IMPROVED VERSIONS OF THE BEES ALGORITHM FOR GLOBAL OPTIMISATION

By

SHAFIE KAMARUDDIN

A thesis submitted to the

University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

School of Engineering

College of Engineering and Physical Sciences

University of Birmingham

June 2017

ABSTRACT

This research focuses on swarm-based optimisation algorithms, specifically the Bees Algorithm. The basic version of the algorithm was introduced in 2005 and was inspired by the foraging behaviour of honey bees in nature. The Bees Algorithm employs a combination of exploration and exploitation to find the solutions of optimisation problems.

This thesis presents three improved versions of the Bees Algorithm aimed at speeding up its operation and facilitating the location of the global optimum. For the first improvement, an algorithm referred to as the Nelder and Mead Bees Algorithm (NMBA) was developed to provide a guiding direction during the neighbourhood search stage. The second improved algorithm, named the recombination-based Bees Algorithm (*r*BA), is a variant of the Bees Algorithm that utilises a recombination operator between the exploited and abandoned sites to produce new candidates closer to optimal solutions. The third improved Bees Algorithm, called the guided global best Bees Algorithm (*g*BA), introduces a new neighbourhood shrinking strategy based on the best solution so far for a more effective exploitation search and develops a new bee recruitment mechanism to reduce the number of parameters.

The proposed algorithms were tested on a set of unconstrained numerical functions and constrained mechanical engineering design problems. The performance of the algorithms on numerical functions was compared with the standard Bees Algorithm and other swarm based algorithms in terms of the solutions found and the convergence speed. In terms of the application on constraint mechanical engineering design problems, the performance was compared with the results of the standard Bees Algorithm and the results of other algorithms in the literature. In addition, a paired test statistical analysis was also carried out on those results.

The results showed that the improved Bees Algorithms performed better than the standard Bees Algorithm and other algorithms on most of the problems tested. Furthermore, the algorithms also involve no additional parameters and a reduction on the number of parameters as well.

ACKNOWLEDGEMENTS

First and foremost, gratitude and Praise is to Allah, the Most Merciful and Most Compassionate for His Blessings, that I am to complete this research. I would also like to record my appreciation to these outstanding individuals for their great contributions:

I am deeply indebted to my supervisor, Prof. D T Pham for all his guidance, advices, knowledge and supervision that guided me throughout my academic program. His immeasurable supports and confidence in me inspired me to complete this thesis.

In addition, I also would like to thanks my wife, Dahlia Nursyarmimi and all my family members for always supporting me during my PhD journey.

Furthermore, I would like also to express my outermost appreciation to all my colleagues, Muhammad Syahril, Nik Mohd Farid, Dr. Silah Hayati, Al Antoni Akhmad and other closet friends for their encouragements and advices during my research period

Also, special thanks to my sponsor, the Ministry of Higher Education (MOHE) of Malaysia and International Islamic University Malaysia (IIUM) for giving me the opportunity to further my study.

Lastly, I offer my regards and blessing to all of those supported me in any respect during completion of the research.

This thesis was copy edited for conventions of language, spelling, and grammar by Szakif Enterprise Academic Services.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

f	Function that represent the optimisation problem	6
$\boldsymbol{x}$	design or decision variables	6
$\mathfrak{R}$	Search space	6
N	Number of variables to be optimised	6
h	Constraint function	6
$\boldsymbol{x}_w$	Worst point	8
$\boldsymbol{x}_b$	Best point	8
$\boldsymbol{x}_g$	Good point	8
$\boldsymbol{x}_c$	Midpoint	9
$\boldsymbol{x}_r$	Reflected point	9
f_{min}	Minimun value of objective function	9
$\boldsymbol{x}_0$	Base point	9
$\boldsymbol{x}_1$	Temporary vector to store temporary point	9
$\boldsymbol{x}_2$	Point obtained by pattern move	9
δ	Step sizes	9
T	Temperature parameter	15
I	Light Intensity	18
D	Number of dimensions	28
X	Vector of variables to be optimised	28
ns	Scout bees	28
nb	Best sites	29
ne	Elite sites	29
ns-nb	The remaining unselected scout bees	29
ngh	Neighbourhood size	29
nre	Bees recruited for elite sites	29
nrb	Bees recruited for best sites	29
ngh(t)	Neighbourhood size at t^{th} iteration	29
t	Number of iteration	23
max_i	Upper domain for variables	23

min_i	Lower domain for variables	23
stlim	Threshold for site abandonment	23
ssngh	Self - adaptive neighbourhood size	34
W	Worst position	41
B	Best position	41
G	Good position	41
d	Distance	41
R	Reflection position	42
E	Extension position	42
M	Mid - point position	42
S	Shrinkage position	42
$\boldsymbol{C_1}$ *or* $\boldsymbol{C_2}$	Centroid position	42
$\boldsymbol{B–G}$	Position between $\boldsymbol{B}$ and $\boldsymbol{G}$	42
$\boldsymbol{B–W}$	Position between $\boldsymbol{B}$ and $\boldsymbol{W}$	42
j	Index of dimension	94
$\mathrm{P}_{Global,j}$	Position of best solution found so far at *j-th* index dimension	95
$\mathrm{P}_{best,j}$	Position of unimproved site at *j-th* index dimension	95
ngh_{x1}	Neighbourhood size for variable x_1	95
ngh_{x2}	Neighbourhood size for variable x_2	95
x_B	Best bee at current iteration	95
x_G	Best solution found so far by the bee	99

CHAPTER 1

Introduction

1.1 Background

Nowadays, many real-world engineering problems involve optimisation, which is the act of attaining the best possible results under given constraints. The most common goal of this optimisation problem is either to maximise (profit, output, efficiency, etc.), or minimise (time, cost, effort, etc.). In order to achieve this goal, various types of optimisation approaches have been developed to deal with optimisation problems. Recently, population- based metaheuristic inspired by natures types of algorithms have attracted a huge attention as the traditional optimisation methods (linear or integer programming) are not adequate to provide best results, due to the complexity of the problems. Among the most common algorithms of this type are Evolutionary Algorithm (EA), Particle Swarm Optimisation (PSO) algorithm, Artificial Bee Colony (ABC) algorithm and the Bees Algorithm.

This research focuses on one of the algorithms mentioned above, which is the Bees Algorithm; an algorithm inspired by the foraging behaviour of honey bee swarms in nature.

1.2 Motivation

The Bees Algorithm is one of the nature-inspired swarm-based optimisation methods inspired by the foraging behaviour of honey bees. It was introduced by a group of researchers at Cardiff University in 2005 (Pham et al., 2005). The algorithm consists of combinations between exploration strategy and exploitation strategy representing the activities of scout bees and recruit bees in a bee colony respectively. Since its establishment in optimisation, numerous improved versions have been proposed to deal with different types of optimisation problems.

The performance of the Bees Algorithm on those problems is superior compared to other state-of-the-art algorithms. However, the “No Free Lunch Theorem” showed that the performances of all algorithms are generally similar when their performance is averaged uniformly over all possible problems. This is because an algorithm that performs better on a class of problem will not perform better on other class of problems (Wolpert and Macready, 1997). Thus, further enhancements to the Bees Algorithm are required to provide alternatives variants of the Bees Algorithm for a wider range of problems.

Previous studies on improving the Bees Algorithm mainly focus on modification of the neighbourhood search but usually those improvements add more parameters to the algorithm. Hence, developing an improved Bees Algorithm without adding more parameters is one of the challenges faced by researchers as the Bees Algorithm has large number of parameters to be tuned. Besides, there is also a need to address the issue of improving the convergence speed of the Bees Algorithm caused by random search directions near the global optima.

In the current operation of the Bees Algorithm, the recruit bees are placed randomly nearby the exploited sites. This random search of recruit bees causes slow convergence speed on the Bees Algorithm or any other random-based algorithm as well. Thus, adding direction information to the recruit bees would help the neighbourhood search in discovering solutions faster than the current approach.

Another issue is lack of information sharing among exploited sites. The best solutions found by the neighbourhood search and the abandoned sites consist of highly good solutions. Exchanging information between these solutions via combination of partial solutions can produce better solutions.

Furthermore, reducing the number of parameters of the Bees Algorithm is one of the main motivations of this research. A small number of parameters would be more beneficial because it will require less tuning for the users. In addition, there is also a lack of study done on neighbourhood shrinking strategy for the unimproved sites. The existing approaches of dealing with unimproved sites are reducing the neighbourhood size in all dimensions, followed by sites abandonment after a predefined consecutive failure to find a better solution. Therefore, it would be beneficial to explore better approaches of neighbourhood shrinking strategy especially for sites being searched near the global optimum. The neighbourhood search nearby the global optimum could cause slow convergence for some types of landscape problems. In order to overcome this problem, the algorithm could rely on the information from the best solution so far to guide the neighbourhood shrinking strategy for better exploitation.

1.3 Aim and Objectives

The overall aim of this study is to further enhance the capability of the Bees Algorithm in dealing with single objective optimisation problems without adding more parameters.

The following objectives were set to accomplish this aim:

i. Provide a direction during the neighbourhood search by using the Nelder and Mead (NM) method to guide the search toward better solution with faster convergence speed.
ii. Implement recombination operator between exploited and abandoned sites to move the solutions found closer to the local or global optima.
iii. Develop a new self-adaptive bee recruitment mechanism to reduce the number of parameters.
iv. Develop a guided neighbourhood shrinking strategy based on best solution found so far.

1.4 Research Methodology

The methodology adopted in this research is as follows:

i. Reviewing previous works on optimisation techniques, focusing on nature-inspired swarm intelligence and more on algorithms based on behaviour of honey bees in nature to know research trends and discover potential solutions.
ii. Developing the proposed Bees Algorithms in ***R*** software.
iii. Evaluating the performance of improved version of the Bees Algorithms on a set of unconstrained continuous numerical function. The results were compared with the standard Bees Algorithm and other swarm-based algorithms in terms of solution found and convergence speed.
iv. Applying the proposed algorithms to constrained mechanical design problems. The results were compared with results of other algorithms obtained in literature.

1.5 Outline of the Thesis

The remainder of this thesis is organised as follows:

Chapter 2 reviews the definition of optimisation, followed by a few conventional approaches for solving optimisation problems. This chapter also reviews two types of metaheuristic; single solution-based metaheuristic and population-based metaheuristic. The review on metaheuristic focuses on nature inspired Swarm Intelligence (SI), specifically more on algorithms based on the foraging behaviour of honey bees. The studies related to the Bees Algorithm are also discussed in detail.

Chapter 3 introduces the Bees Algorithm with the Nelder and Mead (NM) method. The proposed algorithm is tested on a set of numerical benchmark functions. The results are compared with the standard Bees Algorithm and other well-known algorithms in terms of solutions found and convergence speed. In addition, the improved Bees Algorithm is also applied on several constraint benchmark mechanical design problems.

Chapter 4 presents an improved Bees Algorithm that utilises recombination operator during local search and on best abandoned sites. The performance of the proposed algorithm is tested on similar benchmark function and mechanical design problems as in the earlier chapter. Similar methods of comparison as in the previous chapter are also used in this chapter.

Chapter 5 introduces an adaptive recruitment bee mechanism into the standard Bees Algorithm. This version of Bees Algorithm also added a new neighbourhood shrinking strategy that utilises the information from the best solution found so far. Similarly, like previous chapters, this variant of Bees Algorithm is also tested on a similar set of numerical functions. In addition to the mechanical design problems in the previous chapters, this chapter also includes another application on a mechanical design problem.

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

CHAPTER 2

Literature Review

2.1 Preliminaries

This chapter introduces optimisation and presents some free derivative direct search methods. Then, main metaheuristic techniques which are commonly used to solve optimisation problems are reviewed. The chapter further reviews on the mechanism of the Bees Algorithm, improvements done on the Bees Algorithm and applications of the Bees Algorithm.

2.2 Optimisation

In general, optimisation is to find the best combination of variables for a given problem. Most of these optimisation problems involve a wide range of domains and are common in real engineering problems, which required searching of the optimal solution or near optimal solution. The task of optimisation becomes more difficult due to limitation of finance, time and resources. Generally, optimisation problems can be expressed in mathematical form subject to some constraints and a range of variables as follows:

$$\text{Minimise (or maximise) } f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, x_1, \ldots\ldots, x_N),$$

$$\boldsymbol{x} \in \mathfrak{R}^N$$

$$\text{subject to } h_1(\boldsymbol{x}) = 0,\ \boldsymbol{x} = (x_1, x_1, \ldots\ldots, x_N),$$

$$h_2(\boldsymbol{x}) \leq 0,\ \boldsymbol{x} = (x_1, x_1, \ldots\ldots, x_N),$$

where

$f(\boldsymbol{x})$ = called objective function or cost function(s),

$\boldsymbol{x}$ = called design or decision variables, can be real continuous, discrete or a combination of both,

$\mathfrak{R}^N$ = search space of the design/decision variables, N = number of variables to be optimised

The equalities for h_1 and inequalities for h_2 are called constraints.

The main objective of optimisation is to search for optimal solution or satisfactory solution once the problem has been expressed correctly. The conventional approach of solving these optimisation problems is using gradient-based or derivative-based algorithms. However, these methods require knowledge of gradient value of the objective function or constraint function. The information on derivative value or gradient value may not be available or difficult to be computed for some problems. For this reason, the performance of these types of algorithm has limited capability on certain types of problems only and therefore not suitable to be applied on more complex problems (i.e. multi-objective optimisation problems, large scale optimisation problems). Other conventional approaches of solving optimisation are using the gradient descent method known as direct search method, where no gradient information is required. This method relies on the value of objective function only to find better solutions.

Another alternative approach to find a satisfactory solution if the conventional approaches failed to obtain the exact solution is by using a method known as the 'heuristic' method. The word heuristic means *to find* (Greek verb). This method uses probabilistic rule instead of deterministic to find best optimal solutions. It is also has been successfully used to find satisfactory solutions on a variety of optimisation problems but afterwards, Glover (1986) coined a new term called 'Metaheuristic', which means high level (meaning of meta in Greek prefix) of heuristic method. The next sections describe more details on direct search methods, followed by metaheuristic algorithms.

2.2 Direct search methods

Direct search methods refer to an approach of finding the minimum value of a function using direction without computing or approximating the gradient values of the objective functions (Kolda et al., 2003). These methods have been through many improvements since its introduction in the early 1960s up until today. Although more popular and advanced techniques have been discovered in the area of numerical optimisation, direct search methods remain as a reliable alternative for the users (Lewis et al., 2000). One of the reasons direct search method remains popular is because of their easy and simple implementation. Another reason is that they only require few parameter settings compared to sophisticated optimisation techniques. The next section briefly describes two examples of direct search methods in detail.

2.2.1 Nelder and Mead Method

One of the popular direct search methods is known as the Nelder and Mead (NM) method, which uses simplex formed by (N + 1) points in N dimensional space to find the optimum value. Originally, this method was introduced by Spendley et al. (1962) that only described the reflection of the worst point via a midpoint of the opposite face to form a new simplex. Later, Nelder and Mead (1964) developed this original method into an optimisation algorithm by adding a few more additional moves to speed up the convergence speed. Those additional moves are expansion, contraction and shrinking operations. The steps of NM methods are as described below:

i. Randomly generate (N+1) points (solutions) across the search space. Evaluate the fitness values *f* of objective function for all the points generated. Identify the worst point ($\boldsymbol{x}_w$), best point ($\boldsymbol{x}_b$) and good point ($\boldsymbol{x}_g$) from all those points based on fitness value.

Then, reflect the worst point through the midpoint ($\boldsymbol{x}_c$) of best point ($\boldsymbol{x}_b$) and good point ($\boldsymbol{x}_g$).

ii. If the new reflected point ($\boldsymbol{x}_r$) found is a better solution than best point ($\boldsymbol{x}_b$), the reflection point extends towards the same direction as reflection operation. This operation is known as expansion.

iii. If the reflection operation failed to find a better solution than best point ($\boldsymbol{x}_b$), an operation called contraction is performed where the reflection point contracts back towards the opposite direction between the midpoint ($\boldsymbol{x}_c$) and reflected point.

iv. If the contraction operation also failed to find a better solution, the initial simplex shrinks into a smaller simplex consisting of best point ($\boldsymbol{x}_b$), midpoint of ($\boldsymbol{x}_b$) - ($\boldsymbol{x}_g$) and midpoint of ($\boldsymbol{x}_b$) - ($\boldsymbol{x}_w$).

2.2.2 Hooke and Jeeves Method

Another example of a direct search method is the Hooke and Jeeves (HJ) pattern search, which was originally proposed by Hooke and Jeeves (1961). In this strategy, the HJ method also does not require knowledge about gradient values. The mechanism of the HJ method consists of repetitive combinations of exploratory moves and pattern moves about a base point (current solution). It begins with the identification of appropriate direction (exploratory move) by changing all variables at a time based on predefined steps on both directions. Then, a pattern move is made according to the established direction (pattern move). The details of the basic HJ pattern search are described as follows (Gao et al., 2013):

Assume that the $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $f_{\min}$ and $\boldsymbol{\delta} = (\delta_1, \delta_2,..\delta_D)$ are base point (current solution), temporary vector to store the obtained point after exploratory move, point obtained by pattern move, minimum objective function found so far, and step sizes of D directions, respectively

i. First, obtain a point by moving the base point along each dimensional direction at a time (exploratory move). The steps involved in the exploratory move are described as follows:

   Step 1: Initialise $\boldsymbol{x}_1 = \boldsymbol{x}_0, f_{\min} = f(\boldsymbol{x}_0)$, $i = 1$.

   Step 2: Set $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i} + \delta_i$, if $(f(\boldsymbol{x}_1) < f_{\min}), f_{\min} = f(\boldsymbol{x}_1)$, go to step 4; else go to step 3.

   Step 3: Set $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i} - \delta_i$, if $(f(\boldsymbol{x}_1) < f_{\min}), f_{\min} = f(\boldsymbol{x}_1)$, go to step 4; else $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i}$.

   Step 4: If $i < n$, set $i = i + 1$ and go to step 2; else go to step 5.

   Step 5: If $f_{\min} < f(\boldsymbol{x}_0)$, the exploratory move is successful; else it is failing.

ii. If the move is successful, the pattern move is made according to the following formula to generate a new base point: $\boldsymbol{x}_2 = \boldsymbol{x}_1 + (\boldsymbol{x}_1 - \boldsymbol{x}_0)$

   After the new base point is generated, the exploratory move is repeated.

iii. If the move is failing, repeat the exploratory move with smaller step size or terminate the HJ procedures if stopping criterion has been met.

2.3 Metaheuristic

A metaheuristic is an algorithmic framework that is described as a high-level of heuristic approach, designed for a wide range of optimisation problems. Most of the well-known metaheuristics have similar characteristics, which are nature-inspired and include randomness, not using any derivative information and require several parameters to be tuned by users (Boussaïd, et al., 2013). However, it is not necessary for a metaheuristic to consist of all those characteristics as different metaheuristics can vary differently according to their foundations.

The essential component to establish a successful metaheuristic is to keep the balance between exploration (diversification) and exploitation (intensification) on a given problem. This balance helps metaheuristics to identify potential good areas in the search space that has good solution, followed by finding the solution in the identified promising area. Thus, this enables the searching process to focus the search efforts only on regions that have been identified as promising areas, avoiding unnecessary search efforts on those other regions.

In recent years, there has been an increasing amount of literature on the development of metaheuristics due to its effectiveness. Much of the literature available deals with the question on how to improve the existing metaheuristics or applying it on new specific problems. The most likely causes of this recent development in metaheuristics is because more advanced computers with better processing power are available.

Furthermore, metaheuristics are generally classified into two types: single-solution based metaheuristic, also known as *trajectory methods*, and population-based metaheuristic. Single-solution based metaheuristic focuses on finding solutions that begin with single point candidate solution by moving it away, describing a trajectory in the search space. In contrast, population-based metaheuristic uses a set of solution simultaneously instead of a single solution.

2.4 Single-solution based metaheuristics

The single-solution based metaheuristic search mechanism works by repetitively doing slight changes (move search) to a single solution. In each iteration, a new solution is selected from the neighbourhood of previous solutions. These movements are described as trajectory through the search space. In general, different single-solution based metaheuristics use different search movement. This section reviews main examples of single-solution based metaheuristics available in the literature, which include Simulated Annealing (SA), Tabu Search (TS) and Iterated Local Search (ILS).

2.4.1 Simulated Annealing (SA)

The Simulated Annealing (SA) algorithm was initially proposed by Kirkpatrick et al. (1983) based on an algorithm described by Metropolis et al. (1953). The algorithm inspired by the annealing process used by metallurgists. This process involves heating of a material to a high temperature followed by cooling it slowly. The algorithm works by selecting a random solution of $\boldsymbol{x}'$ within the neighbourhood of present solution $\boldsymbol{x}$. The acceptance of $\boldsymbol{x}'$ as the new present solution is according to the following probability:

$$e^{-[f(x\prime)-f(x)]/T}$$

where

$f(\boldsymbol{x})$ = objective function,

T = Temperature parameter.

Parameter T is set to a high value at the beginning of the search, and decreased gradually during the search. Thus, this algorithm has high acceptance probability initially and gradually reduced as the search progresses. So far SA has been successfully applied in different types of applications including continuous and discrete problems (Blum and Roli, 2003; Boussaïd et al., 2013).

2.4.2 Tabu Search (TS)

Tabu Search (TS) is an algorithm that uses memory-based strategies to escape the local optimum and guide the search towards a better solution (Glover, 1986). In general, this algorithm memorises solutions that have been visited previously, preventing the search to visit positions memorised by the algorithm. The list of memorised solutions is called the tabu list.

The tabu list usually updates the list of solutions by replacing the oldest solution with a current solution. One of the most important elements in TS involves determining the length of the tabu list. Although memorising a complete solution in the tabu list would be beneficial in terms of performance, but it requires a large amount of space and time. Thus, this length needs to be set up in order to control the memory of the search process. In order to overcome this issue, Battiti and Tecchiolli (1994) proposed an improved algorithm called the Reactive TS that uses adaptive length tabu list.

2.4.3 Iterated Local Search (ILS)

Iterated Local Search (ILS) is a metaheuristic that makes use the mechanism to escape from local optimum to another promising region in the search space. The framework and features of ILS are described by Stutzle (1998) by highlighting the components of ILS in other algorithms. In general, the basic ILS produces new starting solutions of the next iteration based on the local optimum of current solutions using perturbation mechanisms. The main idea of applying this perturbation mechanism on the local optimum is most likely that the new solution will be produced at a more promising basin. However, determining the level of this perturbation mechanism is crucial for this type of metaheuristic as low perturbation would not suffice to get it to escape from the local optimum whereas high perturbation is similar to generating a randomly new solution (Boussaïd et al., 2013).

2.5 Population-based metaheuristics

Much of the current literature on metaheuristic pays attention to population-based metaheuristics rather than single-solution based metaheuristics, especially metaheuristics inspired by nature (Swarm Intelligence). The rationale of using a set of solution (population) instead of a single solution is the capability to exchange attributes among high quality solutions that will lead to finding good solutions. Population-based metaheuristics are mainly classified into three types: Evolutionary Computation (EC), Nature-Inspired Swarm Intelligence (SI) and other Evolutionary Algorithms (EA). This system of classification provides a basis of identifying different types of algorithms for population-based metaheuristics.

This section describes main algorithms related to population-based metaheuristics. For each category of population-based metaheuristic, a brief description of the algorithms is given with more emphasis on SI algorithms.

2.5.1 Evolutionary Computation (EC)

Evolutionary Computation (EC) is a group of optimisation algorithms inspired by the mechanism of biological evolution and behaviours of living organism (Zhang et al., 2011). Most of these EC algorithms (also known as Evolutionary Algorithms (EA)) have similar framework which is described as the following:

1. Population initialisation
2. Fitness evaluation
3. Repeat these steps until termination condition is met:
    i. Select parents (individual with better fitness has higher probability to be selected).
    ii. Produce new off-springs using variation operators (i.e., crossover, mutation).
    iii. Evaluate new individuals.
    iv. Select individuals for the next generation.
4. End

The framework described above is a basic form of EC algorithm. Despite the variety of EC algorithms available, they all have similar framework as described above. The first step in EC algorithms is population initialisation where a set of solution is generated randomly across the search space. Then, that population of initial solution is evaluated using a fitness function or objective function followed by selection mechanism. Once the fitness function has been evaluated, the algorithm enters the evolutionary iteration which, consists of reproduction and variation of new solutions until stopping the criterion is met.

Nowadays, a considerable amount of literature has been published on EC. These studies include improving the algorithm, using it in real world applications, and investigating the performance. This section describes some main examples of EC algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Genetic Programming (GP) and others. Usually, algorithms classified under this have different ways of variable representation (i.e. real, binary), types of selection mechanism (i.e. roulette wheel, fitness ranking), and types of genetic operators (i.e. crossover, mutation).

Among the most well-known and successful EC methods is Genetic Algorithm (GA). This method is inspired by the evolution of natural population according to natural selection and survival of the fittest mentioned by Charles Darwin in the *Origin of the Species* (Beasley et al., 1993a). The algorithm was originally introduced by John Holland at the University of Michigan.

In GA, the candidate solution is represented by a binary string known as *chromosome*. The main operators used by GA to create new solutions are crossover and mutation operators. During the reproduction stage, solutions from the population are selected and recombined to produce new solutions called offspring. The selection of chromosomes to undergo crossover and mutation is based on the fitness of individual chromosomes. A better fit chromosome has a higher chance to be selected compared to low fit chromosomes (Beasley et al., 1993a; Beasley et al., 1993b).

