_1 + 100}{400}$$
$$-\left(0.4 + 0.4 \cdot \cos\left(2 \cdot \pi \cdot \frac{x_2 - 75}{25}\right)\right) \cdot \frac{x_2 + 100}{400}$$
$$-0.30 \cdot e^{-\left(\frac{(x_1 - 75)^2 + (x_2 - 75)^2}{10}\right)}$$

$$-100 \le x_1 \le 100$$
$$-100 \le x_2 \le 100$$
$$f(75,75) = 0$$

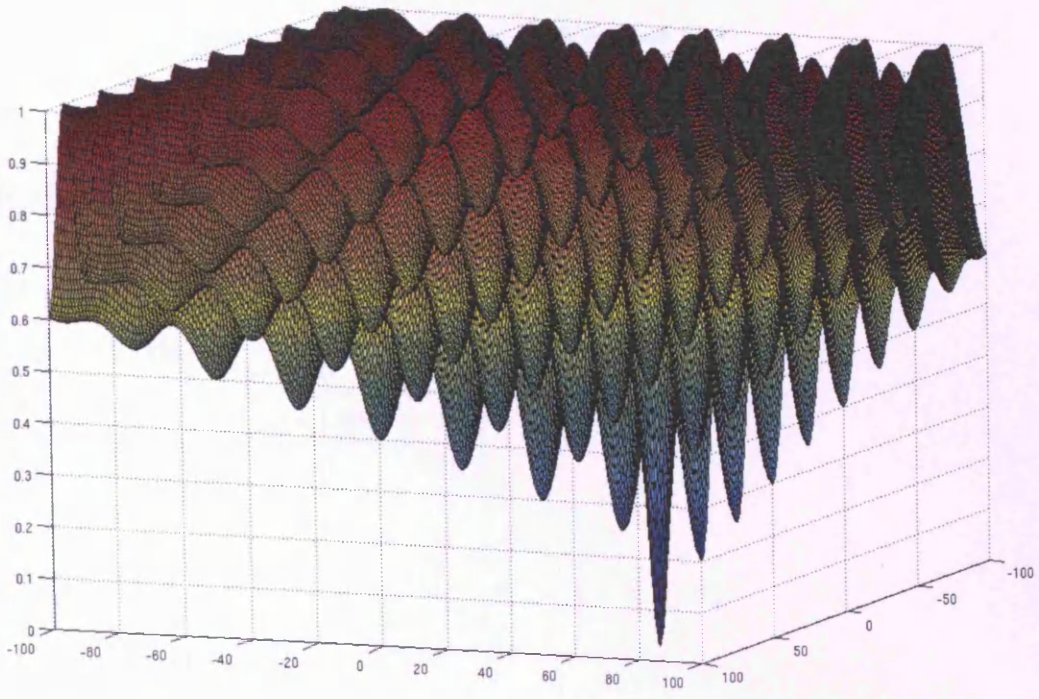


Figure E13a: Visualisation of MCastellani TF 5

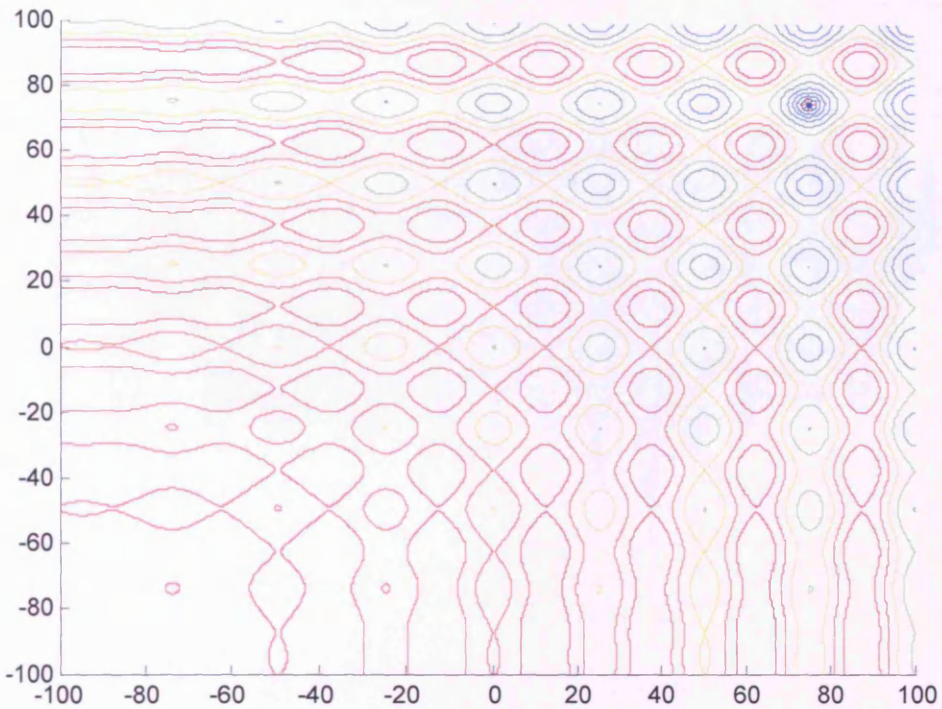


Figure E13b: Contour plot of MCastellani TF 5

14. MCastellani TF 6

$$f(x_1, x_2) = 1.0 +$$

$$-0.2 + 0.2 \cdot \sin\left(2 + \frac{x_2 + 100}{100} \cdot \pi \cdot \frac{x_1}{50}\right)$$

$$-0.2 + 0.2 \cdot \sin\left(2 + \frac{x_1 + 100}{100} \cdot \pi \cdot \frac{x_2}{50}\right)$$

$$-0.2 \cdot e^{-\left(\frac{(x_1 - 50)^2 + (x_2 - 50)^2}{10}\right)}$$

$$-100 \le x_1 \le 100$$

$$-100 \le x_2 \le 100$$

$$f(50,50) = 0$$

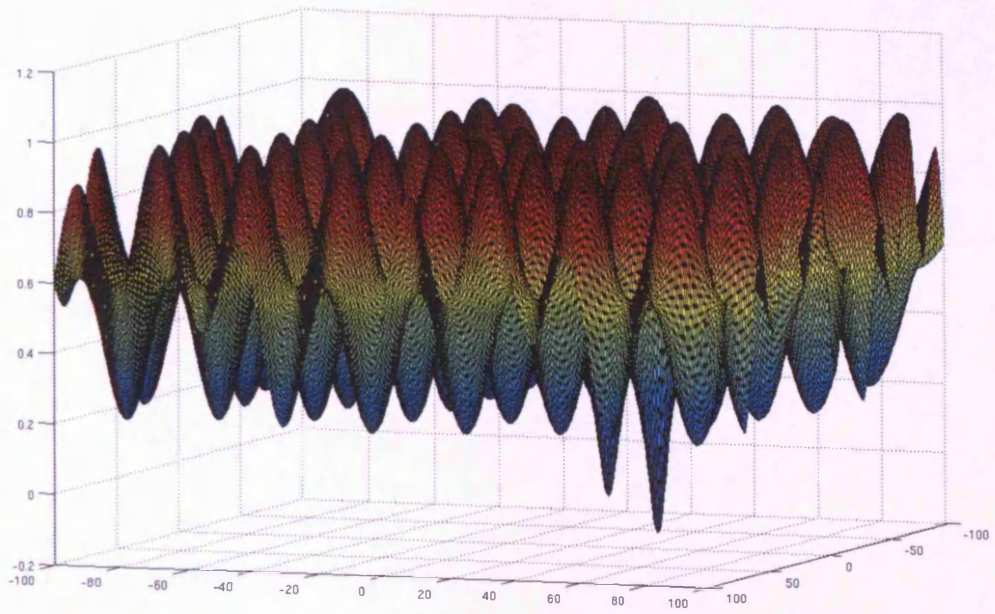


Figure E14a: Visualisation of MCastellani TF 6

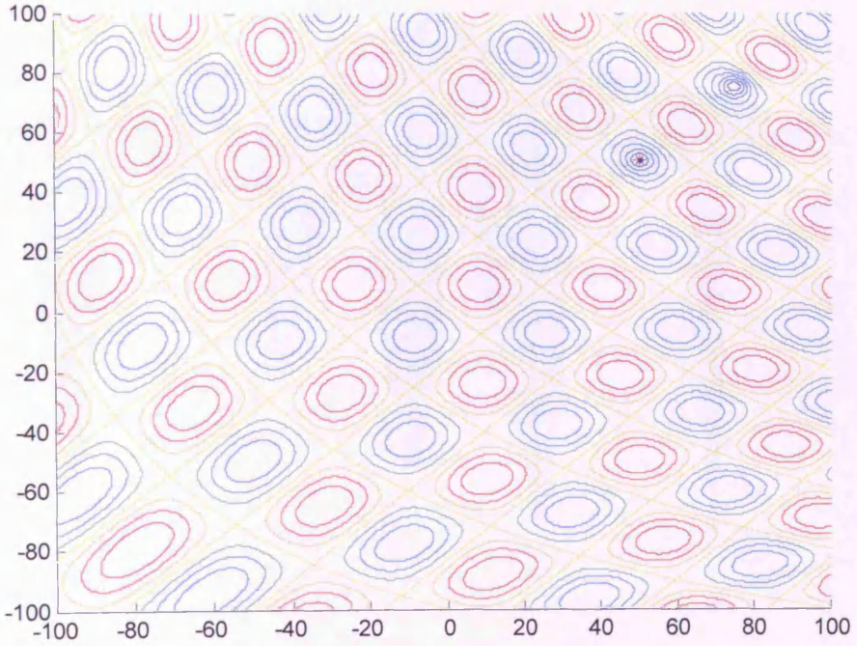


Figure E14b: Contour plot of MCastellani TF 6

15. MCastellani TF 7

$$f(x_1,x_2)=1.0+$$
$$-1.0\cdot e^{-\left(\frac{x_1-90}{25}\right)^2}\cdot e^{-\left(\frac{(x_1+2\cdot x_2)^2}{2}\right)}$$
$$-0.75\cdot e^{-\left(\frac{x_1+100}{25}\right)^2}\cdot e^{-\left(\frac{(x_1+2\cdot x_2)^2}{1000}\right)}$$
$$-0.80\cdot e^{-\left(\frac{(x_1+0)^2+(x_2+0)^2}{150}\right)}$$
$$-0.75\cdot\left(\frac{x_2}{100}\right)^2\cdot e^{-\left(\frac{(3\cdot x_1-x_2)^2}{1000}\right)}$$