Another recent popular EC method is an algorithm known as Differential Evolution (DE) algorithm. This algorithm has gained the interest of researchers due to it simple work mechanism, few parameters and faster convergence rate compared to other evolutionary algorithms (Das et al., 2016). The algorithm is inspired by the principle of natural evolution, similar to GA. Even though both algorithms were inspired by the principle of evolution but the DE algorithm used real values to represent candidate solutions. The DE algorithm was introduced by Storn and Price, (1997) to find the global optimum of multidimensional real valued functions.

The basic form of the DE algorithm also uses mutation and recombination operator to produce new candidate solutions. The algorithm starts by generating a population of solutions randomly across the search space. Then, new solutions are produced by adding the weighted difference between two individuals to a third individual solution at a randomly indexed dimension. The individuals selected for this mutation operator are selected from the population. After performing mutation, recombination takes place. The comparison of fitness between the current solution and old solution determines whether an individual is retained in the population or replaced by the new solutions.

Another type of EC algorithm is known as Genetic Programming (GP). This method was popularised by Koza (1994), inspired by biological evolution and survival of the fittest as well. In general, it works similar to GA but uses a program representation instead of a fixed length of strings. Thus in GP, a population of computer programs is evolved to a better population of programs (Poli et al., 2008). The most common way of expressing these programs are by syntax trees instead of line of codes. This way enables it to be described in a flexible way of LISP language, which is the original programming language used by J. Koza. The leaves of the trees represent variables and constants (*terminals*) of the programs, whereas internal nodes (*functions*) represent arithmetic operations. Both of these terminals and functions produce the alphabet of the programs (Boussaïd et al., 2013).

Besides those EC algorithms described earlier, it is worth to mention a few other EC algorithms such as Evolutionary Strategies (ES), Evolutionary Programming (EP) and Estimation Distribution Algorithm (EDA). Similarly, like other EC algorithms, these algorithms are also inspired by biological evolution and survival of the fittest.

2.5.2 Nature-inspired Swarm Intelligence (SI)

Another type of population-based metaheuristic is a type of algorithm inspired by the Swarm Intelligence (SI). The concept of Swarm Intelligence (SI) was introduced by Beni (2005) when working on Artificial Intelligence, where it is described as the collective behaviour of a decentralised, self-organised system, natural or artificial. Examples of such behaviour in nature are such as flocks of birds, colonies of bees and shoals of fish. One of the interesting elements in this SI system is that it does not have a centralised control system but relies on the capability of exchanging information among individuals (Corne et al., 2012). Another crucial component for this SI system is its effectiveness of allocating different tasks to specific individuals simultaneously (Seeley, 1995). These good attributes of SI behaviour have led to the establishment of a type of swarm-based algorithms in optimisation.

Recent trends in optimisation methods showed that swarm-based optimisation algorithms have gain interest among users to solve optimisation problems. Usually, these swarm-based algorithms are inspired by collective behaviour of a colony of insects or other animals to find near optimal solutions. Several optimisation algorithms inspired by swarm behaviour in nature have been proposed in the literature. Examples of swarm-based optimisation algorithms inspired by nature are Firefly Algorithm, Ant Colony Optimisation (ACO) algorithm, Particle Swarm Optimisation (PSO) algorithm and Bees Inspired algorithm.

2.5.2.1 Firefly Algorithm (FA)

Firefly Algorithm (FA) is an algorithm inspired by the flashing lights of fireflies in nature. The flashing lights emitted by those fireflies is produced by a biochemical process bioluminescence to attract mating partners or warn predators as firefly light are associated with bad taste (Fister et al., 2013). Based on this behaviour of fireflies, Yang (2010) introduced the basic form of FA which follows these three rules:

i. All fireflies are attracted to each other regardless of their sex.
ii. Attractiveness is directly proportional to brightness and both are inversely proportional to distance.
iii. Brightness (I = light intensity) corresponds to landscapes of objective function or fitness function.

The three rules listed above contain important elements in FA, which are variations of light intensity and attractiveness. The FA works by moving the individual firefly towards more attractive fireflies. The attractiveness of the fireflies depends on the intensity of light emitted. With regard to this algorithm, the light intensity is associated to the landscape of objective function. This light intensity is determined by the distance between each individual firefly. The more nearer the fireflies are from each other, the higher the light intensity; which would result in attracting more fireflies toward it. If the fireflies are unable to find brighter fireflies in the population, it is moved randomly across the search space.

2.5.2.2 Ant Colony Optimisation (ACO)

Another popular Swarm Intelligence inspired by nature is an algorithm called Ant Colony Optimisation (ACO) algorithm. The ACO takes inspiration from finding the optimal path during food foraging behaviour of ants. Initially, the first version of ACO known as Ant System was introduced by Dorigo et al. (1996) for combinatorial problems, specifically Travelling Salesman Problem (TSP). The algorithm works by imitating the food searching behaviour in the ant colony. The food finding process starts by sending the ants randomly surrounding the nest. Once the food source is found, the ants return to the nest. During the journey back to the nest, the ants deposits pheromones along the path between food source and nest. With more ants using that path, it indicates that it is a favourable path that will result in more pheromones deposited on that path. Consequently, this path would attract more ants as high intensity of pheromones are present.

Furthermore, as mentioned earlier the ACO was initially used to solve combinatorial problems. Thus, in order to implement the ACO algorithm for continuous optimisation problems, Socha and Dorigo (2008) presented an extended version of the ACO algorithm. In this extended version of the ACO algorithm, a probability density function was utilised to make the algorithm adapt to continuous domain variables. Besides this extended version of ACO algorithm, several other variants were also proposed to improve the original version such as the rank-based Ant System, Ant Colony System (ACS), MAX-MIN Ant System (MMAS) and elitist strategy Ant System (Dorigo and Stutzle, 2010).

2.5.2.3 Particle Swarm Optimisation (PSO)

In 1995, Kennedy and Eberhart developed a novel optimisation algorithm known as Particle Swarm Optimisation (PSO) algorithm. This algorithm is inspired by the flocking of birds in nature to solve optimisation problems. In the PSO algorithm, the set of candidate solutions is represented by a swarm of particles which move around the search space. The movement of these particles around the search space are according to the variation of velocity based on individual particle's previous best position and other particle's best position. This mechanism allows particles of PSO to move towards a better position by relying on the information of individual particles and other particles.

Since it was introduced in 1995, few studies have been done in improving the PSO algorithms. One of the improvements is the introduction of a clamping scheme limit to the velocity avoiding the particle from flying out of the search space (Marini and Walczak, 2015). Other studies added inertia weight on particle's updated equation to keep balance between exploration and exploitation which, overcome premature convergence of the PSO algorithm (Shi and Eberhart, 1998). In terms of application, the PSO algorithm has also been successfully used in a wide range of optimisation problems like dynamic, multi-objective and discrete (Boussaïd, et al., 2013).

Although many modifications claimed to be done on the standard PSO algorithm but the term standard PSO algorithm is defined differently among these studies. Therefore, it would be beneficial to establish a common standard for the PSO algorithm that consists of recent improvements over the original PSO algorithm. So far, there are three versions of standard PSO algorithm that have been defined but the most recent is the Standard Particle Swarm Optimisation 2011(SPSO2011). This SPSO2011 includes the latest theoretical development of PSO, which are adaptive random topology and rotational invariance (Zambrano-Bigiarini et al., 2013).

2.5.2.4 Honey Bees Inspired Algorithm

The Honey Bees Inspired algorithm is another type of population-based metaheuristic based on SI. This type of SI-based algorithms is inspired by the social behaviour of honey bees in nature and has attracted a lot of attention recently in the field of optimisation algorithms. Currently, there are four main behaviours of honey bees being developed for optimisation algorithms which are nest site selection, queen bee evolution process, mating and breeding behaviour, and foraging behaviour (Seeley and Visscher, 2004; Sung, 2003; Haddad et al., 2006; Seeley, 1995). Despite having many different versions of Honey Bee Inspired algorithms available, the most recognised and well known algorithms are algorithms based on the foraging behaviour compared to other bee-based algorithms as a considerable amount of literature has been published on this type of algorithm. This section briefly describes some main honey bee-based algorithms according to their behaviour.

One of the main activities in honey bees is selecting a potential nest or hive. In general, the process of nest selection starts by sending scout bees to the surrounding environment. Then, the scout bees return to the hive to advertise their findings of suitable nest sites. The scout bees communicate with other bees at the hive by performing a dance called the “waggle dance”. Finally, the bees come out with a decision on the site to be selected. Before making the decision in selecting a suitable home for the bee colony, there are several requirements that need to be considered. There are three of them: accurate decision, quick decision, and mutual decision (Seeley and Visscher, 2004). An accurate decision is required to ensure the hive has adequate space to accommodate the bee colony and provide secure protection from predators or rough weather. Meanwhile, quick decision is needed to reduce the time scout bees spend outside the hive because the longer they are outside the hive, the more likely they are being exposed to danger and the energy reserves are reduced as well. The decision also needs to be agreed by all colony members. A non-mutual decision by the bee colony would cause the colony to be divided leading to a non-fully functioning colony, as mostly a colony only has one queen. Thus, taking into consideration all these requirements is essential in the decision-making process of selecting the best quality nest, which is used in the context of optimisation.

By using the model in the decision-making process of honey bees, Diwold et al. (2010) proposed a technique to solve dynamic and noisy optimisation problems. The result of the proposed approach is promising in making decisions for both types of environment problems. In addition, this study also makes use of the behaviour of honey bees called *Apis Florea* (Asian Dwarf honey bees) instead of the usual species known as *Apis Mellifira* (European honey bees), where it is possible to relocate after moving to a new nest if the nest site is not the best one. The proposed technique used iterative nest selections and mimics the relocation behaviour to eliminate multiple potential nest sites for continuous function optimisations, specifically the *Sphere* and *Booth* benchmark functions.

The Queen bee evolution process is another type of bee behaviour in nature that has been utilised for optimisation algorithms. It was initially introduced by Sung (2003) to improve the capability of Genetic Algorithm (GA). In this study, the best solution of GA for each generation corresponds to the fittest bee (Queen bee) in the solution. Then, this fittest bee (Queen bee) produces new solutions using crossover operator by mating it with other bees selected as parents. Although utilising this behaviour in GA improves the exploitation capability of GA, but it also causes premature convergence. In order to overcome this problem, several individual solutions are permitted to mutate regularly, resulting in more balance between exploration and exploitation.

Apart from the study mentioned above, Wang (2009) had also presented a hybrid Bee Evolutionary Genetic Algorithm (BEGA) with a clustering method to solve aircraft sequencing problems. This proposed algorithm is also a type of algorithm based on the Queen bee evolution process. Similarly, like earlier proposed algorithms, the BEGA experiences premature convergence. An alternative approach was used to overcome this problem; by introducing a random population in each iteration. Subsequent to introduction of BEGA, Ming et al. (2010) proposed an improved version of BEGA. This improved version implemented an adaptive selection operator to determine the size of the random population rather than using a fixed size.

Other behaviours of honey bees that have been modelled in the context of optimisation are mating and breeding behaviours. These behaviours were first adopted by Abbass (2001) in an algorithm known as Marriage in Honey Bees Optimisation (MBO) algorithm. The algorithm imitates the evolution of honey bees at a solitary colony (single queen) up until the establishment of eusocial colonies (one or more queens). In order to produce a family, the queen should mate with the drones probabilistically during the mating flight. This flight starts after the queen made the dance, which would be followed by the drones as well to be mated with the queen. During the mating flight on air, the queen mates with drones until the sperm gathered in the *spermatheca* or the queen's energy level arrives at a certain threshold. Usually, the queen bee starts the mating flight with an amount of energy, and progressively decreases over time. Once the queen returns to the hive, the breeding process is done by choosing a random sperm from the *spermatheca*. The broods are produced via a crossover of selected sperm with the queen's genome followed by mutation on the broods. Furthermore, the workers enhance the produced broods and update their fitness after that process. The last stage of this algorithm is replacing the least fit queen with the fittest broods and killing the unselected broods, which would establish a new eusocial colony. The search process of this algorithm continues with another mating flight until the stopping criterion has been met.

Besides the MBO algorithm, Haddad et al. (2006) also presented an algorithm based on similar bee behaviour described earlier named the Honey Bee Mating Optimisation (HBMO) algorithm. Findings of this proposed algorithm in applications of water resources optimisation was promising. However, according to Yang et al. (2007), both algorithms mentioned earlier are slow in terms of computational time due to complex calculation procedures. Therefore, an algorithm called the Fast Marriage in Honey Bee Optimisation (FBMO) algorithm was introduced to overcome that issue (Yang et al., 2007). The proposed fast version of the MBO algorithm mates the randomly generated drones with a finite number of queens instead of mating probabilistically. This FMBO algorithm showed better convergence speed compared to the MBO algorithm and easy to implement as less number of parameters are required.

The foraging behaviour of honey bees is one of the activities in honey bees that have been regularly modelled in terms of optimisation. So far three algorithms have been considered to be the main algorithm categorised under this type of behaviour, which are the Bee Colony Optimisation (BCO) algorithm, Artificial Bee Colony (ABC) algorithm and the Bees Algorithm (Karaboga et al., 2012) .

The first version of algorithm inspired by the foraging of honey bees was proposed by Sato and Hagiwara (1997) named Bee System (BS). In general, this proposed BS is an improvement to the Genetic Algorithm (GA) using honey bees foraging behaviour, where chromosomes of better fitness are considered superior chromosomes and other chromosomes search surrounding the superior chromosomes using multiple solutions. In addition, two new operators called the concentrated crossover and Pseudo-Simplex Method were also introduced into this BS to provide balance between a global search and local search. Although this BS uses bee behaviour; the bees inform other bees by dancing once the feed has been found followed by working together to carry the feed to the hive, but it is still not a fully bee-inspired algorithm as it is considered to be an improved GA. Hence, an alternative of BS was introduced by Lucic and Teodorović (2002) to solve the travelling salesman problem. This version of BS consists of scout bees and forager bees but the scout bees have no guidance during food searching. Thus, this method involved low cost food and low quality food as the aim is to find any kind of food.

Later, Teodorovic and Dell'Orco (2005) presented an extended and generalised version of BS called the Bee Colony Optimisation (BCO) algorithm. There are two main stages involved in this proposed algorithm, which are the *forward pass* and *backward pass*. In the first stage, the bees explore the search space according to a predefined number of moves that would generate partial solutions whereas in the second stage, the returned bees at the hive inform other bees of the quality of solutions found by performing a dance. Based on the quality of solutions advertised, the bees make a decision whether to keep on dancing and recruit more bees for further exploitation or abandon the generated solutions and follow one of the dancer bees. The bees with high quality solutions are likely to be followed by other bees and keep on exploration near previously found solutions. These two processes of *forward pass* and *backward pass* continue iteratively until a predefined stopping criterion has been met.

Since the establishment of BCO, various improvement approaches have been proposed to the algorithm to solve different types of problems. One of the modifications done to BCO is utilising approximate reasoning and fuzzy logic into the bee's communications and actions (Teodorović et al., 2006). The proposed algorithm has been proposed for these combinatorial problems; Routing and Wavelength Assignment in all networks, Travelling Salesman Problem (TSP), and Ride Matching Problem. Furthermore, other improved versions of the BCO algorithm was proposed by Forsati et al. (2015) for document clustering applications. Two new concepts called *cloning* and *fairness* were introduced in this proposed algorithm to increase the exploration capability and propagation of information. Another recent improvement of the BCO algorithm is *weighted* BCO (*w*-BCO) algorithm, where global and local weights are considered, allowing the bees to search purposely (Moayedikia et al., 2015). In addition, the proposed algorithm also adopted a new recruiter selection method to conserve the population diversity.

Another type of honey bee inspired algorithm that has received great attention is the Artificial Bee Colony (ABC) algorithm. The algorithm was originally introduced by Karaboga (2005) to solve unconstrained numerical benchmark functions. In the basic version of the ABC algorithm, there are three groups of bees involve in the searching process; employed bees, onlooker bees and scout bees. An employed bee is a bee that is sent to food sources (potential solutions) and would come back to the hive to recruit onlooker bees whereas an onlooker bee is bee that is placed at the food source found by employed bee according to nectar amounts (quality or fitness of the solutions). The scout bee is a bee that searches randomly across the search space if the food source has been exhausted or no further improvements are found after a predefined number of iteration. The main steps of the ABC algorithms start by sending a number of the employed bees randomly in the search space. After returning to the hive, the employed bees advertise the information regarding the food sources found by performing a dance called the “waggle dance”. Then, the onlooker bees select food sources advertised by the employed bees to continue exploitation. The selection of food sources is based on the roulette-wheel rule. Similarly, like other population-based algorithms, this process is repeated until the stopping criterion has been met.

Previously, there have been misconceptions between ABC algorithm and Bees Algorithm as both algorithms were inspired by the foraging behaviour of bees and share similar concepts (Mirsadeghi and Shariat Panahi, 2012; Biegler-könig, 2013; Durongdumrongchai et al., 2014). Therefore, it is important to highlight the main differences between these two algorithms (Tsai, 2014a). One of the main major components that differentiate them is the method of updating locations. The ABC algorithm updates its location based on the *d*th dimension only while the Bees Algorithm update locations based on all *D* dimensions. The ABC algorithm also uses the roulette wheel selection method which is different from the Bees Algorithm. Apart from that, the ABC algorithm also has the mechanism to self-update the employed bees own locations whereas the Bees Algorithm does not. Based on all these differences, it is apparent that both algorithms are two different algorithms.

The ABC algorithm was initially developed for solving continuous benchmark functions where the results were compared with the PSO algorithm, Genetic Algorithm, Ant Colony Optimisation Algorithm, and Differential Algorithm (Karaboga and Basturk, 2007). Since that introduction, there have been numerous studies on improvements and applications of the ABC algorithm published in literature. One of those studies is an extended version of the ABC algorithm by Akay and Karaboga (2012) in application of engineering design problems. In order to handle constraint problems; a constraint handling strategy was used during the selection step to deal with the constraints. Then, the ABC algorithm has also been extended for multi-objective problems (Hedayatzadeh et al., 2010; Akbari et al., 2012). Furthermore, Karaboga and Gorkemli (2012) had proposed the Quick ABC algorithm (qABC) to improve convergence speed compared to the original version. In this work, a new strategy of updating onlooker bees was introduced, which is more accurate to foraging behaviour in nature. A parameter called *neighbourhood radius* was added in this variant and the results obtained after tuning the new parameter showed faster convergence than the basic ABC algorithm.

Apart from the honey bees inspired algorithm described above, there are several other algorithms based on similar behaviour. The Virtual Bees Algorithm (VBA) is one of the algorithms associated with foraging behaviour (Yang, 2005). This algorithm uses parallel multiple bees that work independently to solve numerical benchmark functions. Then, there is a routing algorithm known as *BeeAdHoc* algorithm, which is developed for energy efficient routing in Mobile Ad Hoc Networks (MANETs). The results of this new routing algorithm obtained better or similar results compared to other algorithms with the least energy consumption.

2.6 The Bees Algorithm

The Bees Algorithm is a nature-inspired algorithm that imitates the food foraging behaviour of honey bees in nature. The food foraging behaviour of honey bees starts with employing part of the bee colony population to search for high quality food sources surrounding the hive. After collecting the nectar, the scout bees return to the hive. The scout bees that had found high quality food sources communicate with other bees by performing a dance known as the "waggle dance". This dance is performed in a specific area of the hive; it gives three points of important information related to the flower patches discovered by the scout bees. The points of information are the direction where it is located, its distance from the hive, and its quality rating. After completing the waggle dance, the dancer bees recruit bees from the hive to go to the visited flower patch. Higher quality of flower patch recruits more bees. This process of food foraging mechanism will continue for the recruited bees.

In the Bees Algorithm, the position of a food source corresponds to a possible solution to the optimisation problem and the nectar amount of each food source represents the quality (fitness function) of the associated solution. In general, the unconstrained optimisation problem that is going to be solved can be represented as a D-dimensional minimisation problem as follows:

$$\text{Min} f(X),\ X = [x_1, x_2, ..x_d.., x_D]$$

where $X = [x_1, x_2, ..x_d.., x_D]$ is the vector to be optimised and D is the number of parameters. At the initialisation stage, ***ns*** scout bees generate a randomly distributed initial population. Each initial solution is evaluated by the fitness function. Then, each solution is ranked according to fitness value. After that, the ***nb*** best sites are selected from the ***ns*** scout bees. These scout bees perform the waggle dance to recruit bees from the hive for local exploration. The selected ***nb*** best sites for local search consist of ***ne*** sites (top rated sites) and ***nb-ne*** sites (remaining best sites). At this stage, the best ***ne*** sites recruit more bees than the remaining best sites ***nb-ne***. The recruited bees for elite sites ***nre*** and best sites ***nrb*** are placed randomly across the patch size (***ngh***). Then, the fitness position of recruited bee is evaluated by the fitness function. The best recruited bees for each patch is selected to do the waggle dance upon returning to the hive.

In the next stage, the remaining ***ns-nb*** bees are sent randomly across the search space for global search. At the last stage, new population of bees is formed combining the remaining ***ns-nb*** bees and selected recruited bee from each ***nb*** best site. The stopping criterion for this algorithm can be set either by a predefined number of iterations or predefined fitness above the threshold value. The flow chart of the basic Bees Algorithm is shown in Figure 2.1 (Pham and Castellani, 2009).

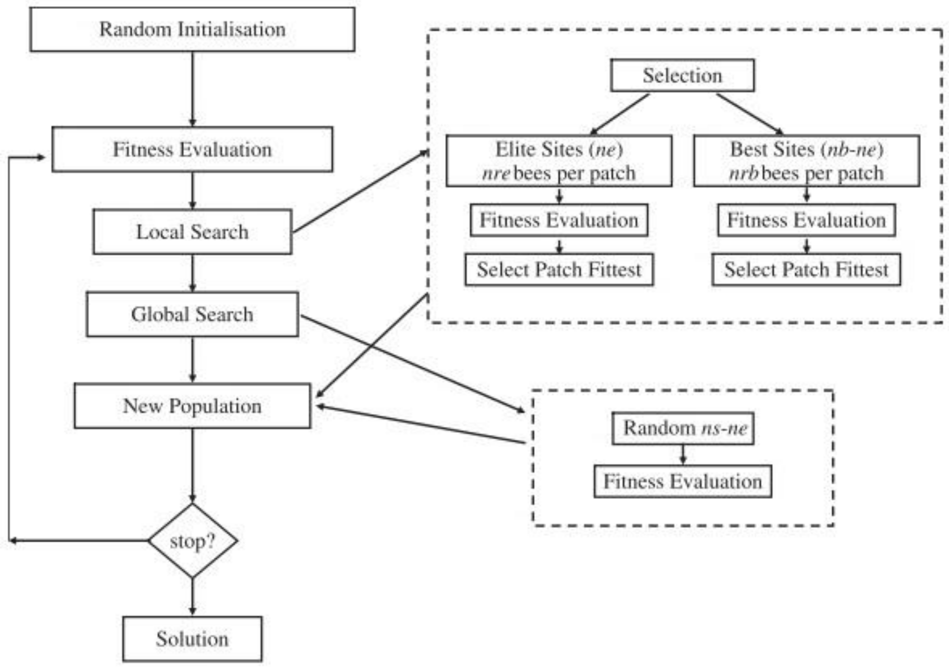

Figure 2.1: Flowchart of the basic Bees Algorithm

In addition to the basic Bees Algorithm, two new strategies were introduced later to improve the basic version of the Bees Algorithm. The first strategy is known as neighbourhood shrinking. In this strategy, the patch size is set to a large size. Then, as the search is in progress, the patch size shrinks to further refine the local search. This patch size remains constant if the fittest recruit bees find better fitness value than the scout bees. However, if the recruit bees failed to find a better fitness value, the neighbourhood size decreases. The neighbourhood shrinking procedure follows the following formula:

$$a_i(t) = ngh(t) * (max_i - min_i), \qquad (2.1)$$

$$ngh(t+1) = 0.8 * ngh(t), \qquad (2.2)$$

where t is the t^{th}-iteration of the Bees Algorithm. The second strategy is known as site abandonment. This strategy is used after a predefined t^{th} times (***stlim***) of neighbourhood shrinking failed to find any improvements. The position being abandoned is assumed to be the local peak of the optimisation problem. If the abandoned position is the best position found so far, it is considered as the global optimum or final solution.

2.7 Improvements

This section reviews improvements done on the Bees Algorithm. In general, the modifications done on the Bees Algorithm are organised into several parts, which are population initialisation, global search and local search, and parameter tuning or adaption. These modifications on some of those parts were done to improve the performance.

The first stage in the Bees Algorithm is known as population initialisation. The most common initialisation procedure is to send the scout bees randomly across the search space. Then, each scout bee evaluates the visited site according to the fitness function. So far, there is only one modification done on the initialisation procedure to improve the Bees Algorithm done by

Hussein et al. (2014). In this study, a novel initialisation algorithm based on the patch concept and Levy flight (movement pattern of biological organisms) distribution is proposed to initialise the population of bees in the Bees Algorithm. This version of Bees Algorithm is known as the Patch Levy Initialisation algorithm–Bees Algorithm (PLIA-BA), which mimics the natural flight patterns of bees by following the Levy flight distribution.

After population initialisation, the next stage of the Bees Algorithm is local search (exploitation) followed by global search (exploration). Up to now, a number of studies have been done on modifying local search or global search to further improve the Bees Algorithm. Among the earliest improvements on the local search was introduced by Pham and Castellani (2009). These improvements are neighbourhood shrinking and site abandonment strategies. Currently, the Bees Algorithm with these two strategies is known as the standard Bees Algorithm. Later, the standard Bees Algorithm was further tested on a set custom-made functions (Pham and Castellani, 2013) and real world problems; protein folding benchmarks (Pham and Castellani, 2015).

On the other hand, Packianather et al. (2009) proposed a new version of the Bees Algorithm based on pheromones to attract bees toward high promising patches. In this version of Bees Algorithm, the recruit bees are sent according to the level of pheromone of the selected sites instead of using a fixed value. This proposed recruitment mechanism eliminated the requirement to set ***nep*** and ***nsp*** parameter values of the Bees Algorithm.

Meanwhile, Pham and Haj Darwish (2010) used a recursive estimator that predicted the optimal parameters of the linear and nonlinear system known as the Kalman filter, in another enhanced version of the Bees Algorithm. The Kalman filter is used to update the position of recruited bees in the local search. In addition, this proposed Bees Algorithm also applied fuzzy greedy selection mechanism as a new method of choosing the best sites and determining the number of recruit bees.

Muhamad et al. (2011) incorporated local search manoeuvres into factor recruitment in the Bees Algorithm. The employment of local search manoeuvres is aimed to overcome the possibilities of recruited bees getting lost during flying towards selected patches. Thus, this added strategy into the Bees Algorithm enables the neighbourhood size to extend in certain directions. Despite this proposed Bees Algorithm showing faster convergence on numerical problems, it requires more parameters compared to the standard Bees Algorithm.

Another improved Bees Algorithm was proposed by Pham et al. (2012) where a new type of bee known as ‘young bees’ are introduced. These ‘young bees’ are the unselected bees in the population that are going to be replaced by the new scout bees. Instead of immediately eliminating these bees from the population, the ‘young bees’ are protected by letting them compete among each other for several iterations until they reached the adult stage. Subsequently, Castellani et al. (2012) applied this modified Bees Algorithm to dynamic optimisation in chemical engineering problems.

Afterward, Shatnawi et al. (2013a) introduced another improved version of Bees Algorithm called Memory-based Bees Algorithm (MBA) by adding local memory and global memory into the global search and local search. Three types of Memory-based Bees Algorithm are presented in this study, which are local memory-based Bees Algorithm, global-memory based Bees Algorithm and combination of both memory based Bees Algorithms. The addition of local memory was to prevent the recruit bees from visiting sites that have been visited. Meanwhile, the global memory was introduced to prevent scout bees from going to sites that have been visited or sites currently being visited. The global memory is also used to check whether the recruit bees should follow the best bee in the patch or position memorised by the global memory.

Another modification on local search of the Bees Algorithm involved neighbourhood size. One of the modifications was proposed by Yuce et al. (2013), in which it added extra enhancement on the two strategies introduced in the standard Bees Algorithm. This approach does not immediately abandon the unimproved sites after predetermined neighbourhood shrinking but goes through further enhancements before being abandoned. Following the introduction of this version of Bees Algorithm, the enhanced Bees Algorithm was also applied in multi-objective supply chain optimisation problem (Yuce et al., 2014).

Other modifications on the Bees Algorithm related to neighbourhood size is presented by Ahmad et al. (2012) using an asymmetrical neighbourhood search in the Bees Algorithm instead of a symmetrical search. Then, Ahmad et al. (2014) used a combination of adaptive enlargement and reduction in the neighbourhood search. It has been demonstrated in those studies that using the asymmetrical neighbourhood search has no significant effects on the Bees Algorithm while combinations of adaptive enlargements and reduction strategies proved to be helpful in solving mechanical design problems.

In another study, Tsai (2013) proposed a hybrid version of the Bees Algorithm and ABC algorithm. The results of this hybrid Bees Algorithm on unconstrained numerical functions highlighted the advantages of hybridising these two algorithms. A year later, Tsai (2014a) tested the ability of the hybrid Algorithm in handling constraint problems as well. Following this application on constraint problems, the hybrid Bees Algorithm was suggested as an alternative to solve constraint problems instead of Bees Algorithm or ABC algorithm due to its good performance. In addition Tsai (2014b) also proposed an enhanced version of Bees

Algorithm that uses a stochastic self-adaptive neighbourhood (*ssngh*) search. For this modified version of Bees Algorithm, the neighbourhood size is not set by the user but varied according to the position between two elite bees.

One of the most recent studies done on modification of the local search is proposed by Zhou, et al. (2015). In this proposed algorithm, the Bees Algorithm was modified to allow it finding multiple optima solutions in multimodal optimisation problems. The modified Bees Algorithm used dynamic colony sizes, radius estimations and Hill-Valley mechanism to ensure each patch does not converge toward similar optima solutions. In addition, a local search method called balanced search technique was also included in the Bees Algorithm to speed up the algorithm.

Another recent modification of the local search of the Bees Algorithm was proposed by Yuce et al. (2015). This version of Bees Algorithm uses a slope angle computation and Hill-Climbing Algorithm in the local search stage of the Bees Algorithm. The proposed Algorithm was tested on numerical benchmark functions and single machine scheduling problems.

So far, most of the studies have tended to focused on local search or global search rather than parameter adaption or tuning. Thus, making the parameters adaptive or reduced would be beneficial for the users. Currently, there is no systematic way to set the parameters of the Bees Algorithm. The most popular method is by fine tuning the parameters until the best solution is found which is time consuming. For this reason, few studies have been attempted to overcome this problem. One of the studies was proposed by Maneechote and Luangpaiboon (2010) that used the Design of Experiment and Modified Simplex Method (MSM) in finding the parameters. In another work done by Pham et al. (2009a), where fuzzy logic was utilised to improve the Bees Algorithm by eliminating some of those parameters.

2.8 Applications

Up to now, a number of studies have been reported on the successfulness of the Bees Algorithm application on a wide range of optimisation problems. Although the Bees Algorithm was initially introduced to solve continuous numerical functions (Pham et al., 2006a; Pham and Koç, 2010), the Bees Algorithm was also applied on discrete and combinatorial problems later. One of the popular applications of the Bees Algorithm is in the area of industrial engineering. Pham et al. (2007a) employed the Bees Algorithm to solve cell formation problems. Pham et al. (2007b) and Packianather et al. (2014) also used the Bees Algorithm to schedule jobs for a machine. Meanwhile, the Bees Algorithm also has been used in the planning of material handling equipment (Sayarshad, 2009), generalised assignment problems (Özbaki et al., 2010), PCB assembly optimisations (Ang et al., 2010), and assembly line balancing problems (Akpinar and Baykasoğlu, 2014a, 2014b; Daoud et al., 2012; Tapkan et al., 2011). Other works that used the Bees Algorithm in the industrial engineering field are optimisation of multi-objective supply chain networks (Mastrocinque et al., 2013; Yuce et al., 2014), container loading problems (Dereli and Das, 2011), manufacturing networks (Xu et al., 2012, 2015), combinatorial circuit designs (Mollabakhshi and Eshghi, 2013) and tower crane layout (Lien and Cheng, 2014).

The Bees Algorithm has also been utilised by some researchers to solve optimisation problems in the field of mechanical engineering. Pham et al. (2009b) and Mirzakhani et al. (2011) applied the Bees Algorithm on mechanical design problems while Parsa et al. (2013) implemented the Bees Algorithm to find optimal design of probe used in Eddy current testing. Zarea et al. (2013) applied the Bees Algorithm to find optimum design of plate fin heat exchangers. Long and Nhan (2012) used the Bees Algorithm in designing a hybrid electric vehicle. Another application of the Bees Algorithm are in detection of cracks on beam structures (Moradi et al., 2011; Moradi and Kargozarfard, 2013), detection of faulty rolling bearing (Attaran et al., 2011; Attaran and Ghanbarzadeh, 2014), updating the structure in finite element models of piping models (Moradi et al., 2010) and determining the neutral stability curve in plane Poiseuille flow (Bahrainian and Ghanbarzadeh, 2013). Moradi et al. (2011) and Xu et al. (2011) also applied the Bees Algorithm to investigate the performance of a spring damper system of a full vehicle model and optimisation of fuel consumption in a semi-track air cushion vehicle, respectively. Meanwhile, Ang et al. (2013) described an approach to generate branded product concepts by combining the Bees Algorithm and shape grammar. In addition, Vejdannik and Sadr (2016) introduced the usage of Bees Algorithm in adopting the smoothing parameters of Probabilistic Neural Network (PNN) and Radial Basis Function (RBF) for automatic microstructure classifications.

A few other applications of the Bees Algorithm are in the areas of electrical engineering. Idris et al. (2009) applied the Bees Algorithm to find the optimal location and parameter settings of Flexible AC Transmission System (FACTS) devices. Afterwards, Idris et al. (2010) utilised the Bees Algorithm to determine the Available Transfer Capability (ATC) of power transactions between source and sink areas in deregulated power systems. Polratanasuk et al. (2010), Sumpavakup et al. (2012), and Anantasate and Bhasaputra (2011) proposed the use of the Bees Algorithm to solve optimal power flow problems. The Bees Algorithm in these studies used the parallel computing approach in combination with the Cultural Algorithm (CA) and crowded selection with fuzzy mechanism in the selection process. Leeprechanon and Polratanasak (2010) presented an application of the Bees Algorithm for Environmental or Economic Dispatch (EED) problems with clustering techniques. Other applications in electrical engineering are determining optimum design for a five-phase surface-mounted permanent magnet synchronous motor (Ilka et al., 2013), finding optimal solar farms and wind farms for power systems (Phonrattanasak et al., 2013; Phonrattanasak, 2011), and selecting optimal location of multiple distributed generation units in power distribution systems. Then, one of the latest applications of Bees Algorithm in electrical engineering is done by Gholipour et al. (2015) that applied it for tuning the back stepping parameters in the thermal plasma technology system.

In the area of electronic engineering applications, the Bees Algorithm was used to design the antenna of different characteristics (Guney and Onay, 2007, 2008, 2010, 2011). Meanwhile, Sayadi et al. (2009) utilised the Bees Algorithm for communication network applications. In addition, Boumazouza et al. (2013) described the usage of the Bees Algorithm in detection of objects in motion for video sequences.

The Bees Algorithm was also applied to overcome optimisation problems involving the area of control engineering. Among of the applications is the tuning of the proportional integral controller (PID) (Ercin and Coban, 2011; Jones and Bouffet, 2008). In these works, the Bees Algorithm is compared to other algorithms as well. Ang et al. (2009) used the Bees Algorithm in finding the minimum time for motion time planning of a robot arm whereas Pham and Kalyoncu (2009) and Zaeri et al. (2011) utilised the Bees Algorithm to tune fuzzy logic controllers for a flexible single link robot arm and pitch displacement of aircraft respectively. In contrast, Eldukhri and Kamil (2013) optimised the parameters to control swing up movements of a robot gymnast. In addition, Fahmy et al. (2011) implemented the Bees Algorithm to tune the parameters for two optimisation tasks related to robot manipulator control.

Another important area of application for the Bees Algorithm is in data mining especially related to data clustering. The most common approach in solving clustering problems is by using the Bees Algorithm along with other algorithms to make use of their benefits (Bonab et al., 2015; Pham et al., 2011; Shafia et al., 2011). Other applications in the area of data mining are extraction of fuzzy measures for sample data (Wang, et al., 2011) and control chart pattern recognition (Ebrahimzadeh et al., 2013). Furthermore, the Bees Algorithm was also used to optimise neural networks for wood defects identification such as described by Pham et al. (2006b), Pham and Darwish (2010) that included Bees Algorithm with Kalman Filter, and Packianather and Kapoor (2015) with introduction of a wrapper based feature concept.

Software engineering is another area of application for the Bees Algorithm. Azzeh (2011) proposed a novel method of software effort estimation using the Case Based Reasoning (CBR) method along with the Bees Algorithm. Wang et al. (2012) used the combination of Bees Algorithm with interval constraint solver for assessing the quality of the software. Hazli et al. (2013) also proposed a preliminary study of using the Bees Algorithm for software testing.

Some other applications are in the area of image processing (Ahmad Farhan and Bilal, 2011; Shatnawi et al., 2013b), civil engineering (Aydogdu and Akijn, 2011), chemistry (Pham et al., 2012) and computational biology (Bahamish et al., 2008).

2.9 Summary

This chapter briefly described the main algorithms associated with metaheuristic in optimisation. The review also included describing direct search methods which is a type that commonly used in the conventional approach. Furthermore, the chapter specifically focused on the Bees Algorithm, its improvements and applications of the Bees Algorithm in different areas.

CHAPTER 3

Bees Algorithm Enhanced with the Nelder and Mead Method

3.1 Preliminaries

The Bees Algorithm comprises of combinations of global exploration and local exploitation. In the global exploration, the Bees Algorithm searches randomly for new promising solutions across the search space. In contrast, local exploitation focuses on exploitation around selected promising solutions. These two mechanisms form an efficient swarm-based optimisation algorithm better than other state-of-the-art optimisation algorithms.

This chapter presents a modification of the local search in the standard Bees Algorithm. In the standard version of the Bees Algorithm, recruited bees are sent randomly across the selected patch without any directional information. This work proposed an improved Bees Algorithm with the Nelder and Mead (NM) method to guide the recruit bees during local search procedures. The selection of this method to improve the Bees Algorithm are based on suitability and easiness of implementation into the Bees Algorithm.

The rest of this chapter is organised as follows: In section 3.2, the NM method and the proposed algorithm are described in detail followed by the experimental design and result comparison. Section 3.3 shows application of the proposed algorithm on several constrained mechanical design problems. Finally, Section 3.4 concludes and reviews the work.

3.2 The Bees Algorithm with the Nelder and Mead (NM) Method

In this section, a modification to the standard Bees Algorithm is proposed to further enhance the capability of the Bees Algorithm. The modification was done in the local search stages of the Bees Algorithm. In the standard version of the Bees Algorithm, recruited bees are randomly placed across the neighbourhood of selected patch. However, in this proposed algorithm, a direct search method known as the Nelder and Mead (NM) method is integrated into local search to provide directional information between recruited bees.

The NM method is a free derivative-based local search method (no derivative value required) established by Nelder and Mead. This method finds the optimum value using simplexes formed by (N + 1) points in the N dimensional space. In two dimensions, it forms a triangle. It begins with a given worst position (**W**), best position (**B**) and good position (**G**) that correspond to a fitness value represent by (N + 1) points in the N dimensional spaces. In each generation of the algorithm, four types of operators are involved which are reflection, contraction, expansion, and shrinkage.

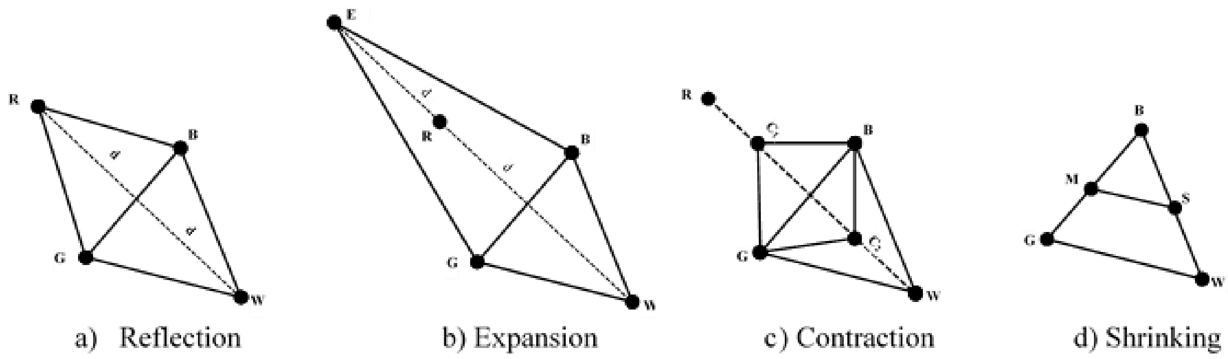

Figure 3.1: NM Operators

To begin this NM method, a simplex (e.g. two dimensional space minimisation problems) is formed from the worst position (**W**), best position (**B**) and good position (**G**). Then, a reflection position (**R**) is produced by reflecting position (**W)** in the direction of position **(B)** with *d* distance as described in Figure 3.1(a). After fitness evaluation of position (**R**), if fitness value of position (**R**) is less than fitness value at position (**G**), another operator known as expansion is performed (Figure 3.1 (b)). In this operator, position (**R**) is moved further towards position (**E**). If the fitness value of position (**E**) is less than fitness value of position (**R**), it is retained for the next operation. However, if the reflection operator failed to give a better fitness value, contraction operator is performed by contracting position (**R**) towards position $\mathbf{C_1}$ or $\mathbf{C_2}$ as shown in Figure 3.1 (c). Finally, if the contraction operator failed to gives a better fitness value, a smaller simplex is formed by replacing the good position with position **M** between position **B**–**G** and worst position with position **S** between **B-W**. This operator is described in Figure 3.1 (d). These operators are performed iteratively until solution is found.

The proposed Bees Algorithm known as the Nelder and Mead Bees Algorithm (NMBA) begins with initialisation followed by fitness evaluation. Then, high ranked sites are selected for local search. Once the elite sites and best sites have been selected, the recruited bees are sent randomly across the patch. In the standard version of the Bees Algorithm, the fittest recruited bee is selected from each site. However, in this proposed algorithm, the worst position ($\boldsymbol{x_w}$), good position ($\boldsymbol{x_g}$) and best position ($\boldsymbol{x_b}$) of the randomly recruited bees are used to form a simplex as in the NM method explained previously. The flowchart of the proposed algorithm is given in Figure 3.2.

Figure 3.3 illustrates implementation of NM method during local search where at least three recruited bees are sent to Patch_1 and Patch_2. After the fitness evaluation, those recruited bees are sorted according to fitness value. The worst position ($\boldsymbol{x_w}$), good position ($\boldsymbol{x_g}$) and best position ($\boldsymbol{x_b}$) of recruited bees are determined to begin the NM method. Then, a reflection position is generated using the expansion operator as described previously. If the reflection position gives better fitness value, an expansion operator is performed to generate position **E**. Position **E** is selected as the fittest for that patch if it gives a better fitness value than position R. However, if position **R** failed to give a better fitness value, a contraction operator is performed followed by a shrinking operator. At the end of the local search stage, the fittest position is selected out of the positions found by the NM operator and randomly recruited bees. After the local search, the remaining unselected scout bees are sent randomly across the search space for global exploration. Then, a new population is formed by merging the fittest from each patch (local search) and unselected scout bees (global search). This mechanism continues until stopping criterion is met. To avoid excessive computation, the NM method was applied only in elite sites instead of elite sites and best sites. The remaining phases of the NMBA are similar to the standard Bees Algorithm.

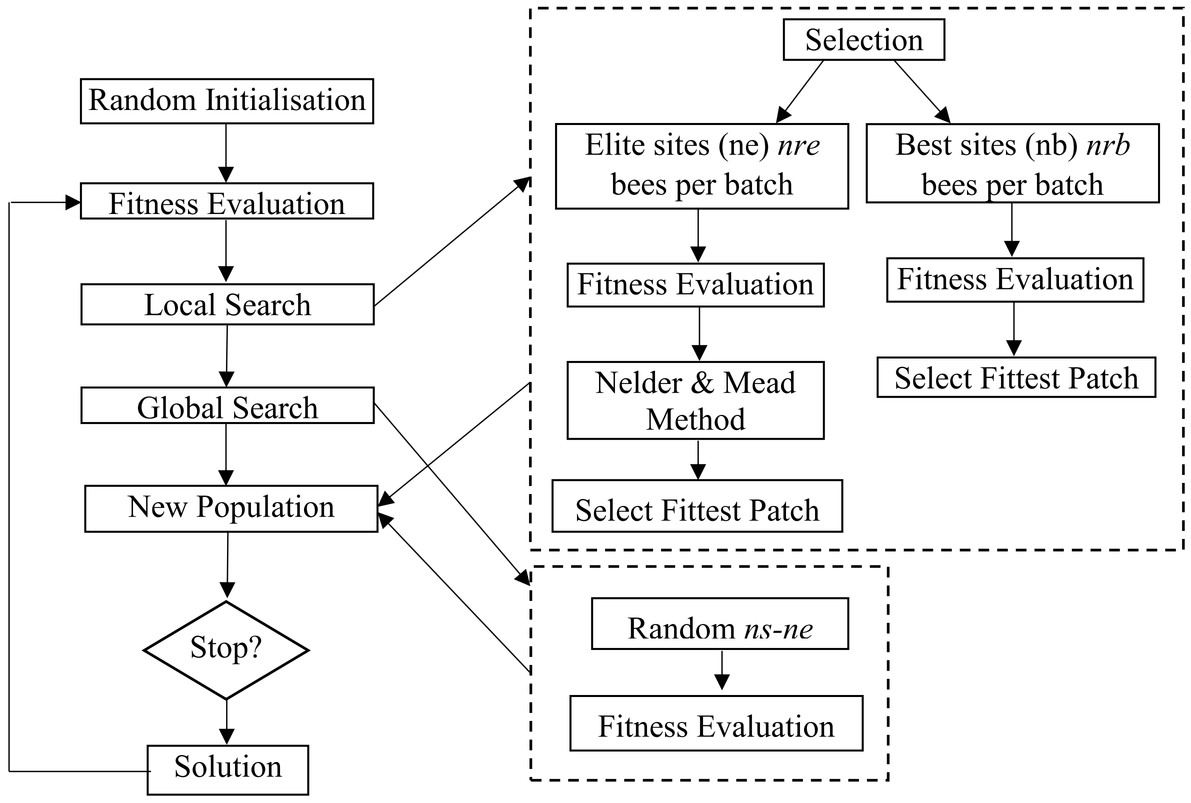

Figure 3.2: Flowchart of the Bees Algorithm with Nelder and Mead method

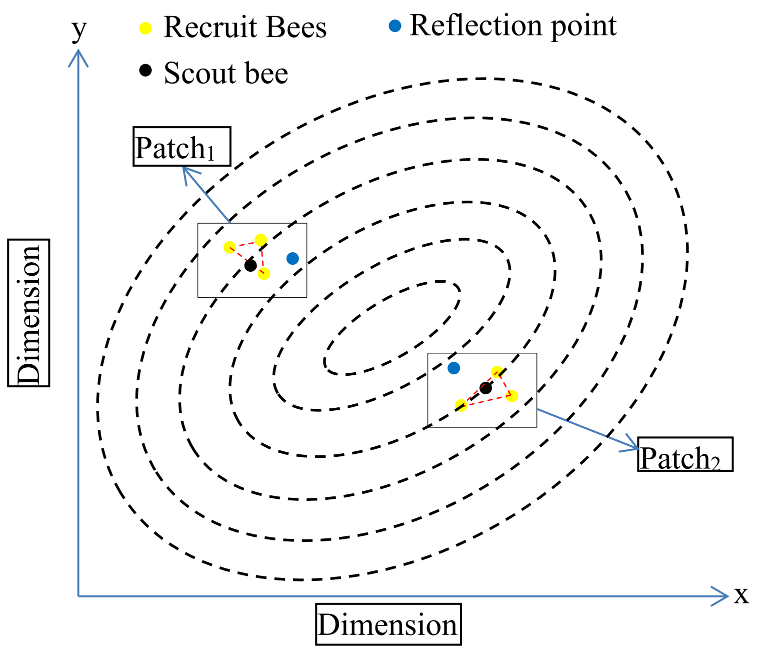

Figure 3.3: Two-dimensional illustration of NMBA implementation

3.2.1 Experimental Setup

In this chapter, the performance of NMBA was tested with a set of unconstrained benchmark functions, as shown in Appendix A. All the benchmark functions used in this chapter are minimisation problems. These fifteen functions are selected based on their popularity and characteristics of those benchmark functions. Jamil and Yang (2013) classified the five characteristics of the benchmark functions: modality, basins, valleys, separability and dimensions.

A function landscape with more than one peak is called multimodal function, whereas a function with one peak is known as a unimodal function. These characteristics correspond to the modality of the function. Normally, multimodal functions are more difficult to solve because the algorithm has the possibility to be trapped in one of those peaks while searching for the global optimum. Basin refers to a relatively steep decline surrounding a large area while valley occurs when a narrow area of little change is surrounded by regions of steep descent. For separability, generally separable functions are easier to solve compared to inseparable functions because each variable of a function is independent of the other variables. Then, dimensionality corresponds to number of parameters to be solved. In most cases, the difficulty of a function increases as the number of dimension increases. The corresponding characteristics of functions used are summarised in Appendix B.

The first five functions shown in Appendix B are functions with unimodal fitness landscapes. The *Martin & Gaddy* function is considered as an easy unimodal function. The *Easom* function is a unimodal function that has flat surfaces and small areas of global minimum. *Trid* function is a simple unimodal benchmark function while *Rosenbrock* and *Zakharov* are non-separable unimodal functions. Even though, *Rosenbrock* and *Zakharov* functions are unimodal functions, but the non-separable feature of both functions cause difficulty in finding the global minimum as each variable of a function is dependent on the other variables (Jamil and Yang, 2013). Moreover, the location of global minimum for the *Rosenbrock* function which lies in narrow, parabolic valleys causes difficulty to find the global minimum (Pham and Castellani, 2009).