$$-100\le x_1\le 100$$
$$-100\le x_2\le 100$$
$$f(0,0)=0$$

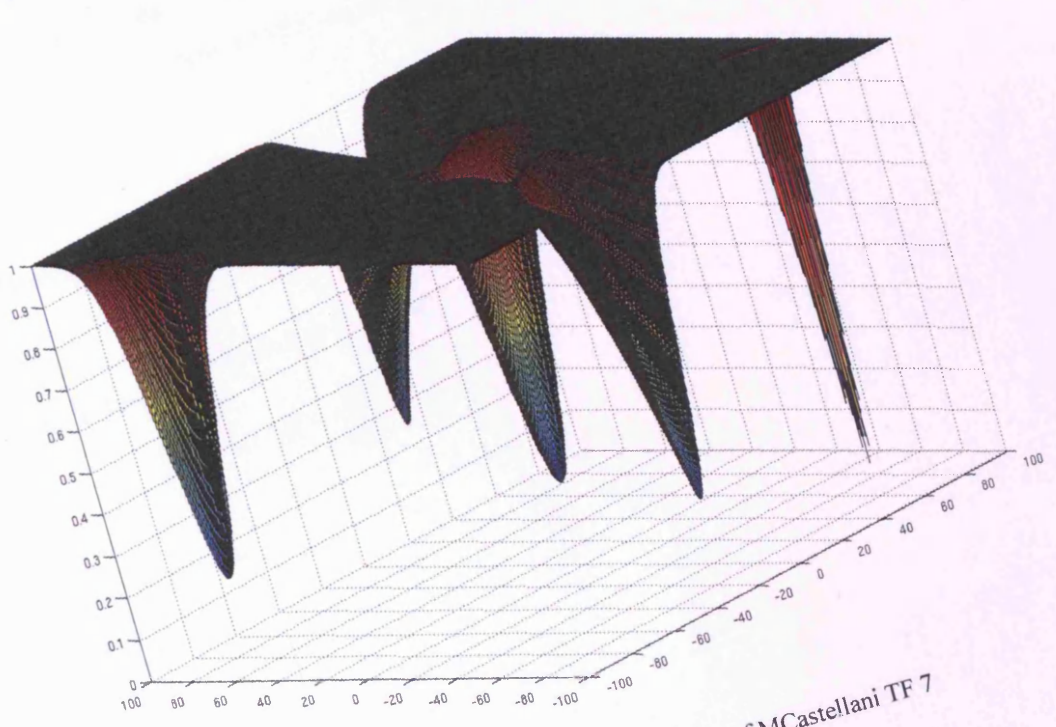


Figure E15a: Visualisation of MCastellani TF 7

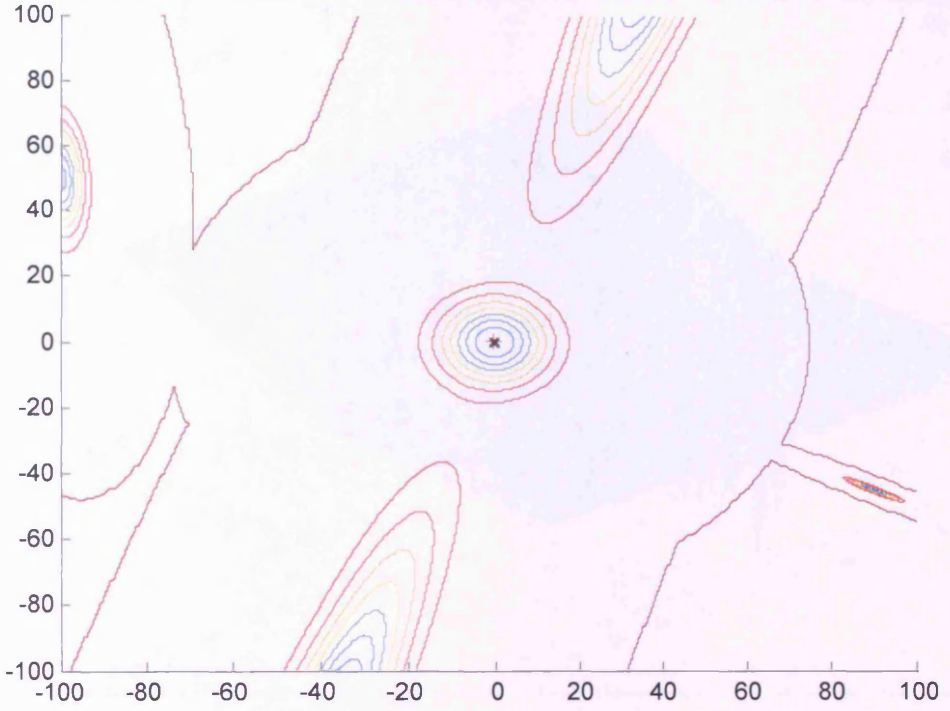


Figure E15b: Contour plot of MCastellani TF 7

16. MCastellani TF 8

$$f(x_1,x_2)=\begin{cases}(x_1\geq 0)\Rightarrow 1.0-\dfrac{x_1^4}{90^4}\cdot e^{-\left(\frac{\left(x_1^2+4\cdot x_2^2-8100\right)^2}{250}\right)}\\ else\Rightarrow 1.0-\dfrac{x_1}{125}\end{cases}$$

$$-100\leq x_1\leq 100$$
$$-100\leq x_2\leq 100$$
$$f(90,0)=0$$

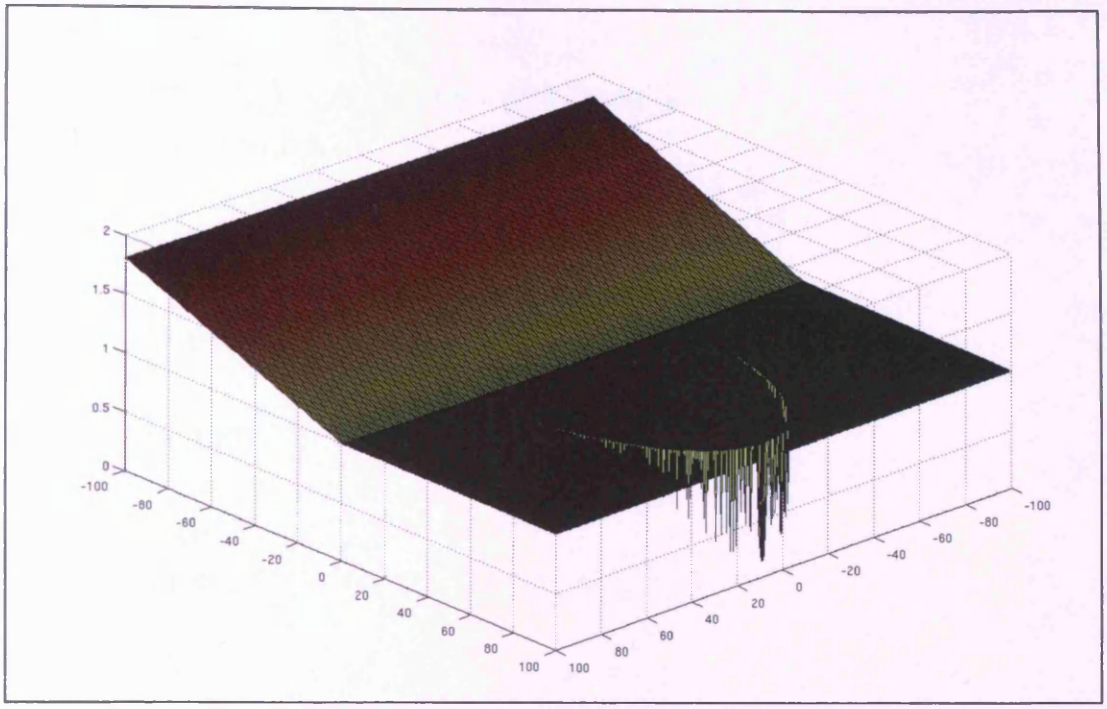


Figure E16a: Visualisation of MCastellani TF 8

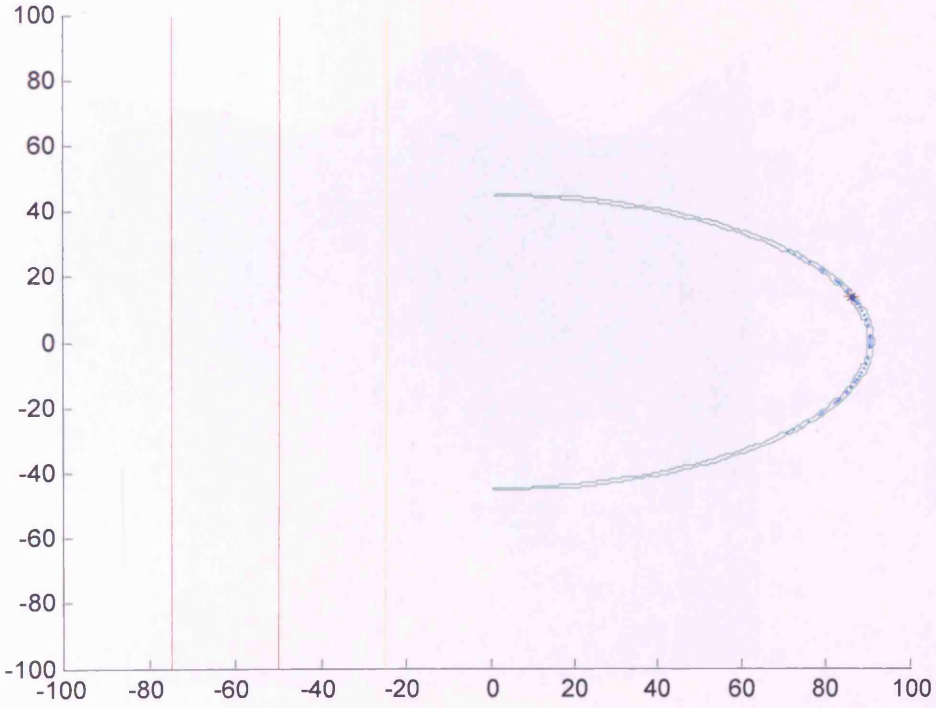


Figure E16b: Contour plot of MCastellani TF 8

17. MCastellani TF 9

$$f(x_1,x_2)=\begin{cases}\left((x_1+x_2)^2\le 625\right)\Rightarrow 1.0-0.25-0.75\cdot e^{-\left(\frac{(x_1+x_2)^2}{3}\right)}\\ else\Rightarrow 1.0+\\ -0.75\cdot e^{-\left(\frac{(x_1+75)^2+(x_2+75)^2}{1500}\right)}-0.75\cdot e^{-\left(\frac{(x_1-75)^2+(x_2-75)^2}{1500}\right)}\\ -0.75\cdot e^{-\left(\frac{(x_1-75)^2+(x_2+75)^2}{1500}\right)}-0.75\cdot e^{-\left(\frac{(x_1+75)^2+(x_2-75)^2}{1500}\right)}\\ -0.75\cdot e^{-\left(\frac{(x_1+0)^2+(x_2+0)^2}{850}\right)}\end{cases}$$