The remaining ten functions shown in Appendix B are multimodal functions with a variety of features. The *Schaffer*, *Rastrigin*, *Schwefel* and *Griewank* functions are multimodal functions with a large number of local optima. These types of functions have a wavelike fitness landscape which could cause difficulties in finding the global minimum. In addition, the *Griewank* function is a non-separable function which has interdependence among the variables (Karaboga and Akay, 2009). Another feature that is included in this set of benchmark function is that the location of the global minimum is very small relative to the search space denoted by the *Michaelewicz* function (Jamil et al., 2013). Then, for both *Shekel_4D* and *Shekel_10D* and the *Langermann* functions, the location of the local minimums is randomly distributed over the search space and it is a non-symmetrical function. Thus, these functions do not give any advantages to the algorithms that take advantage of symmetrical features in benchmark functions (Karaboga and Akay, 2009). For *Camel Six Hump* and *Goldstein & Price* functions, both functions are considered as easy multimodal functions as most of the other algorithms are able to find the global minimum.

One of the important elements in this research is to determine parameter settings of the Bees Algorithm. The most popular adopted method is to finely tune the parameters for each function. This method requires large number of experiments by trial and error until the best parameters are found. However, previous study done by Pham and Castellani (2009) showed that using different parameters on the Bees Algorithm for each problem does not significantly improve its performance. Thus, a set of parameters was used in this research based on the work done by

Pham and Castellani (2009) for the standard Bees Algorithm and NMBA as shown in Table 3.1. The result of NMBA on benchmark functions mentioned above was compared with standard Bees Algorithm.

Table 3.1: Parameter setting values

Parameter		
ns	number of scout bees	24
ne	number of elite sites	2
nb	number of best sites	4
nre	recruited bees for elite sites	30
nrb	recruited bees for remaining best sites	10
ngh	initial size of neighbourhood	0.01
stlim	limit of stagnation cycles for site abandonment	10

In addition to comparison with the standard Bees Algorithm, the proposed Bees Algorithm was also compared with another two-popular swarm-based algorithms known as quick Artificial Bees Algorithm (qABC) and Standard Particle Swarm Optimisation 2011 (SPSO2011). The qABC was selected for comparison with NMBA because both algorithms were inspired by similar foraging behaviour. The qABC (Karaboga and Gorkemli, 2012) was selected out of other variants of ABC as it is considered one of the latest version of ABC. As for SPSO2011 (Zambrano-Bigiarini et al., 2013), it is among the popular swarm-based optimisation in the literature. For this reason, it is considered the most suitable for comparison with the proposed Bees Algorithm. The parameter setting values of both qABC and SPSO2011 are set according to the original paper of SPSO2011 (Zambrano-Bigiarini et al., 2013) and quick-ABC (Karaboga and Gorkemli, 2012). Despite using parameter settings from the original authors, the number of population size for both qABC and SPSO2011 are tailored as 100 populations for fair comparison. The SPSO2011 was run using the open source implementation provided by *pso R package v1.0-3* (Claus, 2012). All algorithms were run in the ***R*** statistical environment 3.2.0.

The stopping criteria for all algorithms were set according to solution found or number of function evaluations. In this study, the algorithm was stopped once acceptable solution has been found or it reached maximum number of evaluations. A solution is considered acceptable if the difference between the global minimum and solution found (accuracy) is less than or equal to 0.001. The maximum number of function evaluations for all algorithms was set to 500000 function evaluations. Moreover, each algorithm had 100 runs for each benchmark function.

3.2.2 Experimental Results

In this section, the results of the standard Bees Algorithm, NMBA, SPSO2011 and qABC on a set of benchmark function as described in the previous section are compared and presented in Table 3.2 and Table 3.3. The median and standard deviation accuracy obtained by each algorithm over 100 runs for each function are recorded in Table 3.2 while the median and standard deviation number of function evaluations (speed) to obtain the corresponding accuracy are recorded in Table 3.3. For the accuracy results in Table 3.2, if the accuracy obtained is less than 0.001, the accuracy recorded is 0.0000.

In order to perform the result comparison of NMBA between standard Bees Algorithm and other algorithms, a comparison method was adopted in this chapter and in the subsequent two chapters as well. In this method, one of those two algorithms are said to perform better than another if the accuracy is less than the other and the difference is statistically significant for each benchmark function. If the difference is statistically insignificant, the speed of both algorithms is considered for comparison instead.

Similar comparison methods are also used for speed. An algorithm is said to perform better than another algorithm if the speed less than the speed of the other algorithm, and the difference is statistically significant. If the difference is statistically insignificant, both algorithms are assumed to be comparable.

Statistical significant difference of the obtained results, were utilised using the Mann Whitney U-tests. The two algorithms are statistically compared at $\alpha = 0.05$ significance level for the various benchmarks. The significance of the difference between the NMBA and other algorithms are presented in Table 3.4 and Table 3.5. The *p*-value shown in both Tables determine the significant difference between two median results; *p*-value less than 0.05 indicates the difference is significant.

Table 3.2: **Comparison on accuracy over 100 runs for NMBA**

Function	SBA		NMBA		SPSO2011		qABC	
	Median	StdDev	Median	StdDev	Median	StdDev	Median	StdDev
Easom	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Trid	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.3016	0.3062
Rosenbrock	4.8919	1.0673	4.7206	1.1237	6.9007	9.4754	**0.0758**	**0.0702**
Zakharov	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0235	0.0867
Schaffer	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0010**
Rastrigin	12.9437	3.2667	11.9511	3.0196	6.4253	2.2037	**0.0000**	**0.0219**
Schwefel	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Michaelewicz	**0.0000**	**0.0133**	**0.0000**	**0.0111**	0.0411	0.0895	**0.0000**	**0.0000**
Goldstein & Price	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Martin & Gaddy	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Camel Six Hump	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Shekel_4D	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**2.5068**	**0.0000**	**0.0014**
Shekel_10D	0.0015	2.6435	**0.0000**	**2.2667**	8.7202	0.2236	8.7231	0.0036
Griewank	0.1091	0.0226	0.0952	0.0176	**0.0341**	**0.0271**	0.0800	0.0389
Langermann	1.2564	0.0989	**0.8469**	**0.0889**	0.9677	0.1363	1.1713	0.0916

Table 3.3: **Comparison on function evaluations over 100 runs for NMBA**

Function	SBA		NMBA		SPSO2011		qABC	
	Median	StdDev	Median	StdDev	Median	StdDev	Median	StdDev
Easom	3124	1408.923	2450	1170.489	4000	885.2301	**1050**	**6888.015**
Trid	13224	4255.14	**8776.5**	**3954.117**	9100	700.657	500000	0
Rosenbrock	**500000**	**0**	**500000**	**0**	**500000**	**117878**	**500000**	**0**
Zakharov	16374	1896.454	**12481**	**1183.614**	35100	4619.083	500000	0
Schaffer	5305.5	2869.263	4029.5	3071.699	**2800**	**1728.957**	43951	225352.2
Rastrigin	500000	0	500000	0	500000	0	**352561.5**	**97711.63**
Schwefel	2825	100.508	2500	1468.152	14100	30860.92	**850**	**12132.96**
Michaelewicz	163807.5	155525.4	142765.5	146994.1	500000	223339.3	**105399**	**88612.96**
Goldstein & Price	1824	729.9545	**1168.5**	**500.9846**	4000	501.387	2350	35168.9
Martin & Gaddy	1464	732.9105	**1174.5**	**617.4853**	2000	459.1725	15350	204091.6
Camel Six Hump	924	315.7083	**442**	**165.017**	3031	581.481	850	405.292
Shekel_4D	10628	8756.234	**10054**	**10975.58**	82500	230204.6	37850	65029.46
Shekel_10D	500000	168591.4	**437767**	**189641.6**	500000	0	500000	0
Griewank	**500000**	**0**	**500000**	**0**	**500000**	**105560.5**	**500000**	**33908.05**
Langermann	**500000**	**0**	**500000**	**0**	**500000**	**0**	**500000**	**0**

The results recorded in Table 3.2 show that NMBA succeeded in finding an accuracy less than 0.001 for most of the benchmark functions considered, except for *Rosenbrock, Rastrigin, Griewank* and *Langermann* functions whereas the standard Bees Algorithm succeeded in finding threshold minimum values for ten benchmark functions except for *Rosenbrock, Rastrigin, Shekel_10D, Griewank* and *Langermann* functions. Meanwhile, SPSO2011 and qABC failed in finding the minimum threshold values for *Rosenbrock*, *Rastrigin*, *Michaelewicz, Shekel_10D, Griewank*, *Langermann* and *Trid*, *Rosenbrock*, *Zakharov*, *Shekel_10D, Griewank*, *Langermann* functions, respectively.

In terms of number of evaluations, the NMBA outperformed the standard Bees Algorithm on most of the functions except for *Rosenbrock, Rastrigin, Griewank* and *Langermann* functions. These four functions failed to converge and reach a maximum number of function evaluations over 100 runs. Hence, their performance is similar in terms of speed whereas for the *Michaelewicz* and *Shekel_4D* functions, despite NMBA's median function evaluations being less than obtained by Standard Bees Algorithm, the Mann Whitney test showed that these results were not statistically significant as the *p*-value were more than 0.05. Thus, both algorithms are said to be comparable in terms of function evaluations for *Michaelewicz* and *Shekel_4D* functions.

Comparison of NMBA with other algorithms showed that NMBA converges faster for most of the benchmark functions as shown in Table 3.3. For comparison with SPSO2011, the NMBA is faster than SPSO2011 for most of the benchmark functions except for *Schaffer, Trid, Rosenbrock, Rastrigin, Griewank,* and *Langermann* functions. For the *Schaffer* function, the SPSO2011 required less function evaluations than NMBA to converge whereas for the *Trid* function, both algorithms are said to be comparable as statistical analysis shown in Table 3.5 demonstrated that the difference is not significant. As for the *Rosenbrock, Rastrigin, Griewank,* and *Langermann* functions, the performance of both algorithms gives comparable results because those functions reached maximum function evaluations.

In other cases, NMBA performed better than qABC for most of the benchmark functions except for *Easom, Rastrigin, Schwefel,* and *Michaelewicz* functions. However, the performance of both algorithms for *Rosenbrock, Griewank,* and *Langermann* functions is similar in terms of function evaluations.

Table 3.4: **Significant difference of NMBA's median results against standard Bees Algorithm**

	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000
Trid	ns	-	s	0.0000
Rosenbrock	ns	0.6527	ns	-
Zakharov	ns	-	s	0.0000
Schaffer	ns	-	s	0.0000
Rastrigin	ns	0.1676	ns	-
Schwefel	ns	-	s	0.0000
Michaelewicz	ns	-	ns	0.2187
Goldstein & Price	ns	-	s	0.0000
Martin & Gaddy	ns	-	s	0.0000
Camel Six Hump	ns	-	s	0.0000
Shekel_4D	ns	-	ns	0.3576
Shekel_10D	ns	0.0629	s	0.0404
Griewank	s	0.0000	ns	-
Langermann	s	0.0000	ns	-

s: statistically significant, ns: not significant
'-': No statistical difference test required

Table 3.5: **Significant difference of NMBA's median results against other algorithms**

	vs SPSO2011				vs qABC			
	accuracy	*p*-value	speed	*p*-value	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000	ns	-	s	0 .0000
Trid	ns	-	ns	0.2041	s	0 .0000	s	0 .0000
Rosenbrock	ns	0.0969	ns	-	s	0 .0000	ns	-
Zakharov	ns	-	s	0.0000	s	0 .0000	s	0 .0000
Schaffer	ns	-	s	0.0000	ns	-	s	0 .0000
Rastrigin	s	0.0000	ns	-	s	0 .0000	s	0 .0000
Schwefel	ns	-	s	0.0000	ns	-	s	0 .0000
Michaelewicz	s	0.0035	s	0.0001	ns	-	s	0 .0000
Goldstein & Price	ns	-	s	0.0000	ns	-	s	0 .0000
Martin & Gaddy	ns	-	s	0.0000	ns	-	s	0 .0000
Camel Six Hump	ns	-	s	0.0000	ns	-	s	0 .0000
Shekel_4D	ns	-	s	0.0000	ns	-	s	0 .0000
Shekel_10D	s	0.0000	s	0.0000	s	0.0000	s	0 .0000
Griewank	s	0.0000	ns	-	s	0.0037	ns	-
Langermann	s	0.0000	ns	-	s	0.0000	ns	-

s: statistically significant, ns: not significant
'-': No statistical difference test required

Another important finding was the overall performance results achieved by NMBA where it excelled in eight out of fifteen benchmark functions tested, followed by qABC (five out of fifteen), and SPSO2011 (three out of fifteen). A comparison of overall performance is summarised in Table 3.6 where the NMBA gives better performance compared to other algorithms in case of the *Trid, Zakharov*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump*, *Shekel _4D*, *Shekel_10D*, and *Langermann* functions.

Table 3.6: **Comparison of overall performance for NMBA**

Function	**SBA**			**NMBA**			**SPSO2011**			**qABC**		
	succ.	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**
Easom	100	X		100	X		100	X		100	X	X
Trid	100	X		100	X	X	100	X	X	0		
Rosenbrock	0			0			21			0	X	X
Zakharov	100	X		100	X	X	100	X		0		
Schaffer	100	X		100	X		100	X	X	68	X	
Rastrigin	0			0			0			88	X	X
Schwefel	100	X		100	X		100	X		100	X	X
Michaelewicz	88	X		92	X		39			100	X	X
Goldstein & Price	100	X		100	X	X	100	X		100	X	
Martin & Gaddy	100	X		100	X	X	100	X		79	X	
Camel Six Hump	100	X		100	X	X	100	X		100	X	
Shekel_4D	100	X	X	100	X	X	59	X		99	X	
Shekel_10D	41			54	X	X	0			0		
Griewank	0			0			6	X	X	2		
Langermann	0			0	X	X	2			0		
Total	1029		1	1046		8	927		3	836		5

succ: Successful Rate, acc: Accuracy, perf: Overall Performance

Regarding comparison of the overall performance with the standard Bees Algorithm, it can be summarised from Tables 3.2–3.3 that, NMBA performed better than the standard Bees Algorithm on eleven out of fifteen functions tested. These results demonstrated that the performance of NMBA on most of the functions are better than the standard Bees Algorithm especially for converged functions where less function evaluations required. Thus, these results further support the idea of providing guide direction to the neighbourhood search in improving the Bees Algorithm.

3.2.3 Discussion

The most obvious finding to emerge from the results reported in the previous section is the effectiveness of the NM method in providing direction during the neighbourhood search. The results of this experiment indicate that the NM method is capable of improving the Bees Algorithm, specifically in terms of the convergence speed. It is shown in the previous section that the NMBA performed better than the standard Bees Algorithm on four unimodal functions and seven multimodal functions.

For unimodal functions such as *Easom*, *Trid*, *Zakharov*, and *Martin & Gaddy*, the NMBA is expected to converge faster than the standard Bees Algorithm as the recruited bees would be directed toward a better position inside or outside of the patches. An example of improvement in terms of convergence speed is illustrated in the plot of convergence for the *Trid* function in Figure 3.4. This figure clearly indicates the superiority of NMBA over the standard Bees Algorithm during the search progress, where better fitness has been found even at the early stages of the search. For multimodal functions such as *Schaffer*, *Schwefel*, *Goldstein & Price*, *Camel Six Hump* and *Shekel_10D*, despite its multimodality, the NMBA still found the acceptable solution at a higher convergence speed. Meanwhile for *Griewank* and *Langermann* functions, despite the NMBA not being capable to find an acceptable solution, but it is nearer to the global optima. Figure 3.5 illustrates an example of a convergence plot for the *Griewank* function at the logarithmic scale. The figure indicates that both algorithms stagnated in most of the search progress, but the NMBA still managed to find a better fitness value.

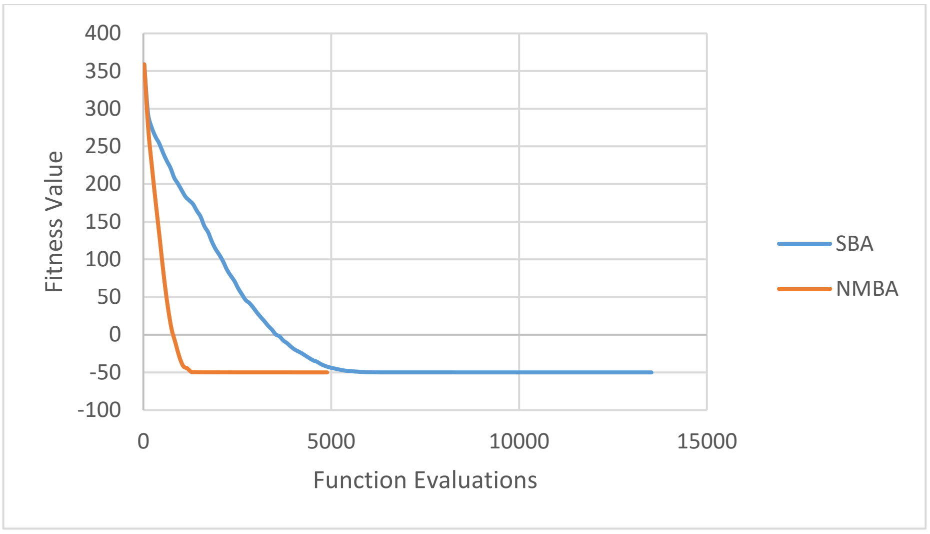

Figure 3.4: Plot of convergence for the *Trid* function

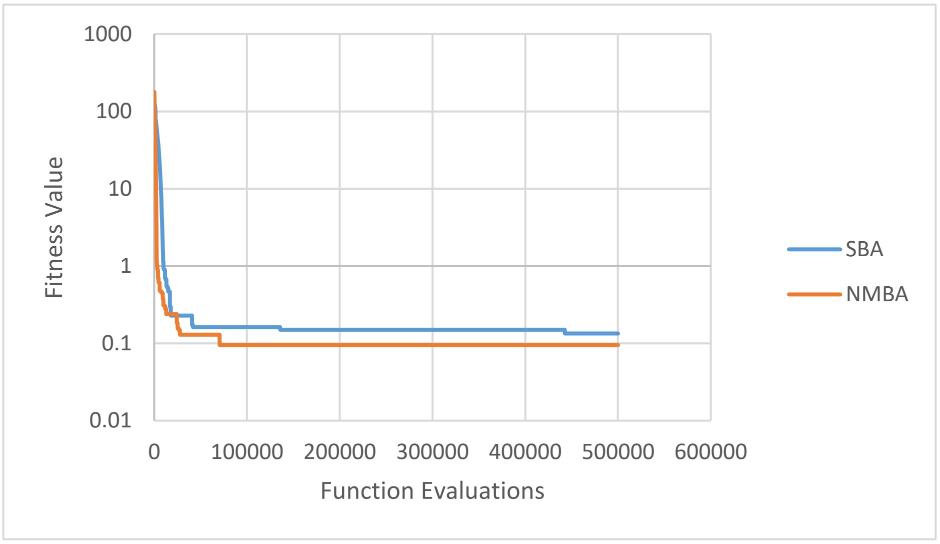

Figure 3.5: Plot of convergence for the *Griewank* function

With respect to equality in the performance of the NMBA with the standard Bees Algorithm for some of the functions (*Michaelewicz*, *Rastrigin*, *Rosenbrock,* and *Shekel_4D* functions), these results are likely to be related to the characteristic of these functions. For the *Michaelewicz* function, despite it being apparent that the median function's evaluations of the improved Bees Algorithm are less than the standard Bees Algorithm, the *p*-value of 0.2187 indicates that the difference is not significant. This result may be explained by the fact that the

Michaelewicz function is a multimodal function with valleys and ridges. For this reason, the NM strategy was less effective in dealing with this type of surface landscape.

Meanwhile, the *Rastrigin* is a function that has unimodal behaviour overall, with a rough multi-modal surface. Thus, the NM operators within the NMBA are unable to aid the recruited bees to locate a better fitness value as the they can be easily trapped between many local optimums.

For the *Rosenbrock* function, even though it is a unimodal function, the location of the global optimum inside a long, narrow, parabolic shaped flat valley causes difficulty in terms of finding global optimum. This might have contributed to the equal results in the performance on the *Rosenbrock* function.

As for the *Shekel_4D* function, the performance was found to be comparable to the standard Bees Algorithm, though at a higher dimension version of this function (*Shekel_10D*), the NMBA showed significant improvement. The reason for this is not clear but it may have something to do with the dimensionality of the problem as for some cases, a higher dimension is easier to solve.

When compared to other algorithms, the NMBA did not perform as well as the SPSO2011 and qABC for the *Easom*, *Rosenbrock*, *Schaffer*, *Rastrigin*, *Schwefel*, *Michaelewicz* and *Griewank* functions. For the *Schaffer* and *Griewank* functions, the SPSO2011 achieved the best overall performance compared to other algorithms with both functions sharing similar characteristics, namely a wavelike surface landscape. This characteristic might provide an advantage to the SPSO2011 over other algorithms in order to deal with this type of function.

As for the *Rastrigin* function, despite its similarities in terms of the surface landscape with the functions mentioned earlier, but its nature as a separable function causes the qABC to perform better than other algorithms. For the remaining functions, where the qABC attained the best overall performance, those functions are separable functions except for the *Rosenbrock* function. It is likely that the qABC gives the best performance for this type of function. Despite it being a non-separable function, the location of global optimum inside the deep valley and the flat surface landscape allows the qABC performed to be best performed on this function as well. The factors that contribute to this best performance on the *Rosenbroc*k function are likely due to long stagnation limit and single index dimension position update of the qABC.

3.3 Mechanical Design Applications

In order to further evaluate the effectiveness of the proposed algorithm, four well known constrained engineering design problems were chosen: Welded Beam, Pressure Vessel, Tension/Compression Spring and Speed Reducer (Akay and Karaboga, 2012). The objective of the Welded Beam and Pressure Vessel problems are to minimise cost subject to constraints whereas the objective of the Tension or Compression Spring and Speed Reducer are to minimise weight subject to constraints. These engineering design problems have different characteristics of objective functions with linear and nonlinear constraints (Rao et al., 2011). The details of these problems are given in Appendix C (Akay and Karaboga, 2012).

In the case of handling constrained problems, a method used by Ahmad (2012) was implemented in this study as the standard Bees Algorithm and NMBA are previously applied to unconstrained problems. In this method, only a feasible solution is allowed to be included in the solution list by checking whether all constraints have been satisfied. Any infeasible solution found during searching was abandoned from the solution list. This searching process continued until required feasible solutions have been found.

The constrained engineering design problems mentioned above have been attempted by several researchers on other algorithms. Thus, the results of NMBA and standard Bees Algorithm were compared against results obtained by other algorithms (Akay and Karaboga, 2012). The parameter values used for NMBA and standard Bees Algorithm for these problems are shown in Table 3.7. In order to make fair comparison, the stopping criterion for NMBA and standard Bees Algorithm were set as 30000 function evaluations. For each constrained engineering design problem, the best, worst, mean and median solution found over 30 runs were recorded as shown in Table 3.8. In general, the average (median) solutions found by NMBA and standard Bees Algorithm are comparable or better than other algorithms.

Table 3.7: **Parameter setting for mechanical design applications**

Problem	***ns***	***ne***	***nb***	***nre***	***nrb***	***ngh***	***stlim***
Welded Beam	10	2	5	10	5	0.08	5
Pressure Vessel	10	2	5	10	5	0.2	5
Tension/Compression Spring	10	2	5	10	5	0.001	5
Speed Reducer	10	2	5	10	5	0.01	5

Table 3.8: **Comparison of NMBA against others** (Akay and Karaboga , 2012)

Problem	Stats.	Sca[a]	Pso[a]	$(\mu+\lambda)$-ES	UPSOm	ABC	BA	NMBA
Welded Beam	Best	NA	NA	1.724852	1.92199	1.724852	1.7332472	1.73176
	Mean	NA	NA	1.777692	2.83721	1.741913	1.7679225	1.7649022
	Median	NA	NA	NA	NA	NA	1.7669785	1.764661
	NFE	NA	NA	30,000	100,000	30,000	30000	30000
Pressure Vessel	Best	6171	6059.7143	6059.70161	6544.27	6059.71474	6103.4175	6067.8964
	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.3081	6381.4714	6377.1937
	Median	NA	NA	NA	NA	NA	6315.1676	6241.594
	NFE	20000	30000	30000	100000	30000	30000	30000
Tension/ Compression Spring	Best	0.012669	0.012665	0.012689	0.01312	0.012665	0.012673	0.0126693
	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	0.0133388	0.0132001
	Median	NA	NA	NA	NA	NA	0.0133214	0.013165
	NFE	25,167	15000	30000	100000	30000	30000	30000
Speed Reducer	Best	2994.74	NA	2996.348094	NA	2997.05841	2999.5471	2997.2193
	Mean	3001.758	NA	2996.34809	NA	2997.05841	3006.3114	3001.9118
	Median	NA	NA	NA	NA	NA	3005.7187	3002.624
	NFE	54,456	NA	30000	NA	30000	30000	30000

Bold face indicates the winner of the algorithms, NA: not available, NFE:Number of function evaluations
[a]The welded beam problems are different from the one employed in this study

To compare the performance of NMBA against the standard Bees Algorithm, the significant differences between both of those algorithms were done using the Mann Whitney Test at $\alpha = 0.05$ significance level for each engineering design problem. If the median solution over number of runs found by NMBA is less than standard Bees Algorithm and the difference is significant, NMBA is said to perform better than standard Bees Algorithm. However, if the difference between those two algorithms is insignificant, then it is said to be comparable. The results in Table 3.8 show that NMBA obtained better median solutions than the standard Bees Algorithm for all the problems. However, a Mann Whitney test revealed that there were no significant differences between the solutions obtained by NMBA and standard Bees Algorithm for all the problems except for the speed reducer problem. The variation of minimum fitness value obtained by standard Bees Algorithm and NMBA over 30 runs are given in Figures 3.6–3.9. These figures validate the significant differences between both algorithms. These results suggest that the NMBA performed better than the standard Bees Algorithm for the speed reducer problem but equal in performance for the remaining three problems.

As for the comparison with other algorithms, the results of the best solution found by other algorithms in the literature are compared instead of the median result. It can be observed from Table 3.8 that, (μ+λ)-E and ABC produced the best solution for the Welded Beam problem whereas Pso and ABC produce the best solution for the Tension/Compression Spring problem. As for the Pressure Vessel and Speed Reducer problems, the best solutions were found by (μ+λ)-ES and Sca respectively. Even though these results show that other algorithms managed to find the best solution for all problems, it does not indicate superiority of other algorithms against NMBA. Further statistical analysis is still required to compare the performance. The most important aspect in comparison with other algorithms is to show the results obtained by the NMBA are generally comparable as they are all slightly different.

In summary, these results show that incorporating the NM method into local search of the standard Bees Algorithm could improve the capability of the standard Bee Algorithm to find better solutions. In addition to the result in Table 3.8, the best solution and constraint values found by all algorithms are summarised in Tables 3.9-3.12.

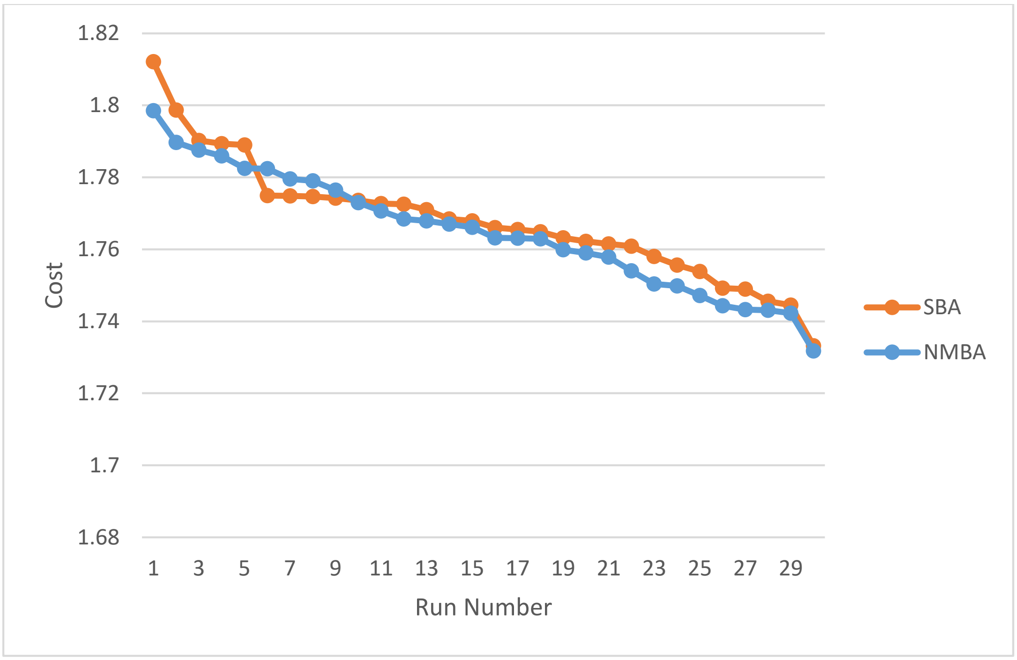

Figure 3.6: Minimum costs found by Standard Bees Algorithm and NMBA over 30 runs for the Welded Beam problem

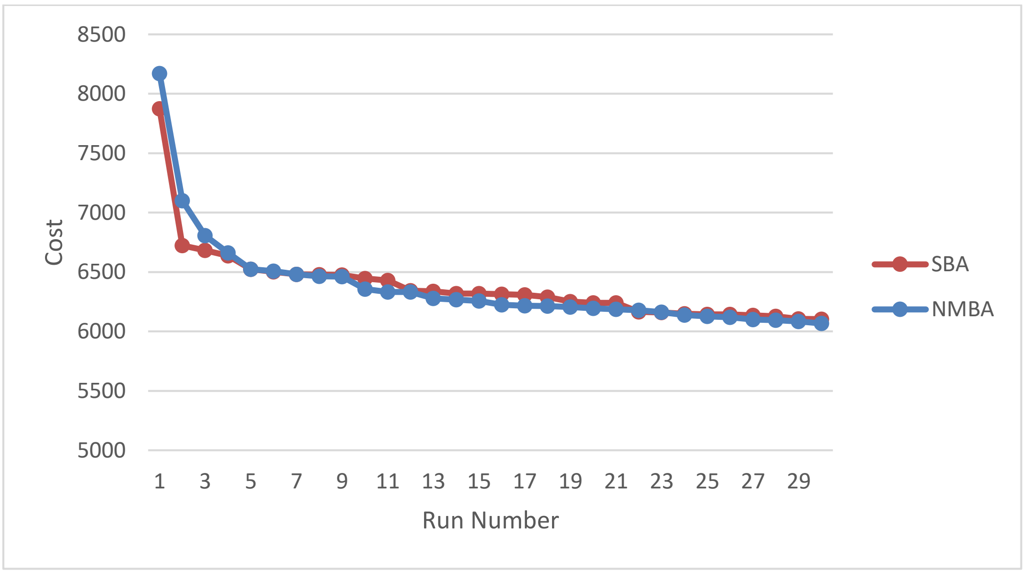

Figure 3.7: Minimum costs found by Standard Bees Algorithm and NMBA over 30 runs for the Pressure Vessel problem

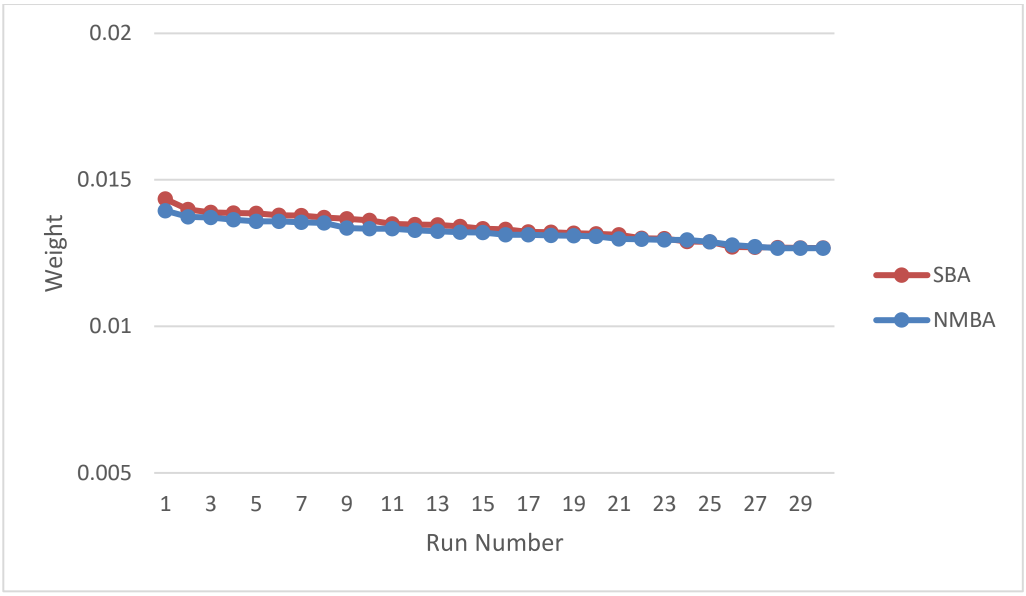

Figure 3.8: Minimum weights found by Standard Bees Algorithm and NMBA over 30 runs for the Tension/Compression Spring problem

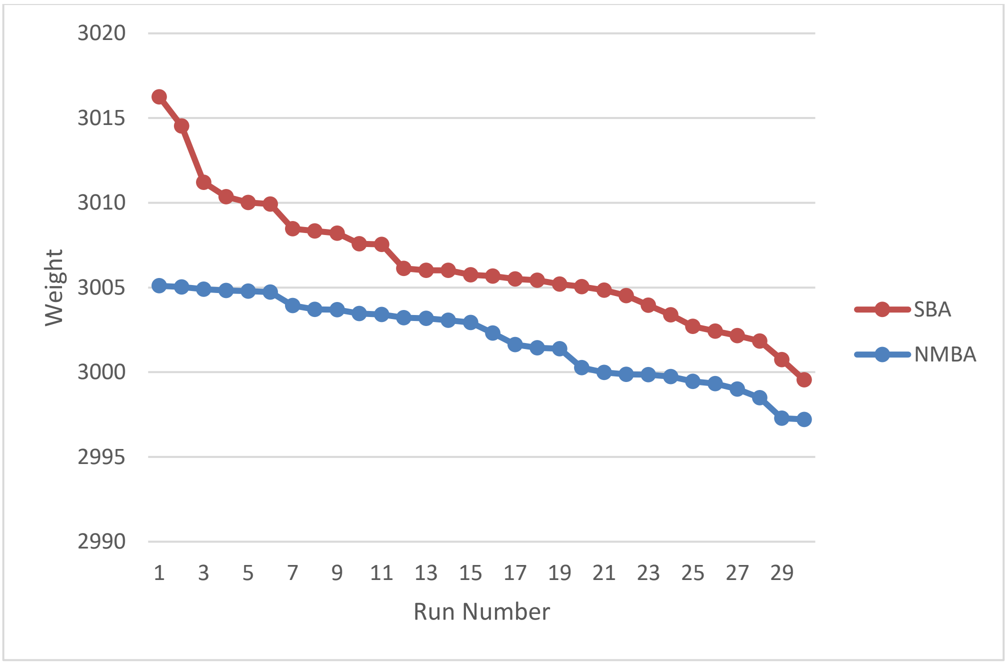

Figure 3.9: Minimum weights found by Standard Bees Algorithm and NMBA over 30 runs for the Speed Reducer problem

Table 3.9: **Parameter and constraint values of the best solutions obtained by NMBA and others for Welded Beam**

	(μ+λ)ES	ABC	BA	NMBA
x_1	0.20573	0.20573	0.20206	0.19886
x_2	3.470489	3.470489	3.56971	3.42656
x_3	9.036624	9.036624	9.03640	9.57797
x_4	0.205729	0.20573	0.20584	0.20326
g_1	0	0	-55.8952	-17.2147
g_2	0.000002	−0.000002	-14.0521	-2970.27
g_3	0	0	-0.00377	-0.0044
g_4	−3.432984	−3.432984	-3.42349	-3.3637
g_5	−0.080730	−0.080730	-0.07706	-0.07386
g_6	−0.235540	−0.235540	-0.23555	-0.23771
g_7	−0.000001	0	-9.23549	-6.86761
$f(x)$	1.724852	1.724852	1.733247	1.73176

Table 3.10: **Parameter and constraint values of the best solutions obtained by NMBA and others for Pressure Vessel**

	SCA	PSO	(μ+λ)ES	ABC	BA	NMBA
x_1	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375
x_3	41.9768	42.098446	42.098446	42.098446	41.811295	42.095279
x_4	182.2845	176.636052	176.636596	176.636596	180.260978	176.677711
g_1	−0.0023	0	0	0	-0.005542	-6.11E-05
g_2	−0.0370	−0.035881	0.03588	−0.035881	-0.038620	-0.035911
g_3	−23420.5966	0	0	−0.000226	-183.1446	-10.4343
g_4	−57.7155	−63.363948	−63.363404	−63.363404	-59.73902	-63.32229
$f(x)$	6171	6059.70161	6059.7143	6059.714339	6103.417489	6067.896372

Table 3.11: **Parameter and constraint values of the best solutions obtained by NMBA and others for Tension Spring**

	SCA	PSO	(μ+λ)ES	ABC	BA	NMBA
x_1	0.0521602	0.05169	0.052836	0.051749	0.052312	0.051413
x_2	0.368159	0.35675	0.384942	0.358179	0.371889	0.350089
x_3	10.648442	11.287126	9.807729	11.203763	10.452475	11.690453
g_1	0	0	−0.000001	−0.000000	-1.91E-05	-5.68E-05
g_2	0	0	0	−0.000000	-2.44E-05	-8.21E-05
g_3	−4.075805	−4.053827	−4.106146	−4.056663	-4.082842	-4.040086
g_4	−0.719787	−0.727706	−0.708148	−0.726713	-0.717198	-0.732331
$f(x)$	0.012669	0.012665	0.012689	0.012665	0.012673	0.012669

Table 3.12: **Parameter and constraint values of the best solutions obtained by NMBA and others for Speed Reducer**

	SCA	(μ+λ)ES	ABC	BA	NMBA
x_1	3.5	3.499999	3.499999	3.501481	3.501600
x_2	0.7	0.699999	0.7	0.700124	0.700016
x_3	17	17	17	17	17
x_4	7.327602	7.3	7.3	7.388378	7.307274
x_5	7.715321	7.8	7.8	7.844692	7.802366
x_6	3.350267	3.350215	3.350215	3.350536	3.350318
x_7	5.286655	5.286683	5.2878	5.287016	5.286723
g_1	−0.073915	−0.073915	−0.073915	-0.074636	-0.074382
g_2	−0.197999	−0.197998	−0.197999	-0.198623	-0.198403
g_3	−0.493501	−0.499172	−0.499172	-0.969473	-9.70E-01
g_4	−0.904644	−0.901472	−0.901555	-0.994106	-0.994199
g_5	0	0	0	-0.000141	-8.10E-05
g_6	0.000633	0	0	-0.000181	-2.21E-05
g_7	−0.7025	−0.702500	−0.7025	-0.702447	-0.702493
g_8	0	0	0	-0.000245	-0.000433
g_9	−0.583333	−0.583333	−0.583333	-0.583231	-0.583153
g_{10}	−0.054889	−0.051325	−0.051326	-0.062608	-0.052249
g_{11}	0	−0.010852	−0.010695	-0.016441	-0.011147
$f(x)$	2994.74424	2996.34809	2997.05841	2999.54709	2997.21932

3.4 Summary

This chapter proposed a modified version of the Bees Algorithm known as NMBA. In this NMBA, a NM method was used in the local search to further enhance the performance of the Bees Algorithm for unconstraint benchmark functions. The addition of NM method in the local search provides guidance for the direction of the local search to find optimal solutions faster than the standard Bees Algorithm. Following the addition of the NM method into the local search, NMBA showed better performance than the standard Bees Algorithm in various benchmark functions, especially in terms of convergence speed.

In addition, the NMBA was also applied to four constrained engineering mechanical problems. Experimental results showed that the NMBA significantly outperformed the standard Bees Algorithm for one out of four mechanical design problems. Despite this, the standard Bees Algorithm still does not outperform NMBA for the remaining three engineering problems. Therefore, this proposed algorithm has the potential to be utilised for different real world engineering problems.

CHAPTER 4

Recombination Based Bees Algorithm

4.1 Preliminaries

The recombination operator is one of the essential elements in Evolutionary Algorithm (EA) for creating new solutions from two or more old solutions in each generation. The successfulness of recombination operator in EA has inspired it to be used in the Standard Bees Algorithm. Thus, this chapter presents a new version of Bees Algorithm with recombination operator at two different stages. In this proposed Bees Algorithm, the recombination operators were added at local search and best abandoned sites to produce new solutions closer to global or local optima, improving the exploitation capability.

The rest of this chapter describes the proposed Bees Algorithm in detail including the performance on a set of unconstraint benchmark functions, performance on constrained mechanical design problems, and the conclusion of this work in section 4.2, section 4.3 and section 4.4 respectively.

4.2 Recombination Based Bees Algorithm

In this section, a new version of the Bees Algorithm known as the recombination based Bees Algorithm (*r*BA) is introduced. The *r*BA uses recombination operator at two different stages of the standard Bees Algorithm, which are the local search and best abandoned sites. Although the Bees Algorithm has been proven to be very good in solving a wide range of real world problems, further improvement is needed to expand the capabilities of the standard Bees

Algorithm. Therefore, utilising recombination operator in the standard Bees Algorithm may produce solutions closer to global or local optima for better exploitation capability.

The recombination operator itself was inspired from commonly used variation operators in Evolution Algorithm operator known as recombination operator or crossover operator, which mimics the reproduction in biological processes. This recombination operator produces one or more child solutions from parent solutions. The main idea of this recombination operator is to combine different partitioned variables solutions into new solutions. An example of recombination operators (D=dimension) between two solutions is described in Figure 4.1 below.

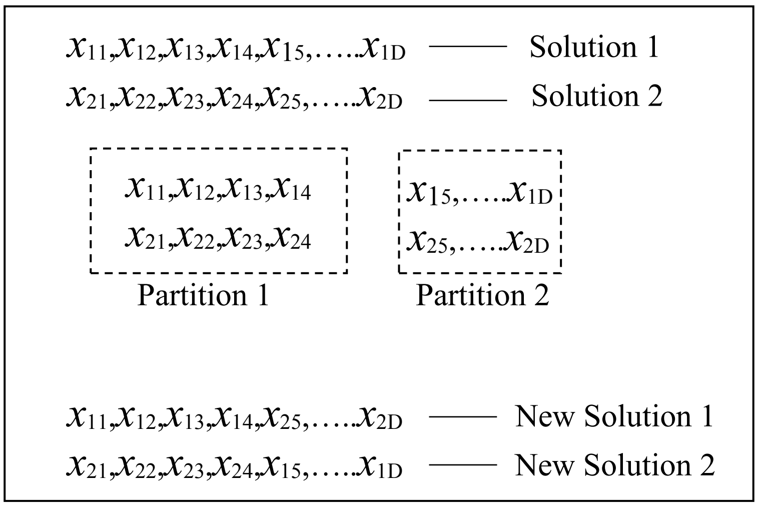
$x_{11},x_{12},x_{13},x_{14},x_{15},\ldots x_{1D}$ — Solution 1
$x_{21},x_{22},x_{23},x_{24},x_{25},\ldots x_{2D}$ — Solution 2
$x_{11},x_{12},x_{13},x_{14}$
$x_{21},x_{22},x_{23},x_{24}$
$x_{15},\ldots x_{1D}$
$x_{25},\ldots x_{2D}$
Partition 1
Partition 2
$x_{11},x_{12},x_{13},x_{14},x_{25},\ldots x_{2D}$ — New Solution 1
$x_{21},x_{22},x_{23},x_{24},x_{15},\ldots x_{1D}$ — New Solution 2

Figure 4.1: Recombination operator mechanism

The proposed recombination operator starts with two good solutions divided into two partitions randomly. The number of variables in the partition is determined randomly. Then, two new solutions are formed by combining two different solutions which have been divided into different partitions. This operator could produce new solutions with better fitness values compared to old solutions. For this reason, this recombination operator is added into the standard Bees Algorithm. In addition, additional parameters are also not required for the implementation of this recombination operator into the standard Bees Algorithm because the solutions are divided into partitions randomly.

The overall flowchart of this proposed algorithm is presented in Figure 4.2. Currently, once the fittest bee for each patch has been found in the standard Bees Algorithm, a global search is performed as described in Section 2.6. However, in this proposed algorithm two additional stages are added into the standard Bees Algorithm. The additional stages of the proposed algorithm are recombination operator between two elite sites after selection of the fittest patch and recombination operators on two best abandoned sites.

The first modification in the standard Bees Algorithm is by adding recombination operator after selection of the fittest patch between two elite sites. Once the fittest recruited bee for each patch has been selected, a recombination operator between those two elite sites is executed to generate two new positions. If the new positions generated by the recombination operator have better fitness than the elite sites, the new position substitutes the elite site as the new elite site. To avoid excessive computation, the recombination operator is only applied to elite sites instead of all best sites. The main idea of performing recombination operator between two elite sites is to combine the components of two good solutions, producing better new solutions than the old solutions.

After completion of recombination operator on elite sites, another modification in the standard Bees Algorithm is adding recombination operator between two best abandoned sites. As the search progresses, the best sites might abandon those sites that failed to find a better solution. Thus, two best abandoned sites are selected as soon as two best abandoned sites are available. Then, recombination operation between these two sites is done to produce two new positions.

If the fitness value of the new position is better than the best solution so far, the new position is merged into the new population for the next generation. The mechanism of merging the improved abandoned site back into the population is essential in improving the standard Bees Algorithm because for few landscape features, the standard Bees Algorithm failed to converged. If the standard Bees Algorithm failed to converged, few best solutions found so far may have good chances to be improved by recombination operator. Therefore, adding recombination operator on those best abandoned sites could produces better new sites.

In order to understand how recombination operator works on two elite sites or on abandoned sites, an example of the search behaviour in two dimensional spaces is described in Figures 4.3-4.5. The search mechanism starts with two distinct elite sites after selecting the fittest recruited bee for each patch. In Figure 4.3, two initial elite sites are selected with fitness value $f(elite_1)$ of $elite_1$ better than fitness value $f(elite_2)$ of $elite_2$. Then, two new solutions are produced by recombination operation as shown in Figure 4.4. After the fitness evaluation of those new solutions, new elite_1 replaces $elite_2$ as new elite site for next generation because the new elite_1 fitness value is better than $elite_2$ fitness value. This additional search behaviour enhances the capabilities of the standard Bees Algorithm to converges to better solutions.

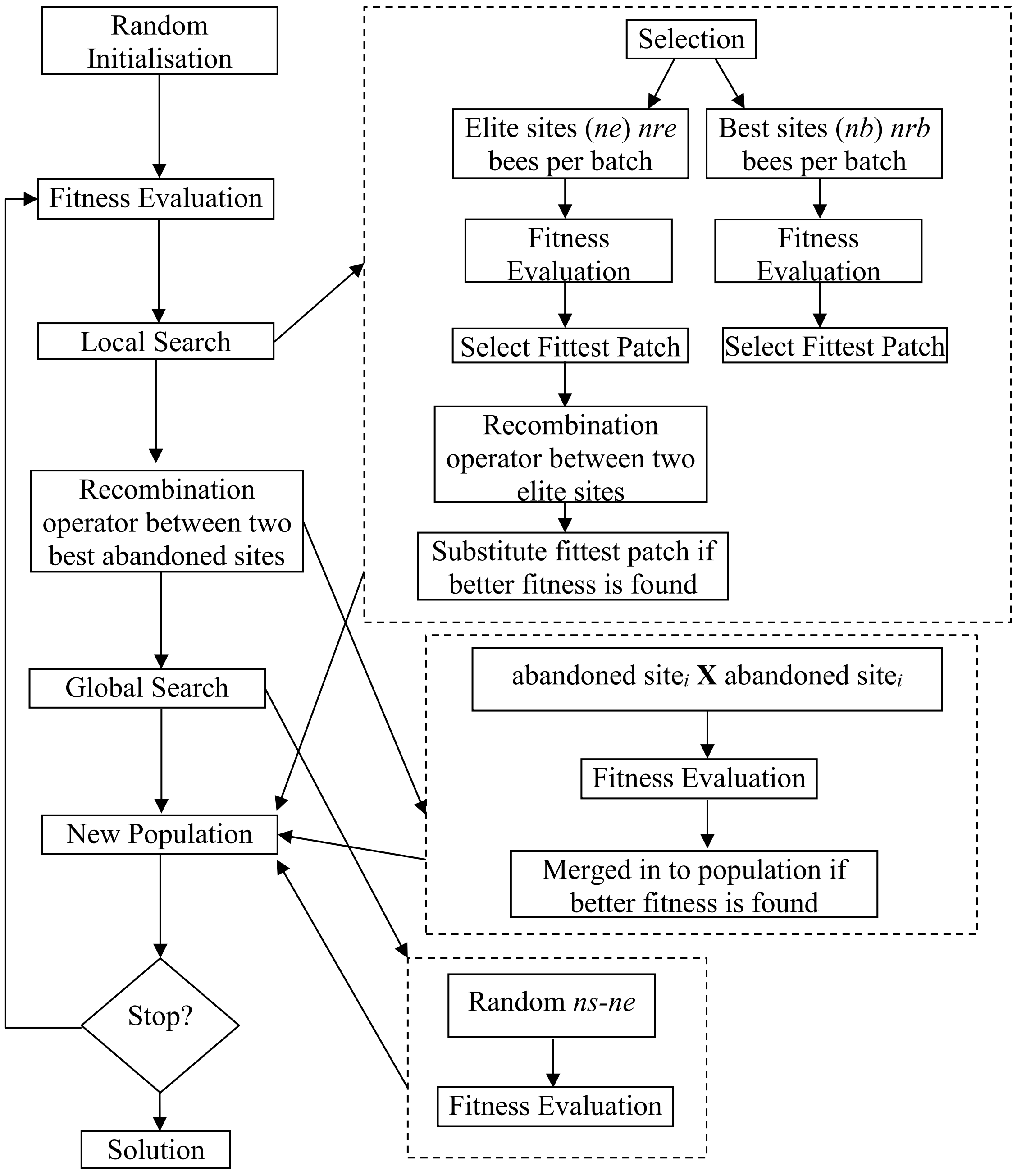
Random Initialisation
Fitness Evaluation
Local Search
Recombination operator between two best abandoned sites
Global Search
New Population
Stop?
Solution
Selection
Elite sites (ne) nre bees per batch
Best sites (nb) nrb bees per batch
Fitness Evaluation
Fitness Evaluation
Select Fittest Patch
Select Fittest Patch
Recombination operator between two elite sites
Substitute fittest patch if better fitness is found
abandoned site_i X abandoned site_i
Fitness Evaluation
Merged in to population if better fitness is found
Random ns-ne
Fitness Evaluation