$$-100\le x_1\le 100$$
$$-100\le x_2\le 100$$
$$f(0,0)=0$$

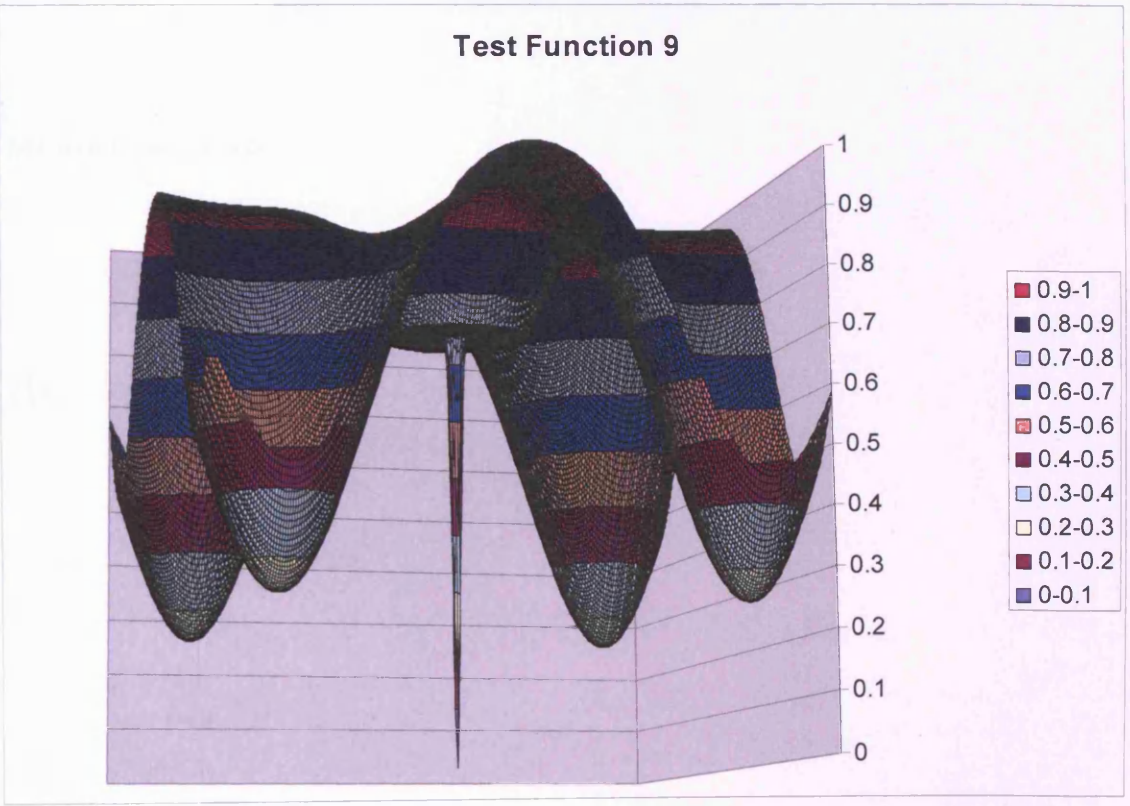


Figure E17a: Visualisation of MCastellani TF 9

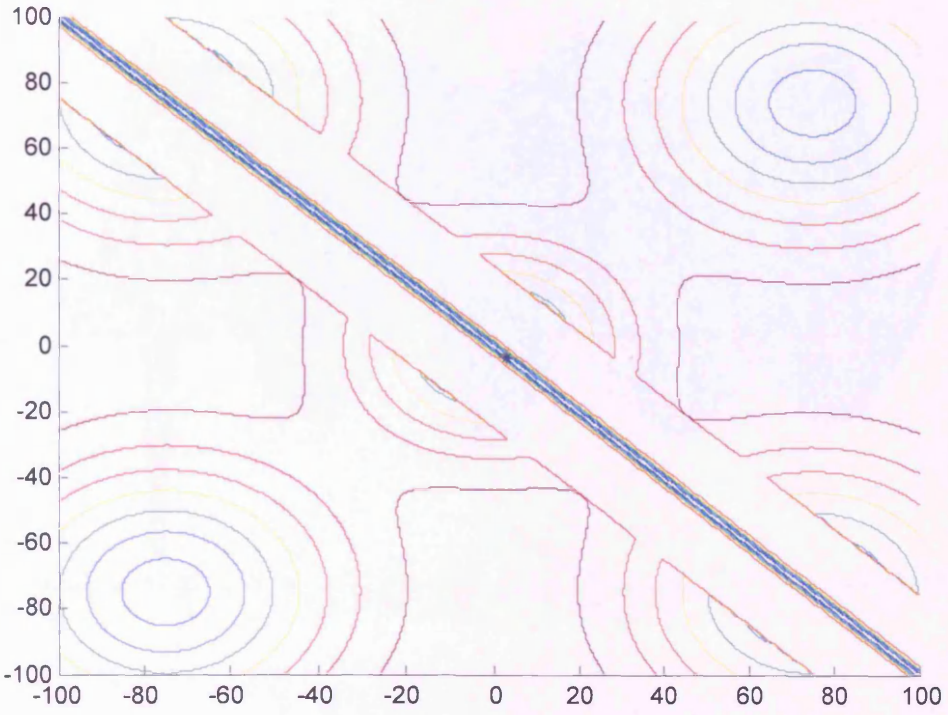


Figure E17b: Contour plot of MCastellani TF 9

18. MCastellani TF 10

$$f(x_1,x_2)=\begin{cases}\left((x_1+75)^2+(x_2+75)^2\le 625\right)\Rightarrow 1.0-0.6-0.4\cdot e^{-\left(\frac{(x_1+75)^2+(x_2+75)^2}{10}\right)}\\ else\Rightarrow 1.0+\\ -0.75\cdot e^{-\left(\frac{(x_1+75)^2+(x_2+75)^2}{2500}\right)}-0.75\cdot e^{-\left(\frac{(x_1-75)^2+(x_2-75)^2}{2500}\right)}\\ -0.75\cdot e^{-\left(\frac{(x_1-75)^2+(x_2+75)^2}{2500}\right)}-0.75\cdot e^{-\left(\frac{(x_1+75)^2+(x_2-75)^2}{2500}\right)}\\ -0.75\cdot e^{-\left(\frac{(x_1+0)^2+(x_2+0)^2}{1000}\right)}\end{cases}$$