Figure 4.2: Flowchart of the recombination based Bees Algorithm

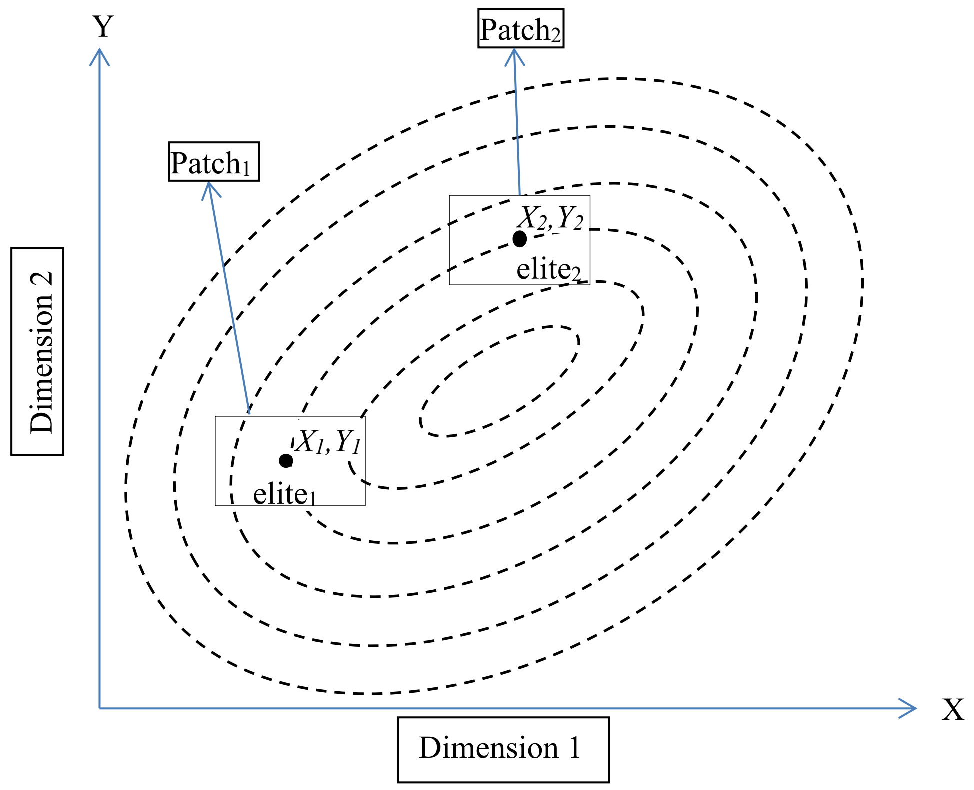

Figure 4.3: Two initial elite solutions ($f(elite_1) < f(elite_2)$)

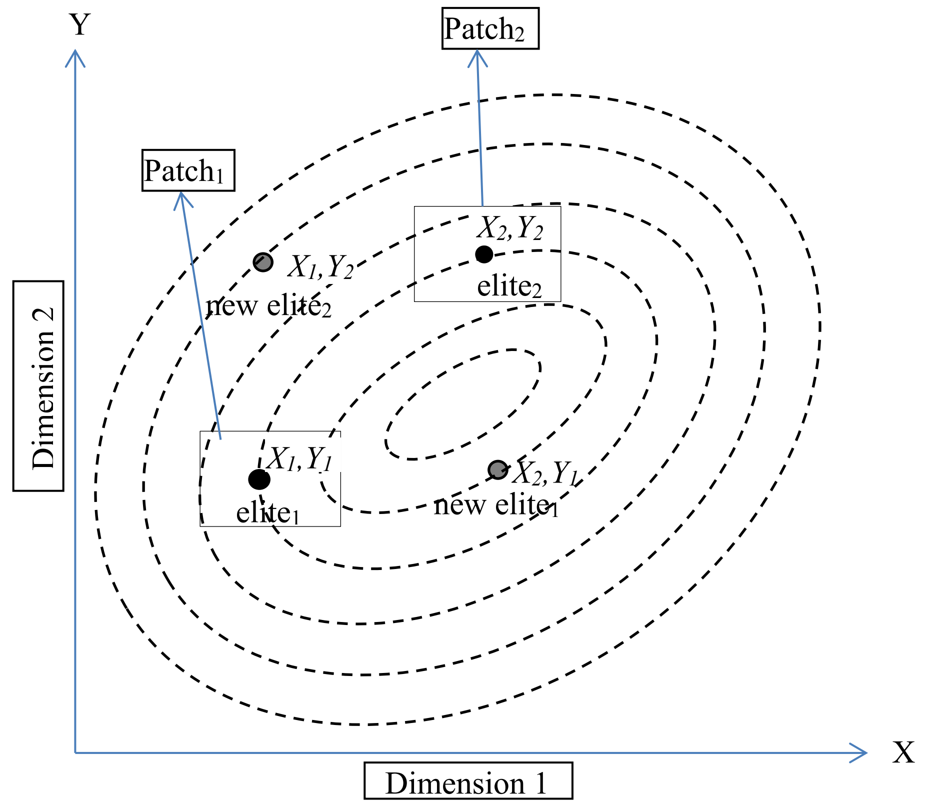

Figure 4.4: Two new solutions produced (f(new $elite_1$)$<f(elite_1)<f(elite_2)<f$(new $elite_2$))

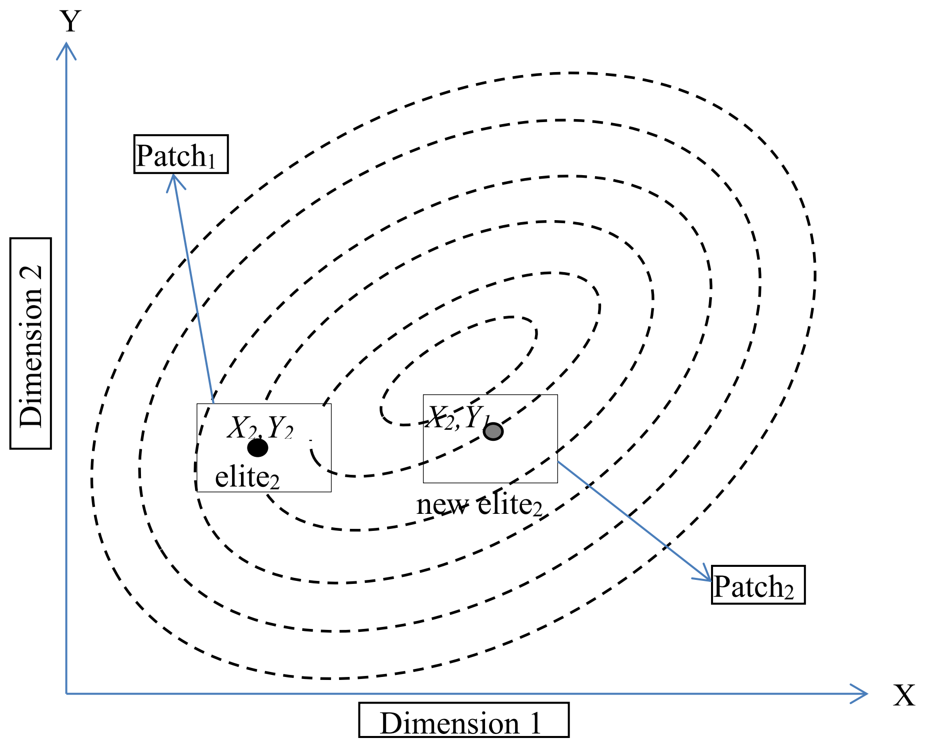

Figure 4.5: The new elite$_2$ selected as new elite site replacing *elite*$_2$

4.2.1 Experimental Setup

This section describes the experimental setup used in this chapter to test the performance of the proposed algorithm. In general, similar experimental set up were utilised in this study by using a similar set of benchmark functions (see Appendix A), similar parameter settings (Table 3.1), stopping criterion and method of comparison as in Section 3.2.

4.2.2 Experimental Results

Table 4.1 presents the median and standard deviation of accuracy achieved by standard Bees Algorithm, *rBA*, SPSO2011 and qABC algorithm over 100 runs. In order to make the comparison clearer, the accuracy on this table is recorded as 0.0000 if the experimental result obtained were less than 0.001. Based on the median accuracy results in Table 4.1, the *r*BA finds acceptable solutions on *Easom*, *Trid*, *Zakharov*, *Schaffer*, *Schwefel*, *Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump* and *Shekel_4D* (ten functions). Similarly, for the standard Bees Algorithm, global optimum values were found on same functions as found by *r*BA.

Meanwhile, SPSO2011 found acceptable solutions on *Easom*, *Trid*, *Zakharov*, *Schaffer*, *Schwefel*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump* and *Shekel_4D* (nine functions) and qABC found on *Easom*, *Schaffer*, *Rastrigin*, *Schwefel*, *Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump* and *Shekel_4D* (nine functions).

Next, Table 4.2 presents experimental result comparisons on function evaluations for *r*BA against standard Bees Algorithm, SPSO2011 and qABC algorithm over 100 runs. The results on Table 4.2 show that *r*BA converged faster than the standard Bees Algorithm in most of the functions tested except for six functions, which were *Rosenbrock*, *Rastrigin*, *Shekel_4D*, *Shekel_10D*, *Griewank*, and *Langermann*. For *Shekel_4D,* although the median function evaluations recorded is less than the standard Bees Algorithm, the Mann Whitney test showed (See Table 4.3) that these results are not statistically significant. Thus, it is said that the convergence speed for this function compared to the standard Bees Algorithm is comparable. For the remaining five functions, the performance is comparable in terms of speed as the median values recorded were maximum function evaluations.

In the case of comparison with SPSO2011, the results on Table 4.2 shows that the *r*BA performed better than SPSO2011 in terms of convergence speed on eight functions, which were *Easom*, *Zakharov*, *Schwefel*, *Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump*, and *Shekel_4D* (eight functions). As for comparison between *r*BA and qABC algorithm, the *r*BA found acceptable solutions with less function evaluations on *Trid*, *Zakharov*, *Schaffer, Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, and *Shekel_4D* (seven functions). For all improvements, the *p*–values are less than 0.05 as shown in Table 4.4, which indicate the difference is significant.

Table 4.1: **Comparison on accuracy over 100 runs for *r*BA**

Function	SBA		*r*BA		SPSO2011		qABC	
	Median	StdDev	Median	StdDev	Median	StdDev	Median	StdDev
Easom	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Trid	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.3016	0.3062
Rosenbrock	4.8919	1.0673	4.3628	1.2898	6.9007	9.4754	**0.0758**	**0.0703**
Zakharov	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0235	0.0867
Schaffer	**0.000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0010**
Rastrigin	12.9437	3.2667	3.9840	1.4408	6.4253	2.2038	**0.0000**	**0.0219**
Schwefel	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Michaelewicz	**0.0000**	**0.0133**	**0.0000**	**0.0002**	0.0411	0.0895	**0.0000**	**0.0000**
Goldstein & Price	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.000**	**0.0000**	**0.0000**
Martin & Gaddy	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.000**	**0.0000**	**0.0000**
Camel Six Hump	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.000**	**0.0000**	**0.0000**
Shekel_4D	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**2.5068**	**0.0000**	**0.0014**
Shekel_10D	**0.0015**	**2.6435**	**0.0015**	**3.2433**	8.7202	0.2236	8.7231	0.0036
Griewank	0.1091	0.0226	0.0668	0.0194	**0.0341**	**0.0271**	0.0800	0.0389
Langermann	1.2563	0.0989	1.2545	0.1512	**0.9677**	**0.1363**	1.1712	0.0916

Table 4.2: **Comparison on function evaluations over 100 runs for *r*BA**

Function	SBA		*r*BA		SPSO2011		qABC	
	Median	StdDev	Median	StdDev	Media	StdDev	Median	StdDev
Easom	3124	1408.92	1897	663.15	4000	885.23	**1050**	**6888.02**
Trid	13224	4255.14	10320	2585.29	**9100**	**700.66**	500000	0
Rosenbrock	**500000**	**0**	**500000**	**0**	**500000**	**117878**	**500000**	**0**
Zakharov	16374	1896.45	**13888**	**1943.12**	35100	4619.08	500000	0
Schaffer	5305.5	2869.26	2988.5	1728.23	**2800**	**1728.96**	43951	225352.2
Rastrigin	500000	0	500000	0	500000	0	**352561.5**	**97711.63**
Schwefel	2825	100.51	1376	455.321	14100	30860.92	**850**	**12132.96**
Michaelewicz	163807.5	155525.4	**13703**	**17848.86**	500000	223339.3	105399	88612.96
Goldstein & Price	1824	729.96	**1480**	**493.77**	4000	501.39	2350	35168.90
Martin & Gaddy	1464	732.91	**1272**	**587.83**	2000	459.17	15350	204091.6
Camel Six Hump	924	315.71	**752**	**306.40**	3031	581.48	850	405.29
Shekel_4D	10628	8756.23	**8470.5**	**10583.84**	82500	230204.6	37850	65029.46
Shekel_10D	**500000**	**168591.4**	**500000**	**180750.20**	**500000**	**0**	**500000**	**0**
Griewank	**500000**	**0**	**500000**	**0**	**500000**	**105560.5**	**500000**	**33908.05**
Langermann	**500000**	**0**	**500000**	**0**	**500000**	**0**	**500000**	**0**

Table 4.3: **Significant difference of *r*BA's median results against standard Bees Algorithm**

	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000
Trid	ns	-	s	0.0000
Rosenbrock	s	0.0159	ns	-
Zakharov	ns	-	s	0.0000
Schaffer	ns	-	s	0.0000
Rastrigin	s	0.0000	ns	-
Schwefel	ns	-	s	0.0000
Michaelewicz	ns	-	s	0.0000
Goldstein & Price	ns	-	s	0.0000
Martin & Gaddy	ns	-	s	0.0076
Camel Six Hump	ns	-	s	0.0019
Shekel_4D	ns	-	ns	0.0082
Shekel_10D	ns	-	ns	-
Griewank	s	0.0000	ns	-
Langermann	ns	0.2670	ns	-

s: statistically significant, ns: not significant

'-': No statistical difference test required

Table 4.4: **Significant difference of *r*BA's median results against other algorithms**

	vs SPSO2011				vs qABC			
	accuracy	*p*-value	speed	*p*-value	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000	ns	-	s	0.0000
Trid	ns	-	s	0.0000	s	0.0000	s	0.0000
Rosenbrock	ns	0.05876	ns	-	s	0.0000	ns	-
Zakharov	ns	-	s	0.0000	s	0.0000	s	0.0000
Schaffer	ns	-	ns	0.4839	ns	-	s	0.0000
Rastrigin	s	0.0000	ns	-	s	0.0000	s	0.0000
Schwefel	ns	-	s	0.0000	ns	-	s	0.0001
Michaelewicz	s	0.0173	s	0.0000	ns	-	s	0.0000
Goldstein & Price	ns	-	s	0.0000	ns	-	s	0.0001
Martin & Gaddy	ns	-	s	0.0000	ns	-	s	0.0000
Camel Six Hump	ns	-	s	0.0000	ns	-	ns	0.0819
Shekel_4D	ns	-	s	0.0000	ns	-	s	0.0000
Shekel_10D	s	0.0000	ns	-	s	0.0000	ns	-
Griewank	s	0.0000	ns	-	s	0.0000	ns	-
Langermann	s	0.0000	ns	-	s	0.0000	ns	-

s: statistically significant, ns: not significant
'-': No statistical difference test required

Table 4.5 presents the comparison results of overall performance of the *r*BA against other algorithms. From Table 4.5, it is shown that the *r*BA achieved top rank performance on most of the functions tested (nine out of fifteen). For comparisons with other algorithms show that SPSO2011 and qABC performed better than *r*BA on two functions (two functions comparable to *r*BA) and four functions respectively. As for comparison with the standard Bees Algorithm, the *r*BA outperformed the standard Bees Algorithm on twelve out of fifteen functions considered. The remaining three functions show comparable results compared to the proposed algorithm. Overall, these results indicate that the *r*BA outperformed other algorithms in most cases, especially in terms of convergence speed where it can find the global optimum in most cases with fewer number of function evaluations.

Table 4.5: **Comparison of overall performance for *r*BA**

Function	**SBA**			***r*BA**			**SPSO2011**			**qABC**		
	succ.	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**
Easom	100	X		100	X		100	X		100	X	X
Trid	100	X		100	X	X	100	X	X	0		
Rosenbrock	0			0			21			0	X	X
Zakharov	100	X		100	X	X	100	X		0		
Schaffer	100	X		100	X	X	100	X	X	68	X	
Rastrigin	0			0			0			88	X	X
Schwefel	100	X		100	X		100	X		100	X	X
Michaelewicz	88	X		100	X	X	39			100	X	
Goldstein & Price	100	X		100	X	X	100	X		100	X	
Martin & Gaddy	100	X		100	X	X	100	X		79		
Camel Six Hump	100	X		100	X	X	100	X		100	X	
Shekel_4D	100	X	X	100	X	X	59	X		99	X	
Shekel_10D	41	X	X	46	X	X	0			0		
Griewank	0			0			6	X	X	2		
Langermann	0			0			2	X	X	0		
Total	1029		2	1046		9	927		4	836		4

succ.: successful rate, acc.: accuracy, perf.: performance

4.2.3 Discussion

The results mentioned in the previous section show that the proposed *r*BA in general accomplished significant improvements compared to the standard Bees Algorithm and other algorithms in terms of finding the global optimum and convergence speed. A possible explanation for this improvement is that the recombination operator helps the exploitation in the local search. It has been explained by Pham and Castellani (2009) that the only local search used in the standard Bees Algorithm, which is random mutations is not sufficient to generate favourable random mutations, especially for high dimensional and difficult fitness landscapes as the fitness solution decreases. Even though the local search in the standard Bees Algorithm assisted by neighbourhood shrinking strategy increases the like hood of finding local peaks, additional exploitation strategies could further improve convergence speed and accuracy of the Bees Algorithm. For this reason, recombination operator applied on elite sites helps the proposed algorithm to find better solutions at fewer function evaluations.

However, the recombination operator may not be suitable for non-separable complex multimodal functions such as *Shekel_4D*, *Shekel_10D* and *Langermann* due to combination of complex landscape and interdependence between variables. Thus, the usage of recombination operator on these functions might cause the recombination operator to produces new positions away from the local peak. This reason explains the similar performance of the proposed algorithm on *Shekel_4D*, *Shekel_10D* and *Langermann* functions against the standard Bees Algorithm.

Furthermore, the *r*BA also failed to find the global optimum on *Rosenbrock*, *Rastrigin*, and *Griewank* functions. A possible explanation for these results may be due to landscape surface of the functions. The location of the global minimum for *Rosenbrock* function on narrow, curved valleys (Kang et al., 2011) might cause difficulty in solving this type of function. The *Rastrigin* and *Griewank* functions are categorised as a function that has an overall unimodal behaviour with large numbers of local optima (Pham and Castellani, 2009). This type of characteristic could attribute to non-convergence on these two functions. Another possible explanation for this is related to the neighbourhood size used, which is the same for all functions. Fine tuning the neighbourhood size value could possibly improve the results.

Although three functions mentioned above were not able to reach global optimum, the Mann Whitney test showed that the results on median accuracy obtained by *r*BA against the standard Bees Algorithm were significantly different (See Table 4.3). It thus can be suggested that the recombination operator moved the local search closer to the global or local peak finding better solutions. Figures 4.6-4.8 present the convergence plot of all these functions during the search. It is clear from these figures that the *r*BA achieved better fitness values than the standard Bees Algorithm.

Meanwhile, Figures 4.9-4.11 display the cumulative frequency of recombination operator on elite and abandoned sites found better fitness for *Rosenbrock*, *Rastrigin* and *Griewank* functions respectively. These figures indicate that the recombination operator is more effective on elite sites rather than the abandoned sites as more better solutions produced by the recombination operator during the search progress. For the remaining of the functions, the *r*BA performed considerately well compared to the standard Bees Algorithm.

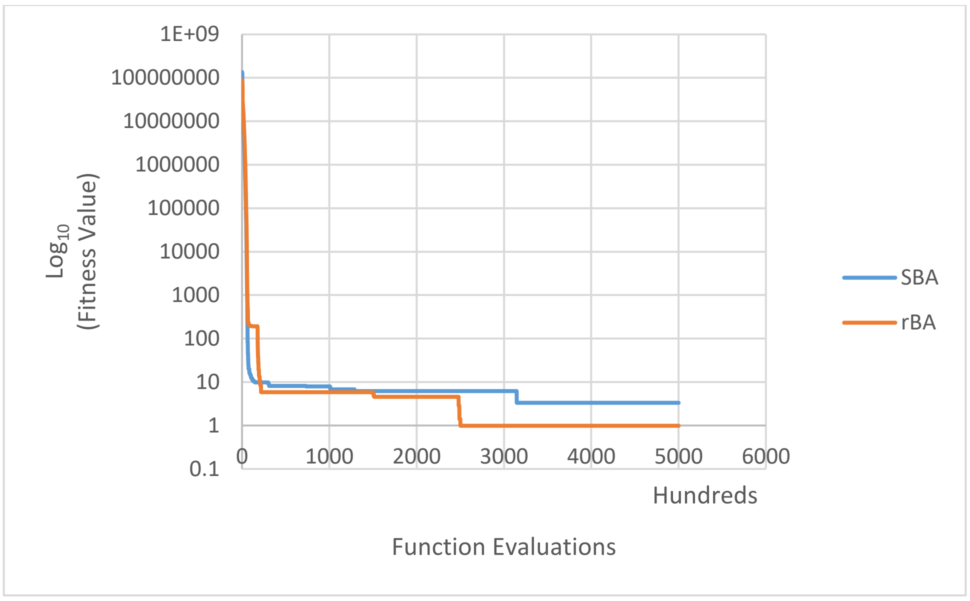

Figure 4.6: Plot of convergence for the *Rosenbrock* function

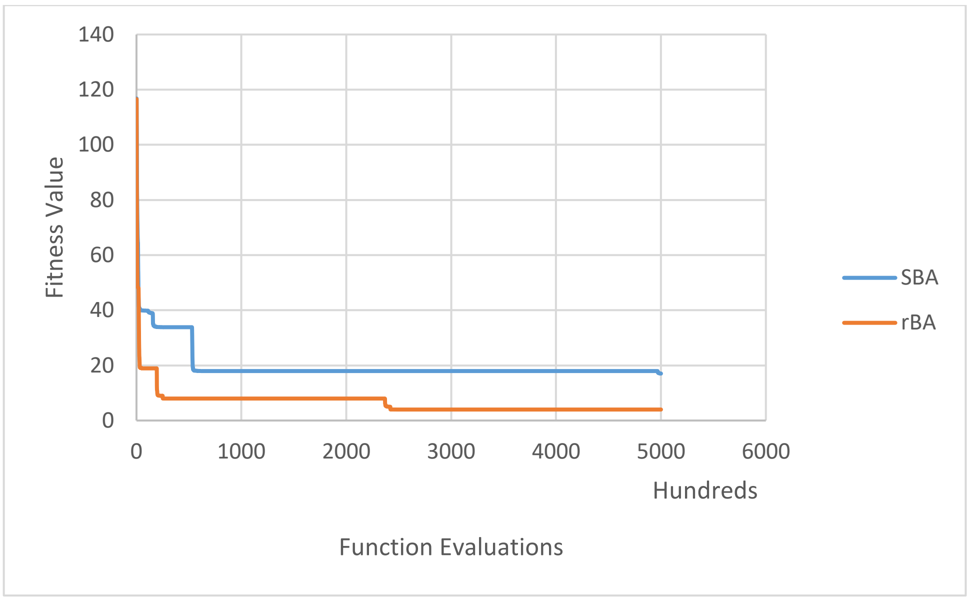

Figure 4.7: Plot of convergence for the *Rastrigin* function

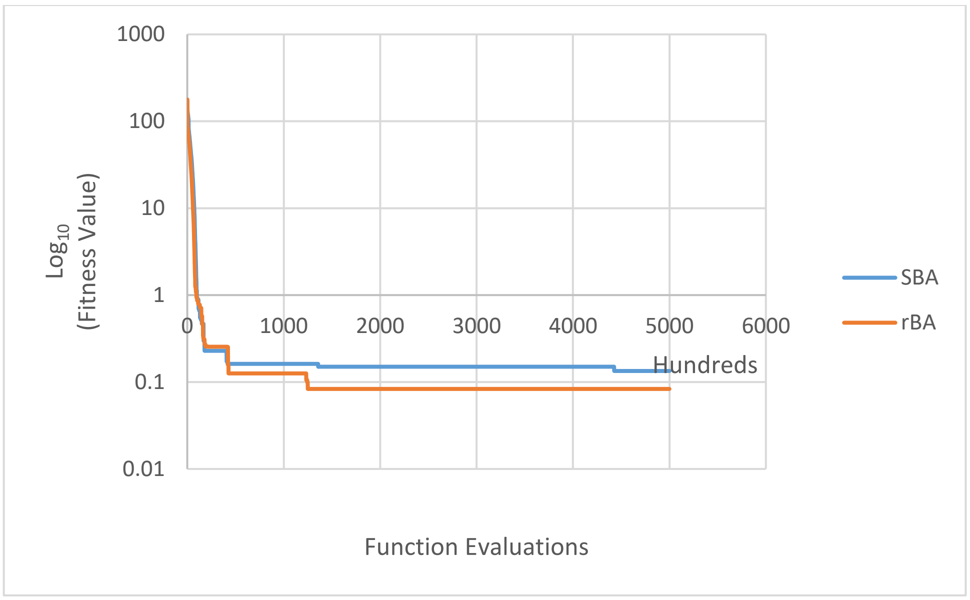

Figure 4.8: Plot of convergence for the *Griewank* function

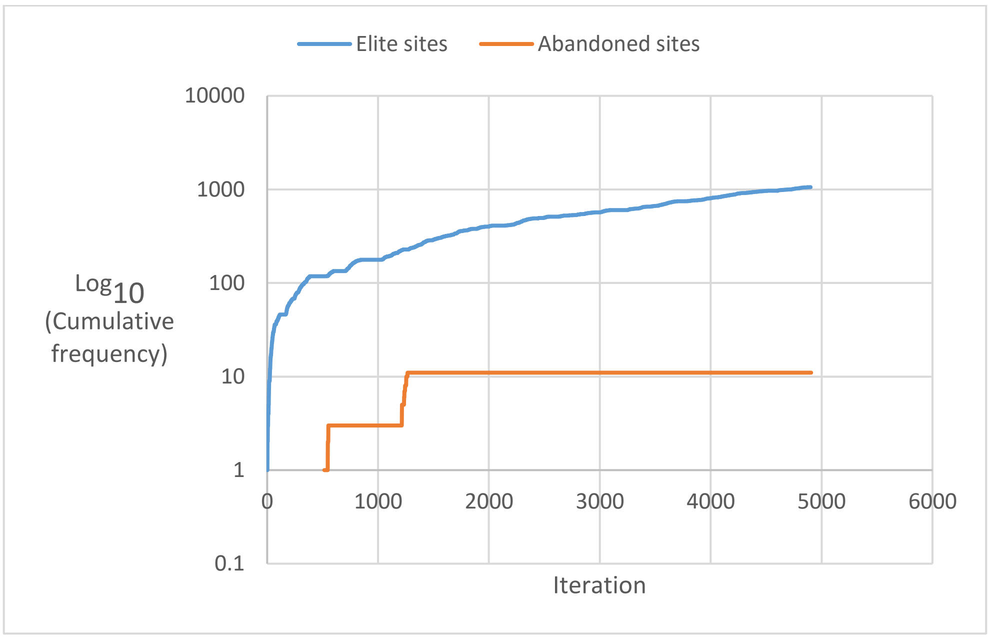

Figure 4.9: Cumulative frequency of recombination operator found better solutions from elite sites and abandoned sites for the *Rosenbrock* function