$$-100\le x_1\le 100$$
$$-100\le x_2\le 100$$
$$f(-75,-75)=0$$

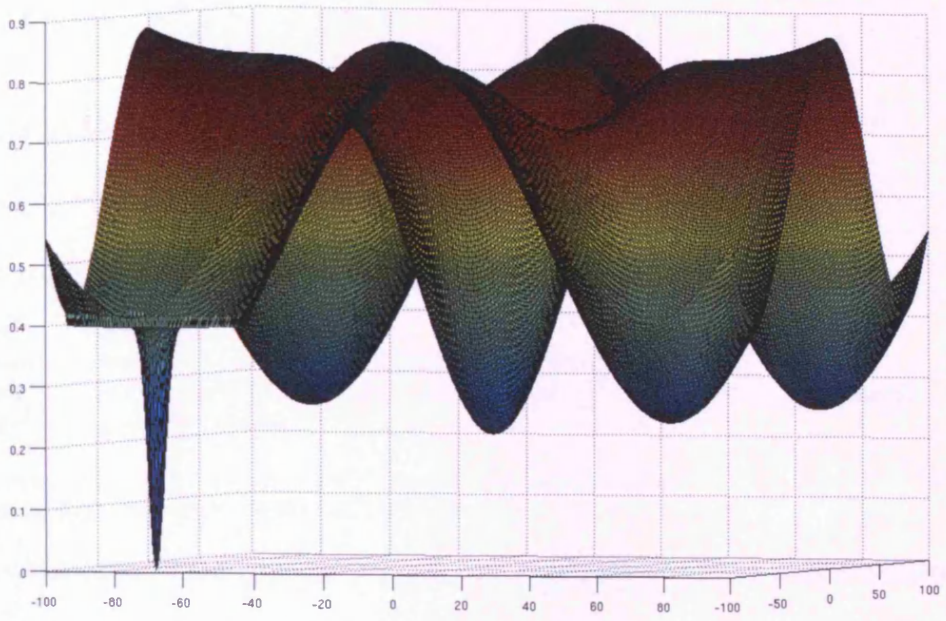


Figure E18a: Visualisation of MCastellani TF 10

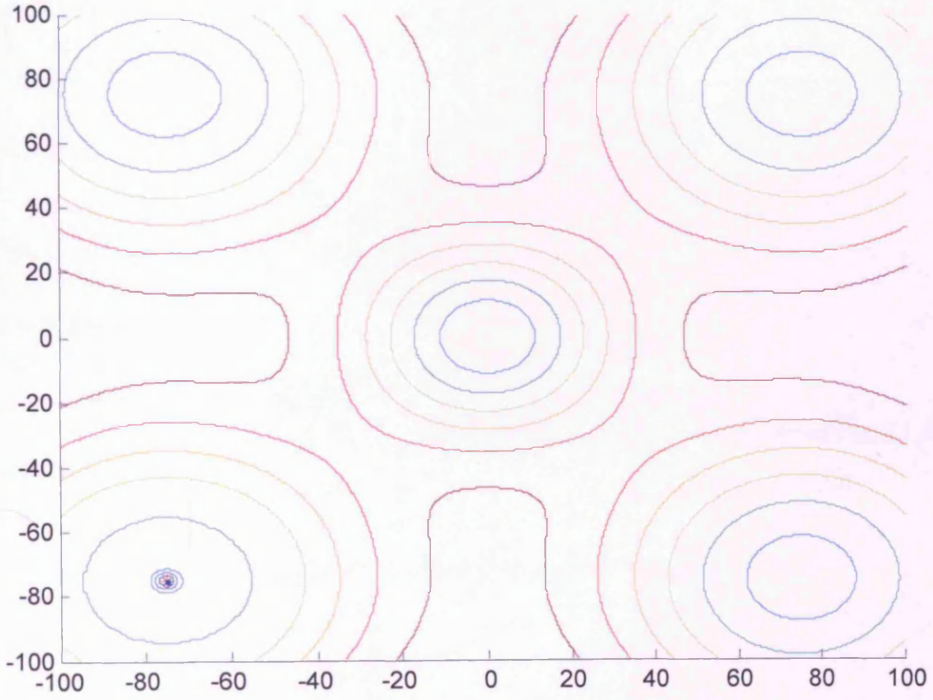


Figure E18b: Contour plot of MCastellani TF 10

Appendix F

glp set

```
%in interval (0,1)
%
% GlpSet        calculated glpset
% DimNum        dimension number, such as the number of parameters
% PntNum        point number you need, such as the number of experiment to be implemented
% GenVect       generating vector,
%
% Programmed by Y. Z. Liang 12/10/1995, Hong Kong Baptist University
% Revised by Feng Gan 18/09/2000, Hunan University
%
% Revised by Michael Sholedolu 10/08/2008, Manufacturing Engineering Centre, Cardiff University

function [GlpSet]=GlpSet (DimNum, PntNum, GenVect)
GlpSet=zeroes (DimNum, PntNum);
[m,n] = size (GenVect);
k = 0;
for i = 1:n
    if GenVect(1,i) = = PntNum
        k = i;
        break;
    else
        k = 0;
    end
end

if k~ = 0
    for i = 1:DimNum
        for j = 1:PntNum
            GlpSet (i,j) = (j*GenVect (i + 1, k) – 0.5) / GenVect
                           (1,k) – fix((j*GenVect (i + 1, k) - …0.5) / GenVect(1,k));
        end
    end
end
    msgbox("Wrong point number!", "Error message", "warn");
    GenVect = [ ];
end
% end of routine
```

Appendix G

Modifications to the PSO Algorithm

Some of the modifications to the PSO Algorithm since its development in 1995 are described in this appendix. These modifications resulted in variants of the algorithm that were proposed to incorporate the aptitude and capabilities of other evolutionary computation methods, such as hybrid versions of the PSO or the adaptation of the PSO parameters for a better performance (adaptive PSO). In other variations of the PSO Algorithm, the nature of the problem to be solved necessitate the PSO to work under complex environments as in the case of the multi-objective, constrained optimisation problems and tracking dynamic systems. Also included are other variants to the original formulation incorporated to improve the performance of the algorithm, such as the stretching and passive congregation techniques to prevent the particles from being trapped in local minima. For convenience, a detailed analysis of these modifications from Valle (Valle *et al.* 2008) are reproduced below:

G1 Hybrid PSO Algorithm

The growth and improvement of the Particle Swarm Optimisation Algorithm was arrived at by integrating routines and procedures that have already been tested in other evolutionary computation techniques. These include incorporating selection, mutation and crossover as well as differential evolution (DE) into the PSO Algorithm. The intension was to increase the diversity of the population through:

- preventing the particles from moving too close to each other and collide (Blackwell and Bentley 2002; Krink *et al.* 2002)
- self-adapting parameters such as the constriction factor, acceleration constants (Miranda and Fonseca 2002c), or inertia weight (Lovbjerg and Krink 2002).

Consequently, the hybrid versions of the PSO Algorithm came into existence and were tested in different combinations such as the hybrid of the Genetic Algorithm and the PSO (GA-PSO), evolutionary PSO (EPSO) and differential evolution PSO (DEPSO).

G2 Hybrid of Genetic Algorithm and PSO (GA-PSO)

The GA-PSO variant combines the advantages of swarm intelligence and a natural selection mechanism, such as the GA, thereby increasing the number of highly evaluated agents, while at the same time, also decreasing the number of lowly evaluated agents at each iteration step. Thus, not only is it possible to successively change the current searching area by considering the P_{best} and G_{best} values, but also to jump from one area to another by the selection mechanism, which results in accelerating the convergence speed of the whole algorithm. A major aspect of the classical GA approach is employed by the PSO Algorithm, which is the potential of "breeding". Furthermore, other authors have also analysed the inclusion of mutation or a simple replacement of the best fitted value, as a means of improvement to the standard PSO formulation (El-Dib *et al.* 2004), (Naka *et al.* 2003). El-Dib (El-Dib *et al.* 2004) considered the application of a reproduction system that modifies both the position and velocity vectors of randomly selected particles in order to further improve the potential of the PSO to reach an optimum.

$$
\begin{aligned}
child_1(x) &= p \,.\, parent_1(x) + (1-p) \,.\, parent_2(x) \\
child_1(v) &= (parent_1(v) + parent_2(v)) \\
&\quad \cdot \frac{|parent_1(v)|}{|parent_1(v) + parent_2(v)|} \\
child_2(x) &= p \,.\, parent_2(x) + (1-p) \,.\, parent_1(x) \\
child_2(v) &= (parent_1(v) + parent_2(v)) \\
&\quad \cdot \frac{|parent_2(v)|}{|parent_1(v) + parent_2(v)|}
\end{aligned} \tag{G.1}
$$

where p~U[0,1], $parents_{1,2}(x)$ represent the position vectors of randomly chosen particles, $parents_{1,2}(v)$ are the corresponding velocity vectors of each parent and $child_{1,2}(x)$, $child_{1,2}(v)$ are the offspring of the breeding process. (Naka *et al.* 2003) suggested replacing agent positions having low fitness values with those having high fitness, according to a selection rate S_T, keeping the P_{best} information of the replaced agent so that a dependence on the past high evaluation position is accomplished.

G3 Hybrid of Evolutionary Programming and PSO (EPSO)

The Evolutionary PSO Algorithm integrates a selection process into the original PSO Algorithm as well as a self-adapting methodology for its parameters. Angeline (Angeline 1998) proposed adding the tournament selection method as employed in evolutionary programming (EP). In this approach, the update formulas remain the same as in the original PSO Algorithm but the particles are selected as follows:

- The fitness value of each particle is compared with other particles and scores a point for each particle with a worst fitness value. The population is sorted based on this score.
- The current positions and velocities of the best half of the swarm replace the positions and velocities of the worst half.
- The individual best of each particle of the swarm (best and worst half) remain unmodified. At each iteration, half of the individuals are moved to positions of the search space that are closer to the optimal solution than their previous positions while keeping their personal best points.

The difference between this technique and the original particle swarm is that the exploitative search procedure is accentuated. This makes it possible for the optimum to be found more regularly than the original particle swarm. Miranda and Fonseca (Miranda and Fonseca 2002a, b, c) introduced self-adaptation capabilities to the swarm in addition to the selection mechanism by modifying the concept of a particle to include, not only the objective parameters but also a set of strategic parameters (inertia and acceleration constants, simply called weights).

The general EPSO scheme is summarised as follows (Miranda and Fonseca 2002a, b, c):

- Replication: Each particle is replicated τ times.
- Mutation: Each particle has its weights mutated.
- Reproduction: Each mutated particle generates an offspring according to the particle movement rule.
- Evaluation: Each offspring has a fitness value.

- Selection: Stochastic tournament is carried out in order to select the best particle which survives to the next generation.

The particle movement is defined as:

$$v_i(t) = w^*_{i1} \cdot v_i(t-1) + w^*_{i2} \cdot \text{rand}_1 \cdot (p_i - x_i(t-1)) + w^*_{i3} \cdot \text{rand}_2 \cdot (p^*_g - x_i(t-1)) \quad \text{(G.2)}$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad \text{(G.3)}$$

where

$$w^*_{ik} = w_{ik} + \tau' \cdot rand \quad \text{(G.4)}$$

and *rand* is a random number with normal distribution, i.e., N(0,1).

The global best is also mutated by

$$p^*_g = p_g + \tau' \cdot rand \quad \text{(G.5)}$$

where τ and τ' are learning parameters that can be either fixed or dynamically changing as strategic parameters.

G4 Hybrid of Differential Evolution and PSO (DEPSO) and Composite PSO (C-PSO)

A differential evolution operator was proposed to improve the performance of the PSO Algorithm in two different ways:

(1) applied to the particles' best position to eliminate the particles falling into local minima (DEPSO) (Zhang and Xie 2003), (Talbi and Batouche 2004), (Moore and Venayagamoorthy 2006).

(2) to find the optimal parameters (inertia and acceleration constants) for the classical PSO or Composite PSO (C-PSO) (Kannan *et al.* 2004).

G4.1 Differential Evolution PSO (DEPSO)

The DEPSO method proposed by Zhang and Xie (Zhang and Xie 2003) interchange the original PSO algorithm and the DE operator, i.e., (1) and (2) above are performed at the odd iterations and equation (G.6) at the even iterations. The DE mutation operator is defined over the particles' best positions p_i with a trial point $T_i = p_i$ which for the dth dimension is derived as

$$\text{If (rand} < \text{CR or } d = k) \text{ then } T_{id} = p_{gd} + \delta_{2d} \tag{G.6}$$

where k is a random integer value within [1, n] which ensures the mutation in at least one dimension, CR is a crossover constant (CR$\leq$ 1) and δ_2 is the case of N = 2 for the general difference vector

$$\delta_N = \frac{1}{N}\sum_{1}^{N}\Delta \tag{G.7}$$

where Δ is the difference between two elements randomly chosen in the p_{best} set.

If the fitness value of T_i is better than the one for p_i, then T_i replaces p_i. After the DE operator has been applied to all the particles' individual best values, the G_{best} value is

chosen from among the P_{best} set, providing the social learning capability, which might speed up the convergence.

G4.2 Composite PSO (C-PSO)

The selection of the PSO parameters (φ_{ic}, φ_1, φ_2) is made through trial and error in the previously presented algorithms. Employing other optimisation algorithms such as the GA, the EP, or the DE, some of the techniques they used help to make the selection procedure more efficient. The Composite PSO algorithm is a method that implements the DE to solve the problem of parameter selection. The resulting algorithm (Kannan *et al.* 2004) is summarised next:

Step 1) Initialise t to 1 and set the maximum number of iterations as T. Generate initial position of particles $x(0)$, initial velocity $v(0)$, and the initial PSO parameters $X = (\varphi_{ic}, \varphi_1, \varphi_2)$ randomly. The size of x, v and X is equal to N_p, the size of the population, and t is the current iteration number.

Step 2) For each X, calculate $v(t)$ and $x(t)$ as

$$v_i(t) = \varphi_{ic} \cdot v_i(t-1) + \varphi_1 \cdot \text{rand}_1 \cdot (p_i - x_i(t-1)) + \varphi_2 \cdot \text{rand}_2 \cdot (p_g - x_i(t-1)) \qquad \text{(G.8)}$$

and with equation (G.3).

Calculate the fitness function value for each particle.

Step 3) Apply mutation, crossover, and selection operators of the DE algorithm to X. Let X^* be the best individual produced by this process. Replace X by X^* and repeat the procedure until a terminal number of iterations of DE (selected *a priori*) is reached.

Step 4) The process continues from Step 2) until the stopping criterion (maximum number of iterations T) is met.

G5 Adaptive PSO Algorithm

In this variant, some researchers have suggested various additional adjustments to the parameters of the PSO algorithm:

- adding a random component to the inertia weight (Eberhart and Shi 2001a; Mohagheghi *et al.* 2005; Valle *et al.* 2005).
- applying Fuzzy logic (Shi and Eberhart 2001a, b).
- using a secondary PSO to find the optimal parameters of a primary PSO (Doctor *et al.* 2004).
- introducing Q-learning (Khajenejad *et al.* 2006).
- using adaptive critics (Venayagamoorthy 2004), (Doctor and Venayagamoorthy 2005).

(Zhang *et al.* 2003) have also considered the adjustment of the number of particles and the neighbourhood size. The PSO algorithm is modified by adding an improvement index for the particles of the swarm.

$$\delta(x) = \frac{f(x_i(t_0)) - f(x_i(t))}{f(x_i(t_0))} \qquad \text{(G.9)}$$

where $f(x_i(t))$ is the fitness function value for particle i at iteration t.

An improvement threshold was defined as the limit for the minimum acceptable improvement. The adaptive strategies include (Zhang *et al.* 2003):

- **Adjusting the swarm size:** If the particle has enough improvement but it is the worst particle in its neighbourhood, then remove the particle. On the other hand, if the particle does not have enough improvement but it is the best particle in its neighbourhood, then generate a new particle.
- **Adjusting the inertia weight:** The more a particle improves itself, the smaller the area this particle needs to explore. In contrast, if the particle has a deficient improvement then it is desirable to increase its search space. This is achieved from the adjustment of the inertia weight.
- **Adjusting the neighbourhood size:** If the particle is the best in its neighbourhood but it has not improved itself enough, then the particle needs more information and the size of the neighbourhood has to be increased. If the particle has improved itself satisfactorily, then it does not need to ask many neighbours and its neighbourhood size can be reduced.

In similar vein, Li (Li 2004) proposed a species-based PSO (SPSO). In this method, the swarm population is divided into species of subpopulations based on their similarity. Each species is grouped around a dominating particle called the *species seed.* At each iteration step, the species seeds are identified and adopted as neighbourhood bests for the species groups. Over successive iterations, the adaptation of the species allows the algorithm to find multiple local optima, from which the global optimum can be identified.

G6 PSO in Complex Environments

This section identifies PSO in complex environments. These include amongst others Multi-objective Particle Swarm Optimisation, Dynamic Neighbourhood PSO and Vector Evaluated PSO.

G6.1 Multi-objective Particle Swarm Optimisation (MOPSO)

Multi-objective optimisation encompasses several objectives that must be achieved simultaneously. A methodology in solving this problem is to aggregate the multiple objectives into one objective function taking into consideration the weights that are fixed or those that change dynamically during the optimisation process (Parsopoulos and Vrahatis 2002a). The shortcoming of this approach is the inability to consistently find the appropriate weighted function. On other occasions, there is the need to take into account the tradeoffs between the multiple objectives - finding the multiple Pareto optimal solutions (Parsopoulos and Vrahatis 2002b).

The main concern to be addressed in the selection of cognitive and social leaders (P_{best} and l_{best}) in MOPSO algorithms is the provision of an effective guidance to reach the most promising Pareto front region, while at the same time maintaining the population diversity.

Two approaches are proposed in the literature:

- selection based on quantitative standards and
- random selection.

In the first approach, the leader is determined by a process excluding any randomness involved, such as the Pareto ranking scheme proposed by (Ray 2002), the sigma method

by (Mostaghim and Teich 2003) or the dominated tree (Fieldsend *et al.* 2003). For the random approach, the selection of a candidate is stochastic and proportional to certain weights alloted to maintain the population diversity - crowding radius, crowding factor, niche count (Hu 2006). Ray and Liew (Ray and Liew 2002) choose the particles that perform better to be the leaders while the other particles have a propensity to move towards a randomly selected leader from the leader group in which the leader having the smallest number of followers has the highest probability of being selected.

Pareto dominance was integrated into the PSO algorithm by Coello and Lechuga (Coello and Lechuga 2002) in which the non-dominated solutions are stored in a secondary population and the primary population utilise a randomly selected neighbourhood best from the secondary population to update their velocities. The authors proposed an adaptive grid to generate well distributed Pareto fronts and mutation operators to enhance the exploratory capabilities of the swarm (Coello *et al.* 2004).

Li (Li 2003) proposed the idea of sorting the entire population into various non-domination levels such that the individuals from better fronts can be selected. In this way, the selection process pushes towards the true Pareto front. This was made possible by preserving the two objectives (obtaining a set of non-dominated solutions as close as possible to the Pareto front and maintaining a well distributed solution set along the Pareto front).

In further works, (Salazar-Lechuga and Rowe 2005) developed different approaches such as combining classical PSO with auto fitness sharing concepts, dynamic neighbourhood PSO or vector evaluated PSO. These are explained in the next two subsections.

G6.2 Dynamic Neighbourhood PSO (DN-PSO)

(Hu and Eberhart 2002b), (Hu *et al.* 2002) developed the dynamic neighbourhood process for solving multi-objective optimisation problems. In this approach, the PSO algorithm was modified to locate the Pareto front.

- The multiple objectives are divided into two groups: F_1 and F_2. F_1 is defined as the neighbourhood objective, while F_2 is defined as the optimisation objective. The selection of F_1 and F_2 is random.
- At each iteration step, each particle defines its neighbourhood by calculating the distance to all other particles and choosing the M closest neighbours. In this case, the distance is described as the difference between the fitness values for the first group of objective functions.
- Once the neighbourhood has been determined, the best local value is found among the neighbours in terms of the fitness value of the second group of objective functions.
- The global best updating strategy considers only the solutions that dominate the current P_{best} value.

Hu (Hu *et al.* 2002) pioneered an extended memory for storing all the Pareto optimal solutions in a current generation to reduce computational time efficiently, improving the algorithm. An archive of fixed size was proposed by (Bartz-Beielstein *et al.* 2003)

whereby the decision for selection or deletion was taken according to the influence of each particle on the diversity of the Pareto front.

G6.3 Vector Evaluated PSO (VEPSO)

The Vector Evaluated Particle Swarm Optimisation (VEPSO) algorithm was proposed by Parsopoulos and Vrahatis (Parsopoulos and Vrahatis 2002b), based on the perception of the Vector Evaluated Genetic Algorithm (VEGA). In the VEPSO algorithm, two or more swarms are used to search the problem hyperspace. Each swarm is evaluated according to one of the objective functions and the information is exchanged between them. As a result, the knowledge coming from other swarms is used to guide each particle's trajectory towards the Pareto optimal points. The velocity update equation for an *M*-objective function problem as formulated by (Parsopoulos *et al.* 2004) is given below:

$$v_i^{[j]}(t) = \chi^{[j]} \{ \varphi_{ic}^{[j]} \cdot v_i^{[j]}(t-1) \ldots + \varphi_1 \cdot \text{rand}_1 \cdot \left(p_i^{[j]} - x_i(t-1) \right) \ldots + \varphi 2 \cdot \text{rand}_2 \cdot \left(p_g^{[s]} - x_i(t-1) \right) \} \quad \text{(G.10)}$$

where

Index j defines the swarm number ($j = 1, 2 \ldots M$)

Index i corresponds to the particle number ($i = 1, 2 \ldots N$)

$\chi^{[j]}$ is the constriction factor of swarm j

$\varphi^{[j]}$ is the inertia weight of swarm j

$p_i^{[j]}$ is the best position found by particle in swarm j

$p_g^{[s]}$ is the best position found for any particle in swarm s

G7 Constraint Handling in PSO

Most real life problems are subject to different constraints that limit the search space to a certain feasible region. In the literature, two methodologies are proposed to handle constraints applied to the PSO Algorithm:

- including the constraints in the fitness function using penalty functions
- dealing with the constraints and fitness separately

In the second approach, there are no extra parameters introduced in the PSO algorithm and there is no limit to the number or format of the constraints (Hu 2006) while the PSO basic equations for velocity and position updates remain unchanged. On determining the new positions for all the particles, the individual solution is checked to establish if it belongs to the feasible space or not. If the feasibility conditions are not met, one of the following actions can be taken:

- the particle is reset to the previous position, or the particle is reset to its p_{best}
- the non-feasible solution is kept, but the p_{best} is not updated (feasible solutions are stored in the memory) (Hu 2006), or the particle is re-randomised (Valle *et al.* 2006). During the initialisation process, all particles can be reinitialised until feasible solutions are found (Hu 2006).

Hu (Hu 2006) concluded in his work on benchmark functions, that the PSO Algorithm is efficient in handling constrained optimisation problems by finding better solutions in less time. Moreover, the PSO Algorithm does not require domain knowledge or complex

techniques and no additional parameters need to be tuned. The limitations of the method appear in problems with extremely small feasible spaces where other constraint handling techniques may need to be developed.

G8 Dynamic Tracking in PSO

The classical PSO Algorithm is attested to be effective, efficient and robust computation wise handling static optimisation problems. However, it is not as efficient in applications to dynamic systems with a constantly changing optimal value. (Hu *et al.* 2004) and (Eberhart and Shi 2001) proposed an adaptive methodology to the original PSO Algorithm to balance this problem. The idea of adaptation was incorporated by either re-randomising particles or dynamically changing the parameters of the PSO.