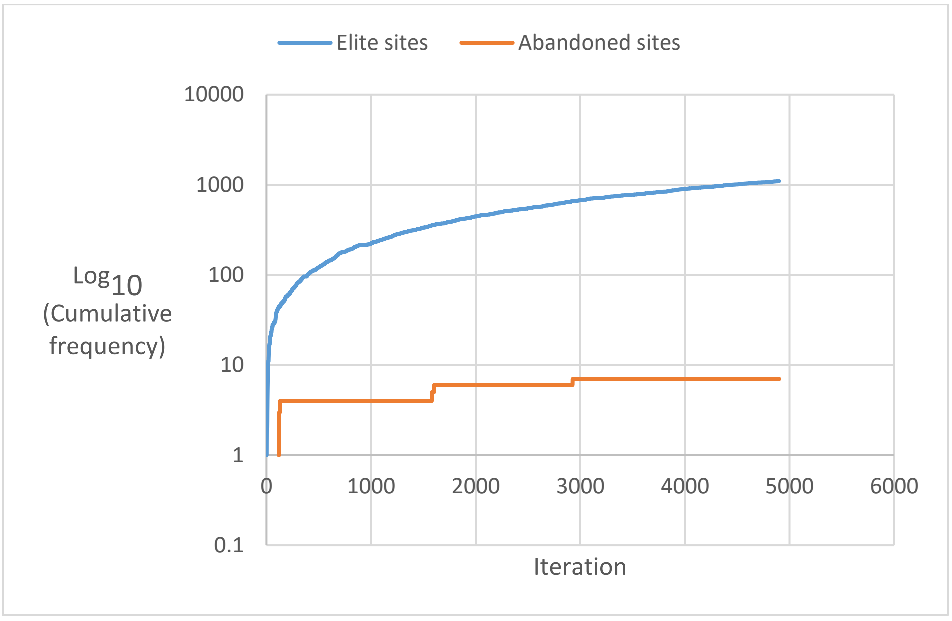

Figure 4.10: Cumulative frequency of recombination operator found better solutions from elite sites and abandoned sites for the *Rastrigin* function

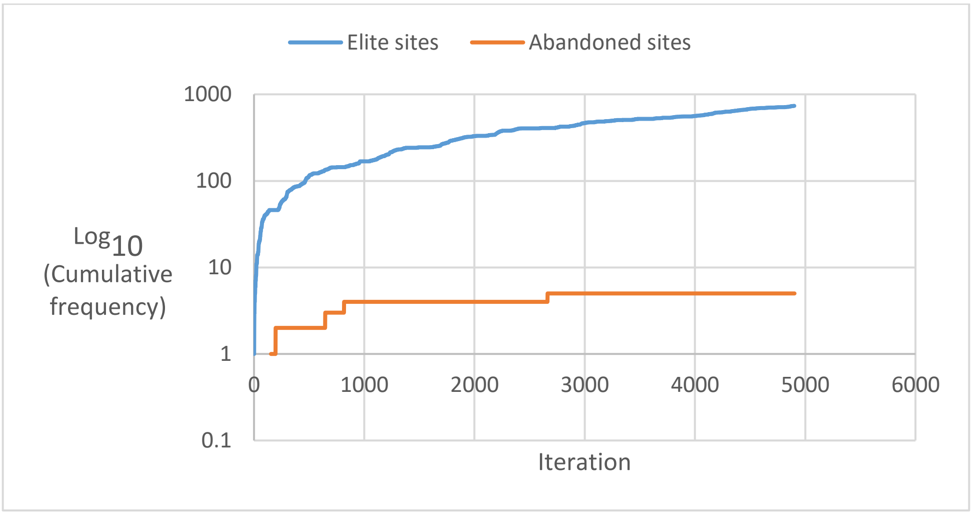

Figure 4.11: Cumulative frequency of recombination operator found better solutions from elite sites and abandoned sites for the *Griewank* function

Another important finding was the overall performance for functions where *r*BA did not perform as well as SPSO2011 and qABC. These results may be explained in terms of search mechanisms in those two algorithms. For SPSO2011, the search mechanism is based on individual and neighbourhood-based best known particle positions (Zambrano-Bigiarini et al., 2013). Moreover, this version of PSO has no mechanism to escape the local optima such as site abandonment strategy in the Bees Algorithm. These two differences in search mechanisms are likely the reasons SPSO2011 outperforms *r*BA on *Griewank* and *Langermann* functions.

For qABC, the search mechanism for the qABC algorithm only updates the position of bees in solution space by one random index dimension instead of all problem dimensions. Furthermore, the qABC also has long stagnation limit thus causing continuous exploitation on promising patches (Karaboga & Gorkemli, 2012). This might explain the top performance of qABC on *Easom*, *Rosenbrock*, *Rastrigin* and *Schwefel* functions.

4.3 Mechanical Design Applications

This section discusses on the application of the proposed Bees Algorithm on the four well known constrained mechanical design problems. The details of these mechanical design problems are described in Appendix C. In this experiment, same parameter settings as shown in Table 3.7 was adopted while the stopping criterion and number of runs were set at 30000 function evaluations and 30 times, respectively. On handling constraints, a similar method as explained in Section 3.3 was used in this experiment where only feasible solutions were accepted to be selected in the population.

The results of *r*BA were compared with the standard Bees Algorithm and the results extracted from Akay and Karaboga (2012) in terms of best solution found over 30 runs, median solution over 30 runs and function evaluation needed to reach that solution. Table 4.6 shows results obtained by *r*BA and the comparison against other results found by other researchers. In the case of comparing the median solution between the standard Bees Algorithm and the proposed Bees Algorithm, the Mann Whitney (at α = 0.05) test was used to check the significant difference between two median solutions. An algorithm is said to be better than another if the median value obtained is less than other one and the Mann Whitney test showed significant difference. If Mann Whitney test showed no significant difference, both algorithms are said to be comparable.

Table 4.6: **Comparison of *r*BA against others** (Akay and Karaboga, 2012)

Problem	Stats.	Sca[a]	Pso[a]	(μ+λ)-ES	UPSOm	ABC	BA	*r*BA
Welded Beam	Best	NA	NA	1.724852	1.92199	1.724852	1.7332472	1.727224
	Mean	NA	NA	1.777692	2.83721	1.741913	1.76792249	1.758381
	Median	NA	NA	NA	NA	NA	1.76697849	1.75392
	NFE	NA	NA	30,000	100,000	30,000	30000	30000
Pressure Vessel	Best	6171	6059.7143	6059.70161	6544.27	6059.71474	6103.41749	6068.358
	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.3081	6381.47138	6476.64
	Median	NA	NA	NA	NA	NA	6315.16763	6313.96
	NFE	20000	30000	30000	100000	30000	30000	30000
Tension/ Compresion Spring	Best	0.012669	0.012665	0.012689	0.01312	0.012665	0.01267303	0.012668
	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	0.01333878	0.012768
	Median	NA	NA	NA	NA	NA	0.0133214	0.01269
	NFE	25,167	15000	30000	100000	30000	30000	30000
Speed Reducer	Best	2994.74	NA	2996.348094	NA	2997.05841	2999.54709	2997.885
	Mean	3001.758	NA	2996.34809	NA	2997.05841	3006.31144	3001.627
	Median	NA	NA	NA	NA	NA	3005.71872	3001.55
	NFE	54,456	NA	30000	NA	30000	30000	30000

NA: not available, NFE:Number of function evaluations
[a]The welded beam problem is different from the one employed in this study

The results in Table 4.6 indicate that the median solutions over 30 runs found by *r*BA are better than the standard Bees Algorithm and the differences are significant for most of the problems except the pressure vessel problem. Figures 4.12–4.15 show variations of solutions obtained between both algorithms for all problems. Therefore, this shows that the *r*BA outperformed the standard Bees on the Welded Beam, Tension Spring and Speed Reducer problems but performed equally on the Pressure Vessel problem.

Moreover, in terms of best results, the (μ+λ) -ES (ABC found the same best), (μ+λ)-ES, PSO and Sca algorithms found the best solution for the Welded Beam, Pressure Vessel, Tension Spring and Speed Reducer problems respectively. Although the comparison of results with other algorithms show that the *r*BA did not perform as well as other the algorithms in terms of best solution, these results are relatively close compared to the results of *r*BA. In general, it seems that the performance of *r*BA on these four mechanical design problems compared with other algorithms is comparable. Tables 4.7–4.10 present the values of the design variables for the best solutions found by *r*BA, standard Bees Algorithm and other algorithms.

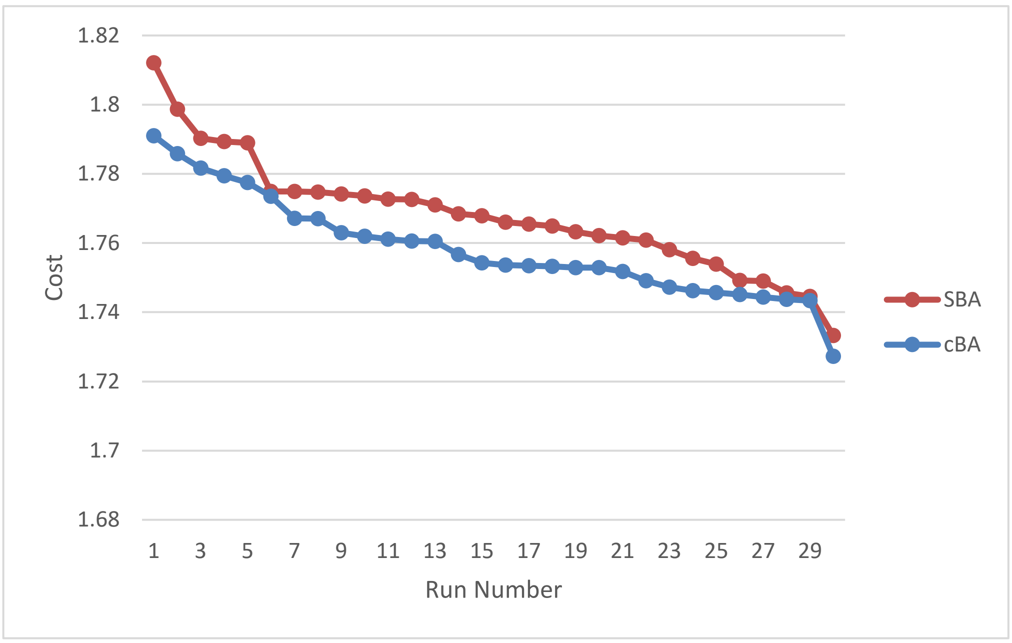

Figure 4.12: Minimum costs found by Standard Bees Algorithm and *r*BA over 30 runs for the Welded Beam problem

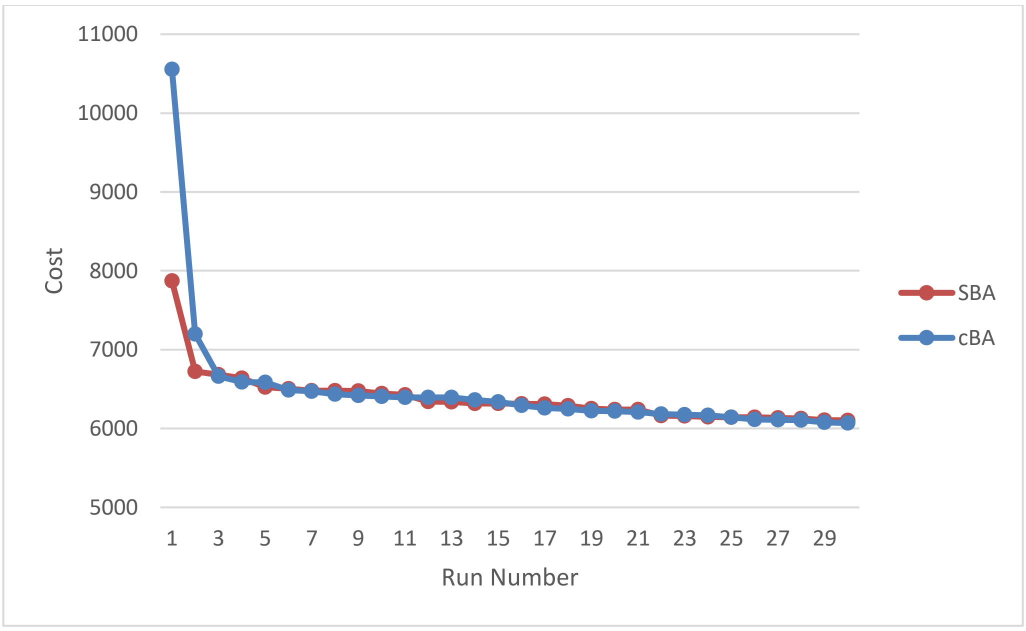

Figure 4.13: Minimum costs found by Standard Bees Algorithm and *r*BA over 30 runs for the Pressure Vessel problem

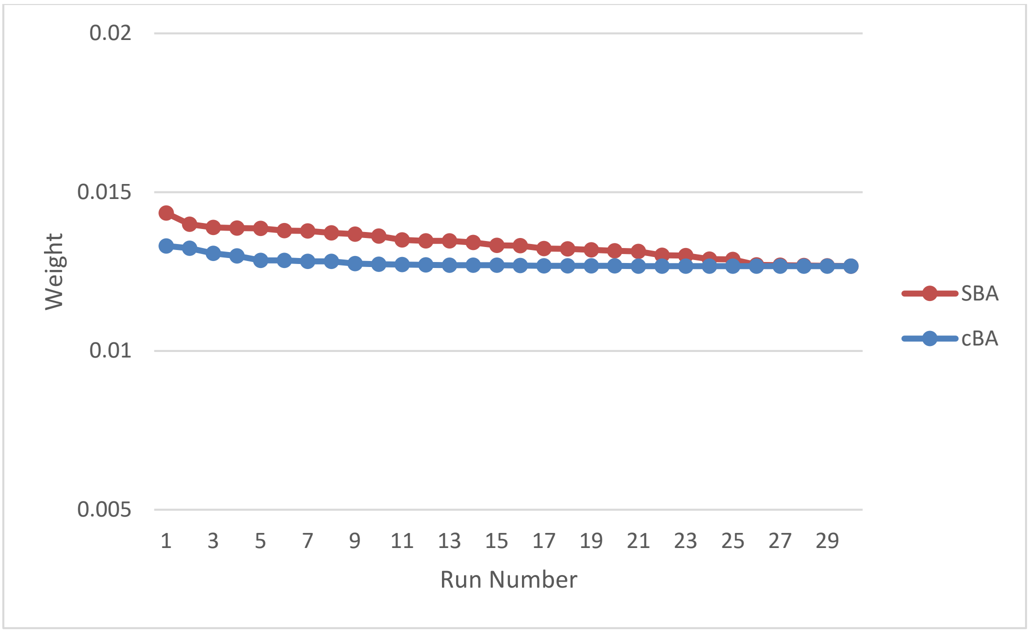

Figure 4.14: Minimum weights found by Standard Bees Algorithm and *r*BA over 30 runs for the Tension/Compression Spring problem

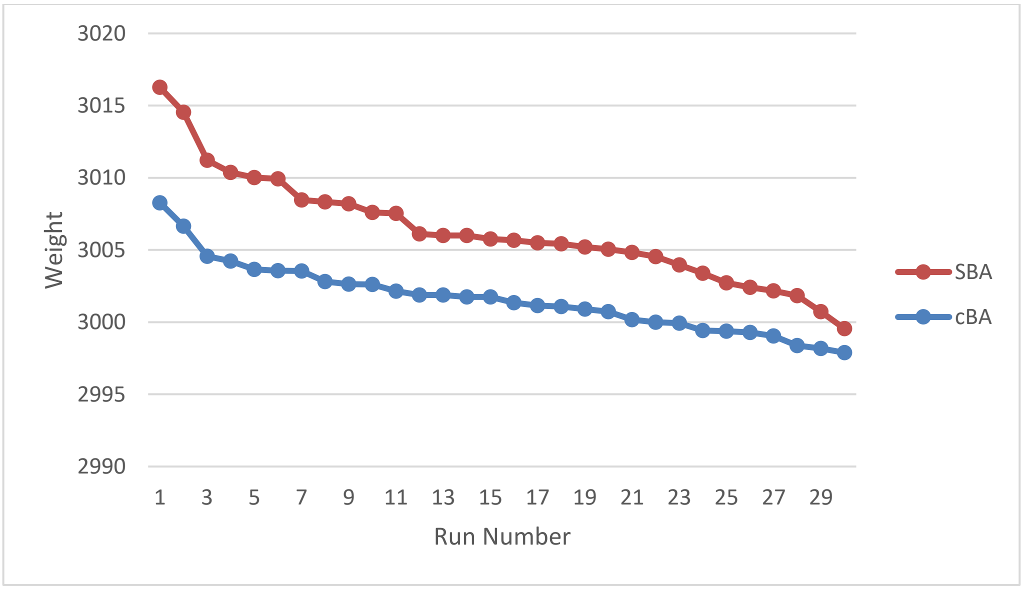

Figure 4.15: Minimum weights found by Standard Bees Algorithm and *r*BA over 30 runs for the Speed Reducer problem

Table 4.7: **Parameter and constraint values of the best solutions obtained by *r*BA and other algorithms for the Welded Beam Problem**

	(μ+λ)ES	ABC	BA	*r*BA
x_1	0.20573	0.20573	0.20206	0.20491
x_2	3.470489	3.470489	3.569718	3.490459
x_3	9.036624	9.036624	9.036399	9.038263
x_4	0.205729	0.20573	0.205836	0.205817
g_1	0	0	-55.89515	-8.98355
g_2	0.000002	−0.000002	-14.05208	-23.61819
g_3	0	0	-0.003773	-0.000906
g_4	−3.432984	−3.432984	-3.42349	-3.430284
g_5	−0.080730	−0.080730	-0.077064	-0.079911
g_6	−0.235540	−0.235540	-0.235547	-0.235554
g_7	−0.000001	0	-9.235489	-8.369106
$f(x)$	1.724852	1.724852	1.733247	1.731760

Table 4.8: **Parameter and constraint values of the best solutions obtained by *r*BA and other algorithms for Pressure Vessel**

	SCA	PSO	(μ+λ)ES	ABC	BA	*r*BA
x_1	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375
x_3	41.9768	42.098446	42.098446	42.098446	41.811296	42.092641
x_4	182.2845	176.636052	176.636596	176.636596	180.260978	176.716452
g_1	−0.0023	0	0	0	-0.005542	-0.000112
g_2	−0.0370	−0.035881	0.03588	−0.035881	-0.038620	-0.035936
g_3	−23420.5966	0	0	−0.000226	-183.1446	-44.05677
g_4	−57.7155	−63.363948	−63.363404	−63.363404	-59.73902	-63.28355
$f(x)$	6171	6059.70161	6059.7143	6059.714339	6103.41749	6068.357986

Table 4.9: **Parameter and constraint values of the best solutions obtained by *r*BA and others algorithm for Tension Spring problem**

	SCA	PSO	(μ+λ)ES	ABC	BA	*r*BA
x_1	0.0521602	0.05169	0.052836	0.051749	0.052312	0.051460
x_2	0.368159	0.35675	0.384942	0.358179	0.371889	0.351232
x_3	10.648442	11.287126	9.807729	11.203763	10.452475	11.619171
g_1	0	0	−0.000001	−0.000000	-1.91E-05	-8.40E-05
g_2	0	0	0	−0.000000	-2.44E-05	-1.65E-05
g_3	−4.075805	−4.053827	−4.106146	−4.056663	-4.082842	-4.042626
g_4	−0.719787	−0.727706	−0.708148	−0.726713	-0.717198	-0.731538
$f(x)$	0.012669	0.012665	0.012689	0.012665	0.012673	0.012668

Table 4.10: **Parameter and constraint values of the best solutions obtained by *r*BA and others algorithm for Speed Reducer problem**

	SCA	(μ+λ)ES	ABC	BA	*r*BA
x_1	3.5	3.499999	3.499999	3.501481	3.500848
x_2	0.7	0.699999	0.7	0.700124	0.700011
x_3	17	17	17	17	17
x_4	7.327602	7.3	7.3	7.388378	7.321624
x_5	7.715321	7.8	7.8	7.844692	7.839082
x_6	3.350267	3.350215	3.350215	3.350535	3.350479
x_7	5.286655	5.286683	5.2878	5.287016	5.286741
g_1	−0.073915	−0.073915	−0.073915	-0.074636	-0.074169
g_2	−0.197999	−0.197998	−0.197999	-0.198623	-0.198218
g_3	−0.493501	−0.499172	−0.499172	-0.969473	-9.70E-01
g_4	−0.904644	−0.901472	−0.901555	-0.994107	-0.994117
g_5	0	0	0	-0.000141	-2.01E-04
g_6	0.000633	0	0	-0.000181	-2.50E-05
g_7	−0.7025	−0.702500	−0.7025	-0.702447	-0.702495
g_8	0	0	0	-0.000245	-0.000226
g_9	−0.583333	−0.583333	−0.583333	-0.583231	-0.583239
g_{10}	−0.054889	−0.051325	−0.051326	-0.062608	-0.054073
g_{11}	0	−0.010852	−0.010695	-0.016441	-0.015776
$f(x)$	2994.744241	2996.348094	2997.058412	2999.547093	2997.884811

4.4 Summary

This chapter has introduced an improved version of Bees Algorithm, known as recombination-based Bees Algorithm (*r*BA), specifically applied for unconstrained numerical benchmark functions and constrained mechanical design problems. In this proposed Bees Algorithm, a technique called recombination operation was used during the local search and between two best abandoned sites, producing a solution closer to the local peak. Experimental results have shown that the Bees Algorithm with recombination operator helped local search produces solution closer to the local peak. This allows the proposed Bees Algorithm to find more accurate solutions with less function evaluations as significant test indicated that the *r*BA outperforms the standard version of Bees Algorithm and other algorithms on most of the problems applied.

CHAPTER 5

A Guided Global Best Bees Algorithm

5.1 Preliminaries

The main disadvantage of the Bees Algorithm compared to other algorithms is the large number of parameters. In general, large number of parameters requires excessive parameter tuning to obtain a good result. Therefore, this chapter proposes a new implementation to reduce the number of parameters by using an adaptive number of recruit bees for each selected site. In addition to the proposed strategy mentioned above, the proposed Bees Algorithm in this chapter also makes use of the best solution found so far to guide the neighbourhood shrinking strategy for the unimproved sites.

The first section of this chapter describes the proposed Bees Algorithm in detail. In this first section, detailed description of the proposed strategies is given followed by experimental set up and experimental results on a set of benchmark functions. It will then go on to the application of the proposed algorithm on constrained mechanical design applications for the next section. Finally, the last section of this chapter concludes and reviews the work done.

5.2 Self Adaptive Bees Recruitment Mechanism

Parameter tuning is one of the most common procedures for finding good solutions in any optimisation algorithms. Usually, these parameters need to be tuned manually by the users for different types of optimisation problems until the best solution is found. Similarly, like other optimisation algorithms, the Bees Algorithm also requires a set of parameters to be tuned for finding the best solution. Although several studies were introduced to overcome this issue on the Bees Algorithm as described in Section 2.7, it remains as the main issue to be solved because lesser parameters would make the algorithm simpler and consume less time for parameter tuning. It is also desirable to have lesser parameters without deteriorating the performance of the Bees Algorithm.