Two techniques was proposed to detect environmental changes: the "changed-gbest-value" and the "fixed-gbest-value" by Hu and Eberhart (Hu and Eberhart 2002a). The earlier technique suggests re-evaluating the fitness function for g_{best} at each iteration step. If g_{best} refers to the same particle but its corresponding fitness function value is different, then it is assumed that the dynamics of the system has changed. In view of the fact that this assumption may not necessarily be true for all dynamic systems, the second technique was proposed in which the locations of g_{best} and the second best particle are monitored. If there are no changes in both in a certain number of iterations, the algorithm assumes that the optimum has been found. An assortment of strategies was employed in both techniques to handle environmental changes by adapting the swarm – this amongst many other include re-randomising a certain number of particles (say 10%, 50%, or 100% of the population), resetting certain particles, re-randomising the g_{best} or a

combination of the previous strategies (Hu and Eberhart 2002a), (Carlisle and Dozier 2000).

A modification to the standard PSO termed small population PSO (SPPSO) was proposed by Das and Venayagamoorthy (Das and Venayagamoorthy 2006; Das *et al.* 2006). Here, the algorithm uses a small population of particles (five or less) that are regenerated every N iterations; all the particles are then replaced except by the g_{best} particle in the swarm while the population p_{best} attributes are passed on to the new generation purposely to keep the memory characteristics of the algorithm. As a consequence, the performance of the PSO improved on problems having dynamic conditions.

G9 Discrete PSO Variants

(Mohan and Al-Kazemi 2001) made modifications to the Binary version of the PSO Algorithm purposely (see modified velocity update equation below) to improve the efficacy and performance of the algorithm in different applications

$$\vec{v}_i(t) = \vec{v}_i(t-1) + \varphi_1 \cdot rand_1 \cdot (\vec{p}_i - \vec{x}_i(t-1)) \ldots + \varphi_2 \cdot rand_2 \cdot (\vec{p}_g - \vec{x}_i(t-1)) \quad \text{(G.11)}$$

(Mohan and Al-Kazemi 2001) proposed three variations:

- Direct methodology, in which the classical PSO Algorithm is applied and the solutions are converted into bit strings using a hard decision decoding process.
- Bias vector methodology, in which the velocity's update is randomly selected from the three parts in the right-hand side of (G.11), using probabilities depending on the value of the fitness function.

- Mixed search methodology, where the particles are divided into multiple groups and each of them can dynamically adopt a local or a global version of the PSO Algorithm.

(Mohan and Al-Kazemi 2001) also proposed coalescing the PSO Algorithm with other evolutionary algorithms and with the quantum theory. In the second scenario, the use of a quantum bit (Q-bit) was put forward to probabilistically denote a linear superposition of states (binary solutions) in the search space (Shi 2004), (Yang *et al.* 2004), (Moore and Venayagamoorthy 2005). Obtained results confirmed that the proposed method was faster and more efficient in contrast to the classical binary PSO and other evolutionary algorithms.

(Cedeño and Agrafiotis 2005) had a different approach, in which the original Particle Swarm Algorithm was adapted to the discrete problem of feature selection by normalising the value of each component of the particles' position vector at each run. By so doing it was possible to view the location of the particles as the probabilities that were used in a roulette wheel to ascertain if the entry x_{ij} takes 1 or 0, invariably determining whether the jth feature in the ith particle was chosen or not for the next generation.

G10 Other Variants of the PSO Algorithm

This section includes other variants of the PSO Algorithm not categorised in the previous sections above. They include the Gaussian PSO, the Dissipative PSO, the PSO with passive congregation, the Stretching PSO, the Cooperative PSO and the Comprehensive Learning PSO.

G10.1 Gaussian PSO (GPSO)

In the classical PSO Algorithm, the search is performed in the median between the global and local best. How the search is performed plus the convergence of the swarm in the optimal area depends on the adjustment of the parameters such as the acceleration coefficient and the inertia weight. While attempting to resolve this weaknesses, some authors (Secrest and Lamont 2003), (Krohling 2004, 2005), introduced Gaussian functions for guiding the movements of the particles. Here, the inertia constant is no longer needed while the acceleration constant is replaced by random numbers with Gaussian distributions (Krohling 2004, 2005).

Secrest and Lamont (Secrest and Lamont 2003) proposed the following update formula:

$$
\begin{aligned}
|v(t)| &= \text{Grand}((1 - C_1) \\
&\quad . |p_i(t-1) - p_g(t-1)| \quad \text{when rand} > C_1 \\
|v(t)| &= \text{Grand}(C_2) \\
&\quad . |p_i(t-1) - p_g(t-1)| \quad \text{when rand} \leq C_1 \\
v(t) &= |v(t)| \,.\, \text{Rand}(\theta)
\end{aligned}
\tag{G.12}
$$

where

$|p_i(t-1) - p_g(t-1)|$ distance between the global and local best. If both points are the same, then it is set to one

C_1 a constant between zero and one that determines the "trust" between the global and local best. The larger the C_1 is the more particles will be placed around the global best

C_2	a constant between zero and one that establishes the point between the global ($p_g(t)$) and the local best ($p_i(t)$) that is a standard deviation from both
Grand(y)	a zero-mean Gaussian random number with standard deviation of y
rand	a random number between zero to one with uniform distribution
Rand(θ)	a random vector with magnitude of one, and its angle is uniformly distributed from zero to 2π

Applying this modification to the PSO algorithm, the neighbourhood around the global and local best is searched primarily. As the global and local best get closer together, the standard deviation decreases and the area being searched converges.

Again, Krohling (Krohling 2004, 2005) proposed a different method for updating the velocity at each iteration step:

$$v_i(t) = \text{rand}_1 \cdot (p_i - x_i(t-1) + \text{rand}_2 \cdot (p_g - x_i(t-1)) \quad \text{(G.13)}$$

Where rand_1 and rand_2 are positive random numbers generated according to the absolute value of the Gaussian probability distribution, *abs*[N(0,1)].

Taking into account the previous modifications in the velocity update formula, the coefficients of the two (p - x) terms are generated automatically by using the Gaussian probability distribution. As a result, there is no need to specify any other parameters.

Furthermore, the author claims that by using the Gaussian PSO, the maximum velocity V_{max} is no longer needed.

G10.2 Dissipative PSO (DPSO)

The DPSO introduced negative entropy purposely to stimulate the model in PSO, thus creating a dissipative structure that prevents premature stagnation (Biskas *et al.* 2005; Xie *et al.* 2002). The negative entropy mainly introduced additional chaos in the velocity of the particles as follows:

$$\text{If (rand} < c_v\text{) then } v_{id} = \text{rand} \cdot V_{max,d} \tag{G.14}$$

where *rand* and c_v are both random numbers between 0 and 1.

Similarly, the chaos for the location of the particles is represented as follows:

$$\text{If (rand} < c_l\text{) then } x_{id} = \text{Rand}(_{ld,ud}) \tag{G.15}$$

where *rand* is a random number between 0 and 1 and Rand(l_d, u_d) is another random number with predefined lower and upper limits (Biskas *et al.* 2005). The chaos basically introduced the negative entropy keeping the system out of the equilibrium state. The self-organisation of the dissipative structures, along with the inherent non-linear interactions in the swarm resulted in sustainable development from fluctuations (Xie *et al.* 2002).

G10.3 PSO with Passive Congregation (PSOPC)

(He *et al.* 2004) proposed passive congregation, a methodology that allows animals to aggregate into groups as a possible alternative to preclude the PSO Algorithm from being

trapped in local optima but also to improve its accuracy and convergence speed. The velocity update formula with passive congregation is given below:

$$v_i(t) \quad \varphi_{ic} \cdot v_i(t-1) + \varphi_1 \cdot r_1 \cdot (p_i - x_i(t-1)) \ldots$$
$$+ \varphi_2 \cdot r_2 \cdot (p_g - x_i(t-1)) + \varphi_3 \cdot r_3 \cdot (X - x_i(t-1)) \qquad \text{(G.16)}$$

where r_1, r_2 and r_3 are random numbers between 0 and 1, φ_3 is the passive congregation coefficient, and X is a particle randomly selected from the swarm.

(He *et al.* 2004) excluded the range of the values for the congregation coefficient as well as its effect on the efficiency and performance of the algorithm.

G10.4 Stretching PSO (SPSO)

The major concern of global optimisation techniques is how to resolve the problem of convergence in the presence of local minima. The solution may fall in the local minima at the beginning of the search, and may even become stagnant. The authors, Parsopoulos and Vrahatis (Parsopoulos and Vrahatis 2002b) proposed a modified PSO algorithm called "stretching" (SPSO) guided towards finding all available global minima.

The deflection, stretching and the repulsion techniques are integrated into the original PSO Algorithm. The first two techniques apply the concept of transforming the objective function by including the already found minimum points. The repulsion technique adds the ability to guarantee that all particles will not move toward the already found minima (Parsopoulos and Vrahatis 2002b), (Kannan *et al.* 2004). As a result, it is possible for the proposed algorithm to avoid already found solutions with more chances of finding the global optimal solution.

The equations are two-stage transformations. When a fitness function f is chosen for a problem, the first transformation stage transforms the original fitness function $f(x)$ into $G(x)$ with x representing any particle, which eliminates all the local minima that are located above $f(\bar{x})$, where $\bar{x}$ represents a detected local minimum

$$G(x) = f(x) + \gamma_1 \| x - \bar{x} \| . (\text{sgn}(f(x) - f(\bar{x})) + 1 \quad \text{(G.17)}$$

The second stage stretches the neighbourhood of $\bar{x}$ upwards, since it assigns higher function values to the points in the upward neighbourhood.

$$H(x) = G(x) + \gamma_2 \frac{\text{sgn}(f(x) - f(\bar{x})) + 1}{\tanh(\mu(G(x) - G(\bar{x})))} \quad \text{(G.18)}$$

In (G.17) and (G.18), γ_1, γ_2 and μ are randomly selected positive constants and sgn(y) is the triple valued sign function.

$$\text{sgn}(y) = \begin{cases} 1, & \text{If } y > 0 \\ 0, & \text{If } y = 0 \\ -1, & \text{If } y < 0 \end{cases} \quad \text{(G.19)}$$

The two stages do not adjust the local minima located below $\bar{x}$. Accordingly the location of the global minimum is left unchanged (Parsopoulos and Vrahatis 2002b).

G10.5 Cooperative PSO (CPSO)

The cooperative PSO (CPSO) Algorithm was put forward by Van den Bergh and Engelbrecht (Bergh and Engelbrecht 2004). The CPSO Algorithm utilises cooperative behaviour to improve the performance of the original PSO algorithm. It uses multiple swarms to optimise different components of the solution vector cooperatively.

This is analogous to the approach by Potter's cooperative co-evolutionary genetic algorithm (CCGA). The search space in the CPSO Algorithm is explicitly partitioned by dividing the solution vectors into smaller vectors. (Bergh and Engelbrecht 2004) proposed two new algorithms, the CPSO-S_k and CPSO-H_k.

In the CPSO-S algorithm a swarm having n-dimensional vectors is partitioned into n-swarms of one-dimensional vectors, each swarm optimising a single component of the solution vector. A credit assignment mechanism is used for the evaluation of each particle in each swarm. In the CPSO-S approach, only one component is modified at a time resulting in many combinations formed using different members from different swarms producing the desired fine-grained search plus a noteworthy increase in the solution diversity.

The CPSO-S_k is a modification of the preceding technique in which the position vector is divided in parts instead of n. In contrast, because the PSO has the ability to escape from pseudo-minimisers while the CPSO-S_k algorithm has faster convergence on some functions, the CPSO-H_k combines these two techniques by executing one iteration of CPSO-S_k followed by one iteration of the standard PSO algorithm.

(Baskar and Suganthan 2004) proposed a cooperative method titled the concurrent PSO (CONPSO) – here the problem hyperspace is implicitly partitioned by having two swarms searching concurrently for a solution with regular message exchange of information on the (g_{best}).

(El-Abd and Kamel 2006) proposed a hierarchical Cooperative Particle Swarm Optimiser which combines the implicit and explicit space decomposition methodology that was adopted in CPSO-S and CONPSO. This amalgamation of methodologies was achieved with two swarms concurrently searching for a solution with each one employing the CPSO-S technique. The authors in their results demonstrate that the proposed approach outperforms the CONPSO, the CPSO-S, and the CPSO-H on four selected benchmark functions: the Rosenbrock function – uni-modal, the Griewank function – multi-modal, the Ackley function – multi-modal, and the Rastrigin function – multi-modal (El-Abd and Kamel 2006).

G10.6 Comprehensive Learning PSO (CLPSO)

(Liang *et al.* 2006) modified the conventional equation for the velocity update to:

$$v_i^d(t) = \varphi_{ic} \cdot v_i^d(t-1) + \varphi \cdot \text{rand}_1 \cdot (pbest^d_{fi(d)} - x_i^d(t-1)) \qquad \text{(G.20)}$$

where d corresponds to the dimension index (d: 1→D) and $f_i(d)$ defines which particles' p_{best} the particle i should follow.

A random number is generated for each dimension of particle i; when this number is greater than a certain value Pc_i (Pc is the learning probability), the particle follows its own p_{best}, else it learns from another particle's p_{best}. In the second situation, a tournament selection is applied to determine which particle's p_{best} will be used.

(1) Two random particles are selected from the swarm

$$f1_i^d = \lceil \text{rand}_1 . ps \rceil$$

$$f2_i^d = \lceil \text{rand}_2 . ps \rceil \qquad \text{(G.21)}$$

where ps is the population size.

(2) Their p_{best} values are compared and the best one is selected.

(3) The winner particle is used as an example to learn from.

To ensure that the particles learn from good exemplars and to minimise the time wasted following poor directions, the particles are allowed to learn until a refreshing gap m, defined as a certain number of iterations, is reached. After that the values of f_i are reassigned for all particles in the swarm.

In the CLPSO algorithm, the parameters φ, Pc and m have to be tuned. In the case of the learning probability Pc, (Liang *et al.* 2006) have proposed using a different value for each particle to give them different levels of exploration and exploitation ability. In this scheme, the advantages of this learning strategy are that all the particles are potential leaders; the chances of getting trapped in local minima are reduced by the cooperative behavior of the swarm. In addition, the particles used different exemplars for each

dimension, which are renewed after some iterations (refreshing gap), giving more diversity in the searching process.

Bibliography (Appendix G)

Angeline, P. J. 1998. Using Selection to Improve Particle Swarm Optimisation. *Proceedings of IEEE International Comference on Evolutionary Computation*. IEEE Press. pp. 84 - 89

Bartz-Beielstein, T. and Limbourg, P. and Mehnen, J. and Schmitt, K. and Parapoulos, K. and Vrahatis, M. 2003. Particle Swarm optimisers for Pareto Optimisation with enhanced Archiving Techniques. *in Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1780 - 1787.

Baskar, S. and Suganthan, P. N. 2004. A Novel Concurrent Particle Swarm Optimisation. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 792 - 796.

Bergh, F. and Engelbrecht, A. 2004. A Cooperative Approach to Particle Swarm Optimisation. *IEEE Transactions on Evolutionary Computation* vol. 8(3), pp. 225 - 239.

Biskas, P. and Ziogos, N. and Tellidou, A. and Zoumas, C. and Bakirtzis, A. and Petridis, V. and Tsakoumis, A. 2005. Comparison of Two Metaheuristics with Mathematical Programming Methods for the Solution of OPF. *In Proceedings of International Conference on Intell. Syst. Appl.*, pp. 510 - 515.

Blackwell, T. and Bentley, P. 2002. "Don't push me! Collision-avoiding swarms". *in Proceeding IEEE Congress on Evolutionary Computation* vol. 2, pp. 1691 - 1696.

Carlisle, A. and Dozier, G. 2000. Adapting Particle Swarm Optimisation to Dynamic Environments. *International Conference on Artificial Intelligence (ICAI)*, pp. 429 - 434.

Cedeño, W. and Agrafiotis, D. 2005. A Comparison of Particle Swarm Techniques for the Development of Qualtitative Structure-Activity Relationships for Drug Design. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference - Workshops*. IEEE Computer Society. pp. 322 - 331

Coello, C. and Lechuga, M. 2002. MOPSO: A Proposal for Multiobjective Particle Swarm Optimisation. *In Proceeding IEEE Transaction on Evolutionary Computation* vol. 8(2), pp. 1051 - 1056.

Coello, C. and Pulido, G. and Lechuga, M. 2004. Handling Multiple Objectives with Particle Swarm Optimisation. *In Proceedings of IEEE Transaction on Evolutionary Computation* vol. 8(3), pp. 256 - 279.

Das, T. and Venayagamoorthy, G. 2006. Optimal Design of Power Systems Stabilisers using a Small Population Based PSO. *in Proceedings of IEEE PES General Meeting.*

Das, T. K. and Jetti, S. R. and Venayagamoorthy, G. K. 2006. Optimal Design of a SVC Controller using a Small Population based PSO. *In Proceedings of IEEE Swarm Intelligence Symposium*, pp. 156 - 161.

Doctor, S. and Vanayagamoorthy, G. and Grudise, V. 2004. Optimal PSO for Collective Robotics Search Applications. *In Proceeding IEEE Congress on Evolutionary Computation* vol. 2, pp. 1390 - 1395.

Doctor, S. and Venayagamoorthy, G. 2005. Improving the Performance of Particle Swarm Optimisation using Adaptive Critics Design. *In Proceedings of IEEE Swarm Intelligence Symposium*, pp. 393 - 396.

Eberhart, R. and Shi, Y. 2001. Tracking and Optimising Dynamic Systems with Particle Swarms. *in Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 94 - 100.

El-Abd, M. and Kamel, M. S. 2006. A Hierarchical Cooperative Particle Swarm Optimiser. *In Proceedings of IEEE Swarm Intelligence Symposium*, pp. 43 - 47.

El-Dib, A. and Youssef, H. and El-Metwally, M. and Osman, Z. eds. 2004. *In Proceedings International Conference on Elect., Electron., Comput., Eng.* IEEE

Engelbrecht, A. P. 2005. *Fundamentals of Computational Swarm Intelligence.* John Wiley and Sons Ltd, p. 672 pages.

Fieldsend, J. E. and Everson, R. M. and Singh, S. 2003. Using Unconstrained Elite Archives for Multiobjective Optimisation. *IEEE Transactions on Evolutionary Computation* vol. 7(3), pp. 305 - 323.

He, S. and Wen, J. and Prempain, E. and Wu, Q. and Fitch, J. and Mann, S. 2004. An Improved Particle Swarm Optimisation for Optimal Power Flow. *In Proceedings of International Conference on Power Systems Technologies*, pp. 1633 - 1637.

Hu, X. 2006. Particle Swarm Optimisation. *in Tutorial of the IEEE Swarm Intelliegence Symposium.*

Hu, X. and Eberhart, R. 2002a. Adaptive Particle Swarm Optimisation: Detection and Response to Dynamic Systems. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 2, pp. 1666 - 1670.

Hu, X. and Eberhart, R. 2002b. Multiobjective Optimisation using Dynamic Neighbourhood Particle Swarm Optimisation. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 2, pp. 1677 - 1681.

Hu, X. and Eberhart, R. and Shi, Y. 2002. Particle Swarm with Extended Memory for Multiobjective Optimisation. *In Proceedings of IEEE Swarm Intelligence Symposium*, pp. 193 - 197.

Hu, X. and Shi, Y. and Eberhart, R. 2004. Recent Advances in Particle Swarm. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 90 - 97.

Kannan, S. and Slochanal, S. M. and Subbaraj, P. and Padhy, N. P. 2004. Application of Particle Swarm optimisation Technique and its Variants to Generation Expansion Planning Problem. *ELSERVIER Electric Power Systems* vol. 70(3), pp. 203 - 210.

Khajenejad, M. and Afshinmanesh, F. and Marandi, A. and Araabi, B. N. 2006. Intelligent Particle Swarm Optimisation using Q-Learning. *in Proceedings of IEEE Swarm Intelligence Symposium*, pp. 7 - 12.

Krink, T. and Vesterstrom, J. and Riget, J. 2002. Particle Swarm Optimisation with Spacial particle extension. *In Proceeding IEEE Congress on Evolutionary Computation* vol. 2, pp. 1474 - 1479.

Krohling, R. 2004. Guassian Swarm: A Novel Particle Swarm Algorithm. *In Proceedings of IEEE Conference on Cybern. Intell. Syst.* vol. 1, pp. 372 - 376.

Krohling, R. 2005. Guassian Particle Swarms with Jumps. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 2, pp. 1226 - 1231.

Li, X. 2003. A Non-dominated Sorting Particle Swarm Optimiser for Multiobjective Optimisation. *In Proceedings of Genetic and Evolutionary Computation Conference*, pp. 37 - 48.

Li, X. 2004. Adaptively choosing Neighbourhood bests using Species in a Particle Swarm Optimiser for Multimodal Function Optimisation. *In Proceedings of Genetic and Evolutionary Computation Conference*, pp. 105 - 116.

Liang, J. and Qin, A. and Suganthan, P. and Baskar, S. 2006. Comprehensive Learning Particle Swarm Optimiser for Global Optimisation of Multimodal Functions. *in Proceedings of IEEE Transaction on Evolutionary Computation* vol. 10(3), pp. 281 - 295.

Lovbjerg, M. and Krink, T. 2002. Extending Particle Swarms with Self-Organised Criticality. *in Proceedings of IEEE Congress on Evolutionary Computation* Vol. 2, pp. 1588 - 1593.

Miranda, V. and Fonseca, N. 2002a. EPSO - Best-of-two-worlds Meta-heuristic applied to Power System Problems. *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02.* IEEE Computer Society.

Miranda, V. and Fonseca, N. 2002b. EPSO - Evolutionary Particle Swarm Optimisation, A New Algorithm with Applications in Power Systems. *Proceedings of IEEE/PES Transmission and Distribution Conference Exhibition* vol. 2, pp. 745 - 750.

Miranda, V. and Fonseca, N. 2002c. New Evolutionary Particle Swarm Algorithm (EPSO) applied to Voltage/VAR Control. *In Proceedings of 14th Power Systems Computation Conference.* Sevilha, Portugal:

Mohan, C. and Al-Kazemi, B. 2001. Discrete Particle Swarm Optimisation. *In Proceedings of Workshop on Particle Swarm Optimisation, Purdue School of Engineering and Technology.*

Moore, P. and Venayagamoorthy, G. 2005. Evolving Combinational Logic Circuits using a Hybrid Quantum Evolution and Particle Swarm Inspired Algorithm. *In Proceedings of the 2005 NASA/DoD Conference of Evolution Hardware (EH'05)*, pp. 97 - 102.

Moore, P. and Venayagamoorthy, G. 2006. Evolving Combinational Logic Circuits using Particle Swarm, Differential Evolution and Hybrid DEPSO. *Proceedings of the 2005 NASA/DoD Conference of Evolution Hardware (EH'05)* vol. 16(2), pp. 163 - 177.

Mostaghim, S. and Teich, J. 2003. Strategies for Finding Good Local Guides in Multiobjective Particle Swarm Optimisation (MOPSO). *in Proceedings of IEEE Swarm Intelligence Symposium*, pp. 26 - 33.

Naka, S. and Genji, T. and Yura, T. and Fukuyama, Y. 2003. A Hybrid Particle Swarm Optimisation for Distribution State Estimation. *in Proceedings of IEEE Transactions on Power Systems* vol. 18(1), pp. 60 - 68.

Parsopoulos, K. and Tasoulis, D. and Vrahatis, M. 2004. Multiobjective Optimisation using Parallel Vector Evaluated Particle Swarm Optimisation. *in Proceedings of International Conference on Artificial Intelligence Application* vol. 2, pp. 823 - 828.

Parsopoulos, K. and Vrahatis, M. 2002a. Particle Swarm Optimisation Method in Multiobjective Problems. *In Proceedings of ACM Symposium on Appl. Comput.*, pp. 603 - 607.

Parsopoulos, K. and Vrahatis, M. 2002b. Recent Approaches in Global Optimisation Problems through Particle Swarm Optimisation. *Natural Computing* vol. 1, pp. 253 - 306.

Ray, T. 2002. Constrained Robust Optimal Design using a Multiobjective Evolutionary Algorithm. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 419 - 424.

Ray, T. and Liew, K. 2002. A Swarm Metaphor for Multi-objective Design Optimisation. *Engineering Optimisation*. Vol. 34, no. 2. pp. 141 - 153.

Salazar-Lechuga, M. and Rowe, J. E. 2005. Particle Swarm Optimisation and Fitness Sharing to solve Multi-Objective Optimisation Problems. *In Proceedings of IEEE Swarm Intelligence Symposium* vol. 2, pp. 1204 - 1211.

Secrest, B. R. and Lamont, G. B. 2003. Visualising Particle Swarm Optimisation - Gaussian Particle Swarm Optimisation. *In Proceedings of IEEE Swarm Intelligence Symposium*, pp. 198 - 204.

Shi, Y. 2004. Feature Article on Particle Swarm Optimisation. *IEEE Neural Network Society*, pp. 8 - 13.

Shi, Y. and Eberhart, R. C. 2001a. Fuzzy Adaptive Particle Swarm Optimisation.*Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press. pp. 101 - 106

Shi, Y. and Eberhart, R. C. 2001b. Particle Swarm Optimisation with Fuzzy Adaptive Inertia Weight. *in Proceedings of Workshop on Particle Swarm Optimisation, Purdue School of Engineering and Technology*. Indianapolis, IN:

Talbi, H. and Batouche, M. 2004. Hybrid Particle Swarm with Differential Evolution for Multimodal Image Registration. *In Proceedings IEEE International Conference on Industrial Technology (ICIT)* vol. 3, pp. 1567 - 1572.

Valle, Y. D. and Hernandez, J. C. and Venayagamoorthy, G. and Harley, R. G. 2006. Multiple STATCOM Allocation and Sizing using Particle Swarm Optimisation. *in Proceedings of IEEE PES Power Syst. Conference Expo.*, pp. 1884 - 1891.

Valle, Y. D. and Venayagamoorthy, G. K. and Mohagheghi, S. and Hernandez, J. and Harley, R. G. 2008. Particle Swarm Optimisation: Basic Concepts, Variants and Applications in Power Systems. *in Proceedings IEEE Transactions on Evolutionary Computation* Vol. 12(2), pp. 171 - 195.

Venayagamoorthy, G. 2004. Adaptive Critics for Dynamic Particle Swarm Optimisation. *In Proceedings of IEEE International Symposium on Intelligent Control*, pp. 380 - 384.

Xie, X. and Zhang, W. and Yang, Z. 2002. A Dissipative Particle Swarm Optimisation. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 2, pp. 1456 - 1461.

Yang, S. and Wang, M. and Jiao, L. 2004. A Quantum Particle Swarm Optimisation. *In Proceedings of IEEE Congress on Evolutionary Computation* vol. 1, pp. 320 - 324.

Zhang, W. and Liu, Y. and Clerc, M. 2003. An Adaptive PSO Algorithm for Reactive Power Optimisation. *in Proceedings of 6th International Conference on Advances in Power Systems Control, Operations and Management (APSCOM)*. Hong Kong: pp. 302 - 307

Zhang, W. and Xie, X. 2003. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. *IEEE International Conference on Systems, Man & Cybernetics (SMCC)* vol. 4, pp. 3816 - 3821.

Appendix H

Review of the Bees Algorithm

Bees-inspired algorithms are motivated by the natural behaviour of swarms of bees (Yang 2008). The foraging behaviour of swarms of honey bees (Seeley 1996) and the selection of nesting site (Passino and Seeley 2006) have been modelled computationally and employed as optimisation methods in both combinatorial and continuous search space.

The honey bee algorithm was proposed by Tovey (Tovey 2004) implemented to optimise an internet server. The BeeHive algorithm by Waddle *et al.* (Wedde *et al.* 2004) was applied to routing problems in packet switching networks (Muddassar 2009) where the agents (BeeAgents) are used to route packets among network nodes.

A further implementation of the bee behaviour called the Bee Colony Optimisation was presented by Teodorovic and Dell'orco (Teodorovic and Dell'orco 2005) to solve transportation problems. This algorithm employs a constructive approach which is similar to Ant Colony Optimisation Algorithm.

Later, Yang (Yang 2005) presented the Virtual Bees Algorithm (VBA) as a model of the natural foraging behaviour of honey bees. The algorithm had PSO-like parameters that were implemented to solve continuous optimisation problems. Karaboga and Basturk (Karaboga and Basturk 2008) developed the Artificial Bees Colony (ABC) algorithm inspired by the foraging behaviour of honey bees that has been successfully applied to continuous optimisation problems.

Quijano and Passino (Quijano and Passino 2007a, b) proposed a model of the social foraging behaviour of honey bee as an algorithm to solve optimal resource allocation problems.

Bibliography (Appendix H)

Karaboga, D. and Basturk, B. 2008. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing* 8(1), pp. 687-697.

Muddassar, F. 2009. Bio-inspired telecommunications.*Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. Montreal, Qu\&\#233;bec, Canada: ACM.

Passino, K. M. and Seeley, T. D. 2006. Modeling and analysis of nest-site selection by honeybee swarms: the speed and accuracy trade-off. *Behav. Ecol. Sociobiol.* 59(4), pp. 27 - 42.

Quijano, N. and Passino, K. M. 2007a. Honey Bee Social Foraging Algorithms for Resource Allocation. *Part I: Algorithm and Theory - Proceedings of the 2007 American Control Conference New York City, NY*, pp. 3383 - 3388.

Quijano, N. and Passino, K. M. 2007b. Honey Bee Social Foraging Algorithms for Resource Allocation. *Part II: Algorithm and Theory - Proceedings of the 2007 American Control Conference New York City, NY*, pp. 3389 - 3394.

Seeley, T. D. 1996. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Cambridge, Massachusetts: Harvard University Press.

Teodorovic, D. and Dell'orco, M. 2005. Bee colony optimization - A cooperative learning approach to complex transportation problems. *Advanced OR and AI Methods in Transportation*, pp. 51 - 60.

Tovey, C. A. 2004. *The Honey Bee Algorithm: A Biologically Inspired Approach to Internet Server Optimization.* Engineering Enterprise Magazine. Spring 2004. pp. 13-15.

Wedde, H. F. and Farooq, M. and Zhang, Y. 2004. BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. *Lecture Notes in Computer Science.* pp. 83-94.

Yang, S. X. 2008. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yang, X.-S. 2005. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. *IWINAC (2)*. Vol. 3562. Springer, pp. 317-323.