In order to reduce the number of parameters, several rules on selection of best sites and how to send recruited bees were introduced into the standard Bees Algorithm thus resulting in elimination of some of the parameters. These rules were applied to the best selected sites where instead of fully ranking it according to fitness value only, the order of these best sites and how recruit bees are sent goes to the following rules.

i. The first top ranked best sites should be best sites discovered by the global search. If there is more than one site, the sites discovered by the global search should be ranked according to fitness value.
ii. The second top ranked best sites should be the improved best sites from the previous iteration. If there is more than one site, the improved sites should be ranked according to fitness value.
iii. The remaining best sites are the unimproved best sites from the previous iteration. If there is more than one site, the unimproved sites should be ranked according to fitness value.
iv. For the number of recruited bees being sent to each best site, it should be linearly dependant on the rank of the best site.

Overall, these new rules reduce the number of parameters from seven parameters to six parameters as it eliminates the number of elite sites and number of recruited bees to be sent to both elite site(s) and best sites. It is also more like foraging behaviour in nature in terms of selecting promising sites. However, the user still needs to set the total number of recruited bees to be sent to each best site. The total Bees Algorithm parameters after applying these rules are:

i. ***ns*** – number scout bees,
ii. ***nb*** – number of best sites,
iii. ***nrb_t*** – total number of recruit bees for selected sites,
iv. ***ngh*** – neighbourhood size,
v. ***stlim*** – stagnation limit.

5.3 Guided Neighbourhood Shrinking Strategy

In addition to the new introduced rules described above, another strategy called guided neighbourhood shrinking based on global best solution found so far is proposed. This proposed strategy uses the information from the global best solution found so far to guide the neighbourhood shrinking for the unimproved sites for better exploitation capabilities. Based on the assumption that the global best solution found so far often leads to optimal solutions, the neighbourhood shrinking bias toward the global best solution found so far on certain conditions is believed to guide the local search in finding the optimum solution. This proposed Bees Algorithm is named as global best guided Bees Algorithm (*g*BA).

The flowchart of the proposed *g*BA is illustrated in Figure 5.1 where index j is the index of dimension D, j:1,2,3,…,D. For the standard Bees Algorithm, the neighbourhood size shrinks according to Eq. 2.2 whereas for this proposed strategy, the position of the site being search is compared against the best solution found so far before proceeding to the next steps. The proposed algorithm starts once the neighbourhood search failed to find better fitness value, the global best solution found so far has better fitness value than the fittest recruited bee, and at least one index position of fittest recruited bee is within the neighbourhood of best found so far. If those three conditions are satisfied, the position of the fittest recruited bee at each dimension is checked. The neighbourhood size for position at particular dimensions which is not within the neighbourhood of the best solution found so far, it is set to a new neighbourhood size; distance between best recruited bee and best solution found at corresponding dimension ($P_{Global,j,}$-$P_{best,j,}$). For position at a dimension that is within the neighbourhood of the best solution found so far follows the standard neighbourhood shrinking procedure. After all the neighbourhood dimension sizes have been updated, the recruited bees are sent to the new defined patch.

To explain how the proposed guided global best Bees Algorithm works, the following simple 2D examples shown in Figures 5.2–5.4 are used. In this example, it is assumed that the Best bee (x_B) failed to find better fitness value and the Best solution found so far (x_G) has better fitness value than the Best bee (x_B). Figure 5.2 shows ngh_{x1} is not within the neighbourhood range of best solution so far whereas ngh_{x2} is within the neighbourhood range of best solution found so far. Therefore, a new neighbourhood size of ngh_{x1} for the Best bee (x_B) is set with the distance between the Best bee (x_B) and Best solution found so far (x_G) corresponding to x_1. However, the neighbourhood size for ngh_{x2} shrinks similar as the standard Bees Algorithm procedure. These new neighbourhood sizes which follow the procedures mentioned earlier are shown in Figure 5.3. Once both neighbourhood sizes have been updated, the recruited bees are sent to this site accordingly as described in Figure 5.4.

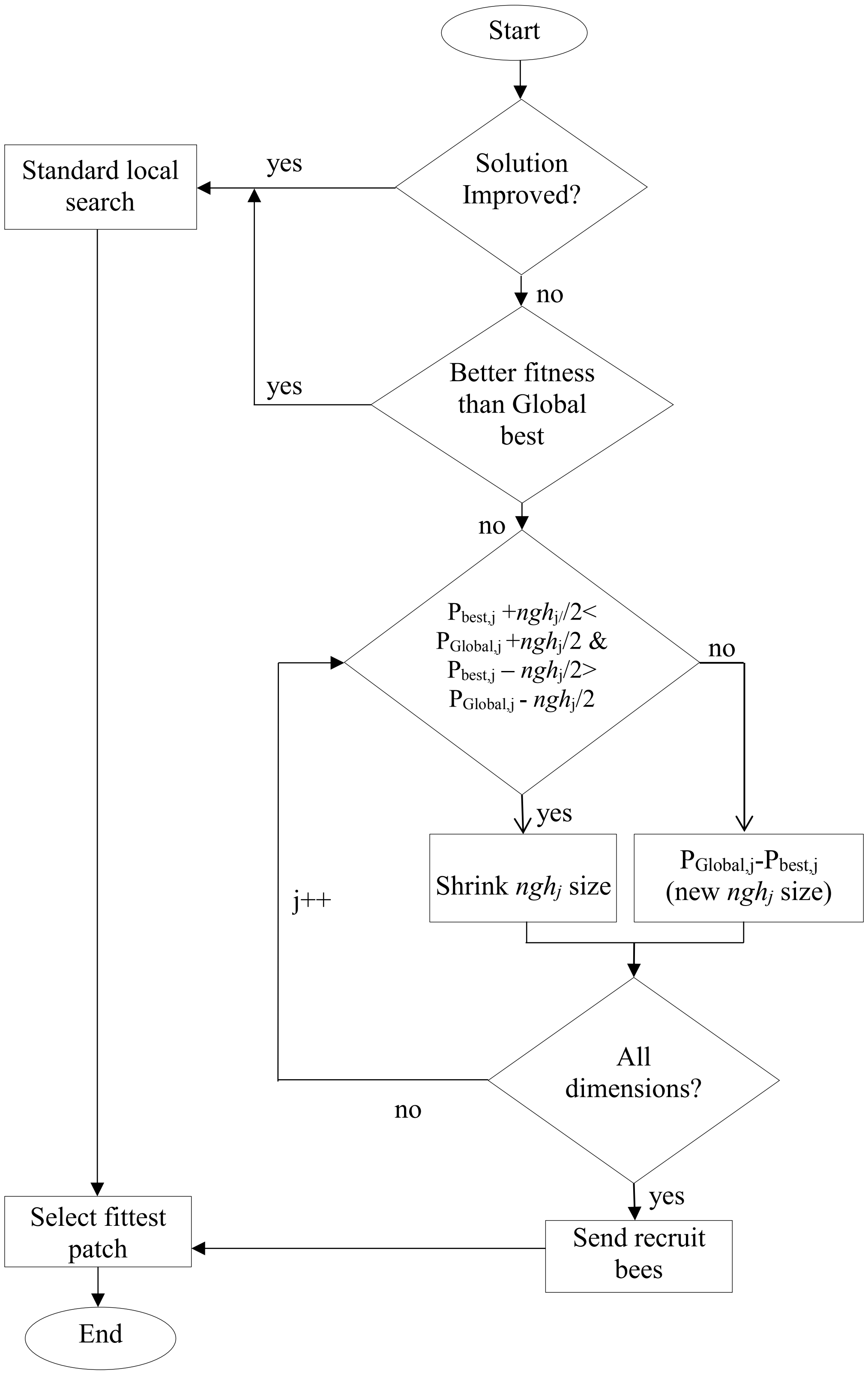
Start
Solution Improved?
yes
Standard local search
no
Better fitness than Global best
yes
no
$P_{best,j} + ngh_j/2 < P_{Global,j} + ngh_j/2$ & $P_{best,j} - ngh_j/2 > P_{Global,j} - ngh_j/2$
no
yes
Shrink ngh_j size
$P_{Global,j} - P_{best,j}$ (new ngh_j size)
j++
All dimensions?
no
yes
Select fittest patch
Send recruit bees
End

Figure 5.1: Flowchart of the guided neighbourhood shrinking strategy for unimproved sites

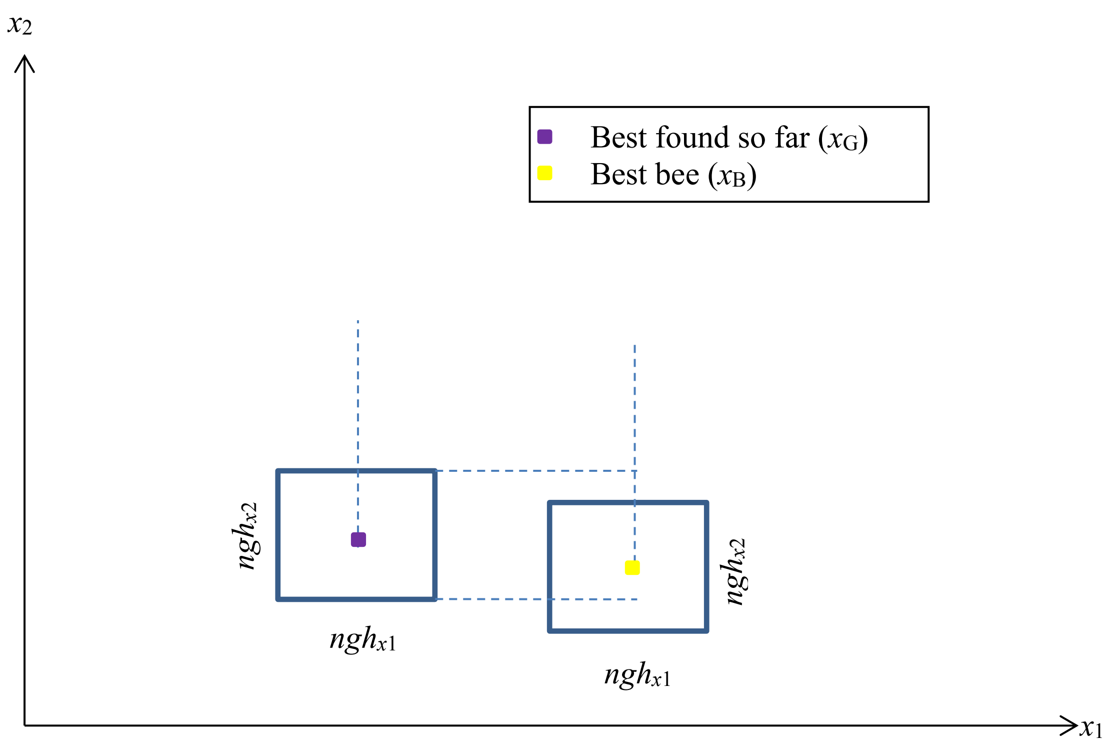

Figure 5.2 Illustration of guided neighbourhood shrinking strategy for unimproved site – satisfy the condition

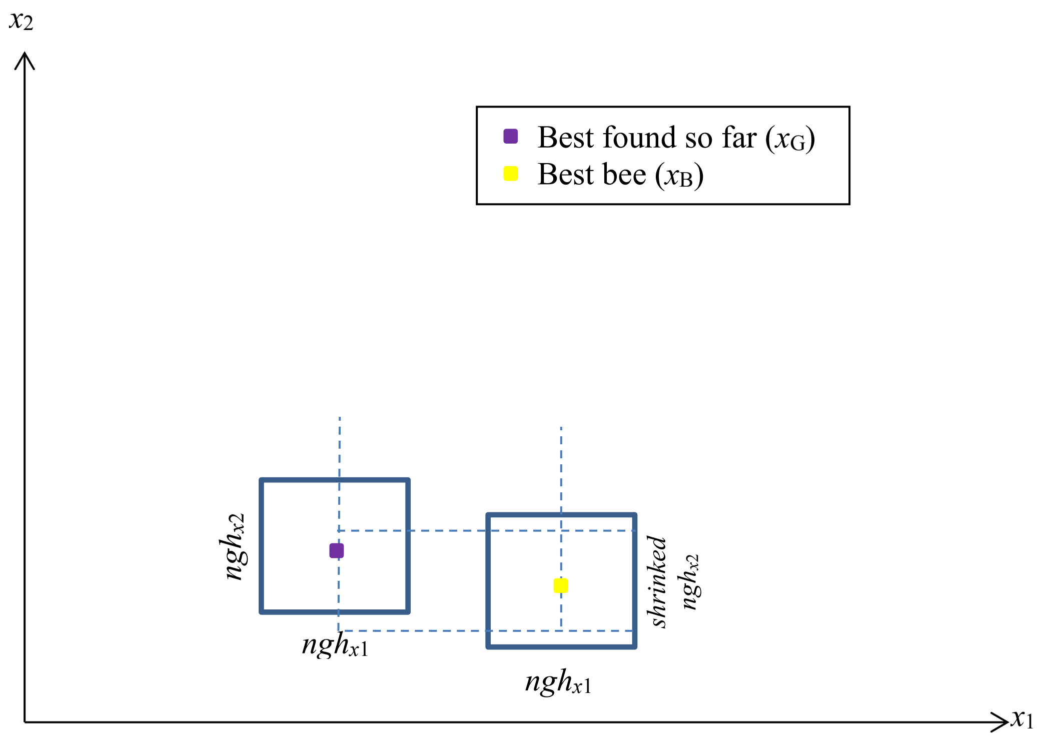

Figure 5.3 Illustration of guided neighbourhood shrinking strategy for unimproved site – set ***ngh*** sizes

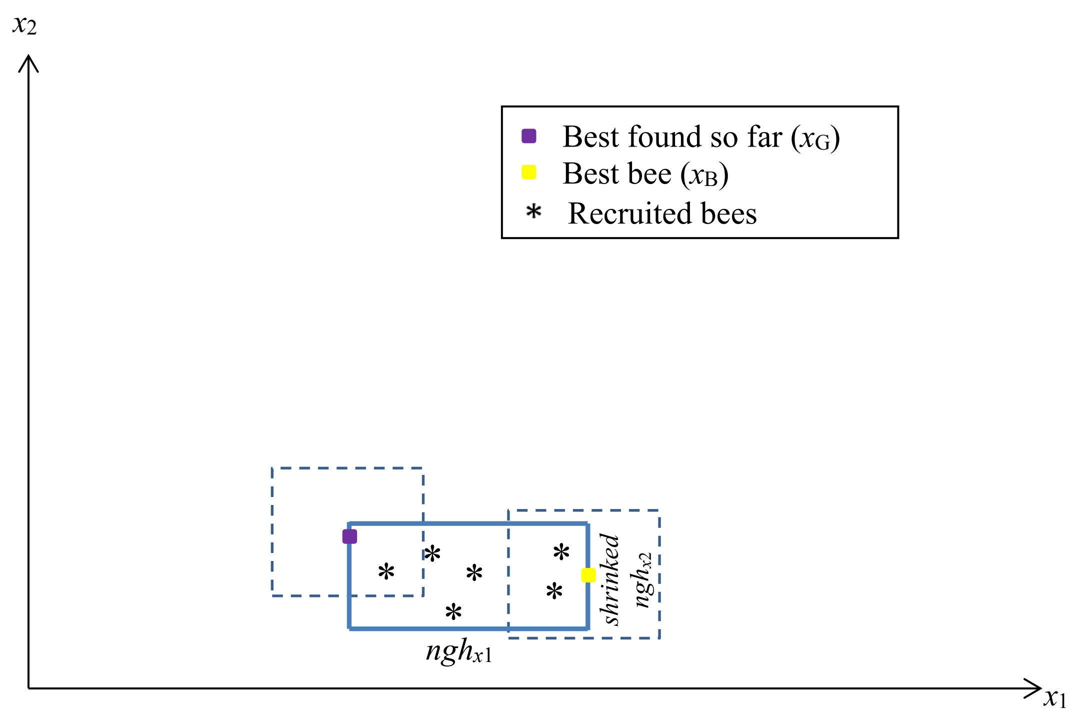

Figure 5.4 Illustration of guided neighbourhood shrinking strategy for unimproved site – send recruited bees

5.4 Benchmark Functions Experiment on *g*BA

In general, similar experimental set up as in Section 3.2.1 and Section 4.2.1 were utilised in this chapter. This study used a similar set of benchmark functions (see Appendix A), as well as the parameters settings (Table 3.1) and stopping criterion as described in Section 3.2.1. Similar method of comparison with the standard Bees Algorithm, SPSO2011 and qABC were also utilised in this study as in Chapter 3 and Chapter 4.

5.4.1 Experimental Results

In this experiment, each benchmark function was run independently 100 times with the standard Bees Algorithm, *g*BA, SPSO2011 and qABC algorithm. The results found by each algorithm in terms of accuracy (global optimum - best solution found) and speeds (function evaluations reached) are given in Table 5.1 and Table 5.2 respectively. Table 5.1 shows the median and standard deviation of accuracy results over 100 runs for the four algorithms. For median accuracy that obtained less than 0.001 in the experimental result, the value recorded in Table 5.1 is 0.0000 to make the comparison simpler.

Table 5.1 shows that the *g*BA obtained a median accuracy of less than 0.001 with 100% success rate on most of the functions except on the *Rosenbrock*, *Rastrigin*, *Shekel_10D, Griewank*, and *Langermann* functions. For the *Shekel_10D* function, although it did not achieve 100% success rate, it still managed to find accuracy less than 0.001 with 71% success rate. Meanwhile, with the exception of *Shekel_10D, Michaelewicz,* and non-successful functions of *g*BA, the standard Bees Algorithm accomplished 100% success rate on a similar set of functions.

For SPSO2011, the algorithm attained 100% success rate in finding the minimum on *Easom, Trid, Zakharov, Schaffer, Schwefel, Goldstein & Price, Martin & Gaddy,* and *Camel Six Hump* (eight functions) while on the *Shekel_4D* function, the SPSO2011 succeeded to find the threshold accuracy at 59% success rate. Furthermore, Table 5.1 also shows that qABC found a median accuracy of less than 0.001 on *Easom, Schaffer, Rastrigin, Schwefel, Michaelewicz, Goldstein & Price, Martin & Gaddy, Camel Six Hump* and *Shekel_4D* (nine functions).

Next, Table 5.2 shows a comparison of median and standard deviation of function evaluations required to achieve accuracy presented in Table 5.1. From these results, for the functions that achieved accuracy of less than 0.001; five functions converged faster than the standard Bees Algorithm, six functions are comparable in terms of convergence speed as the Mann Whitney Test (see Table 5.3) showed no significant difference on these results. Meanwhile, for the remaining four functions where the *g*BA could not achieve the accuracy of less than 0.001 and reached maximum function evaluations, the performance is considered comparable with the standard Bees Algorithm in terms of function evaluations.

Table 5.1: **Comparison on accuracy over 100 runs for *g*BA**

Function	SBA		*g*BA		SPSO2011		qABC	
	Median	**StdDev**	**Median**	**StdDev**	**Median**	**StdDev**	**Median**	**StdDev**
Easom	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Trid	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.30155	0.3062
Rosenbrock	4.8919	1.0673	3.3271	1.3973	6.9007	9.4754	**0.0758**	**0.0702**
Zakharov	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0235	0.0867
Schaffer	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0010**
Rastrigin	12.9437	3.2667	0.0108	0.4498	6.4253	2.2037	**0.0000**	**0.0219**
Schwefel	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Michaelewicz	**0.0000**	**0.0133**	**0.0000**	**0.0000**	0.0411	0.0895	**0.0000**	**0.0000**
Goldstein & Price	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0003**	**0.0000**	**0.0000**
Martin & Gaddy	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Camel Six Hump	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Shekel_4D	**0.0000**	**0.0002**	**0.0000**	**0.0000**	**0.0000**	**2.5068**	**0.0000**	**0.0001**
Shekel_10D	0.0015	2.6435	**0.0006**	**1.8110**	8.7202	0.2236	8.7231	0.0036
Griewank	0.1091	0.0226	0.0663	0.0252	**0.0341**	**0.0271**	0.0800	0.0389
Langermann	1.2563	0.0989	**0.9323**	**0.1794**	0.9676	0.1363	1.1713	0.0916

Table 5.2: **Comparison on function evaluations over 100 runs for *g*BA**

Function	SBA		*g*BA		SPSO2011		qABC	
	Median	**StdDev**	**Median**	**StdDev**	**Median**	**StdDev**	**Median**	**StdDev**
Easom	3124	1408.923	2424	727.3015	4000	885.2301	**1050**	**6888.015**
Trid	13224	4255.14	12724	9430.22	**9100**	**700.657**	500000	0
Rosenbrock	**500000**	**0**	**500000**	**0**	**500000**	**117878**	**500000**	**0**
Zakharov	**16374**	**1896.454**	16424	1760.32	35100	4619.083	500000	0
Schaffer	5305.5	2869.263	**1824**	**1166.363**	2800	1728.957	43951	225352.2
Rastrigin	500000	0	500000	0	500000	0	**352561.5**	**97711.63**
Schwefel	2825	100.508	2624	1127.964	14100	30860.92	**850**	**12132.96**
Michaelewicz	163807.5	155525.4	**15029**	**17257.13**	500000	223339.3	105399	88612.96
Goldstein & Price	**1824**	**729.9545**	**1824**	**565.2424**	4000	501.387	2350	35168.9
Martin & Gaddy	1464	732.9105	**1224**	**568.7697**	2000	459.1725	15350	204091.6
Camel Six Hump	924	315.7083	924	340.1411	3031	581.481	**850**	**405.292**
Shekel_4D	10628	8756.234	**9228**	**6329.679**	82500	230204.6	37850	65029.46
Shekel_10D	500000	168591.4	**324926.2**	**149701**	500000	0	500000	0
Griewank	**500000**	**0**	**500000**	**0**	**500000**	**105560.5**	**500000**	**33908.05**
Langermann	**500000**	**0**	**500000**	**0**	**500000**	**0**	**500000**	**0**

With respect to comparison with other algorithms, it is observed that the proposed algorithm converged faster than the SPSO2011 algorithm for the *Easom*, *Zakharov*, *Schaffe*r, *Schwefel*, *Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, *Camel Six Hump*, *Shekel_4D*, and *Shekel_10D* functions. As for comparison with qABC, the *g*BA outperformed the qABC in terms of convergence speed for the *Trid*, *Zakharov*, *Schaffe*r, *Michaelewicz*, *Goldstein & Price*, *Martin & Gaddy*, *Shekel_4D*, and *Shekel_10D* functions (eight functions) while one function showed comparable convergence speed (*Camel Six Hump* function) as the *p*-value is more than 0.05 (See Table 5.4). For the rest of the functions, the qABC results obtained 500000 maximum function evaluations.

Furthermore, a comparison of overall performance is also made with other algorithms. Table 5.5 summarises the overall performance results achieved in Table 5.1 and Table 5.2. These results show that, the performance of *g*BA surpassed other algorithms on nine out of fifteen functions. From those nine functions, six functions (*Trid*, *Zakharov*, *Schwefe*l, *Goldstein & Price*, *Camel Six Hump* and *Shekel_4D*) showed equal performance compared to the standard Bees Algorithm.

For comparison of *g*BA with SPSO2011, SPSO2011 demonstrates best overall performance over *g*BA only on one unimodal function (*Trid*). Besides, these results also show that qABC performed better than *g*BA only on four functions (*Easom*, *Rosenbrock*, *Rastrigin*, *Schwefel*) while one functions (*Camel Six Humps*) showed equal performance with *p*-value of 0.5552. The results in this section indicate that the guided global best Bees Algorithm is effective in improving the performance of the standard Bees Algorithm, especially high dimensional multimodal problems. The next section moves on to discuss the results reported in this section.

Table 5.3: **Significant difference of *g*BA's median results against standard Bees Algorithm**

	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000
Trid	ns	-	ns	0.5222
Rosenbrock	s	0.0000	ns	-
Zakharov	ns	-	ns	0.5823
Schaffer	ns	-	s	0.0000
Rastrigin	s	0.0000	ns	-
Schwefel	ns	-	ns	0.1336
Michaelewicz	ns	-	s	0.0000
Goldstein & Price	ns	-	ns	0.5552
Martin & Gaddy	ns	-	s	0.0003
Camel Six Hump	ns	-	ns	0.5552
Shekel_4D	ns	-	ns	0.3735
Shekel_10D	s	0.0000	s	0.0041
Griewank	s	0.0000	ns	-
Langermann	s	0.0000	ns	-

s: statistically significant, ns: not significant
'-': No statistical difference test required

Table 5.4: **Significant difference of *g*BA's median results against other algorithms**

	vs SPSO2011				vs qABC			
	accuracy	*p*-value	speed	*p*-value	accuracy	*p*-value	speed	*p*-value
Easom	ns	-	s	0.0000	ns	-	s	0.0000
Trid	ns	-	s	0.0000	s	0.0000	s	0.0000
Rosenbrock	s	0.0308	ns	-	s	0.0000	ns	-
Zakharov	ns	-	s	0.0000	s	0.0000	s	0.0000
Schaffer	ns	-	s	0.0000	ns	-	s	0.0000
Rastrigin	s	0.0000	ns	-	s	0.0000	s	0.0000
Schwefel	ns	-	s	0.0000	ns	-	s	0.0000
Michaelewicz	s	0.0003	s	0.0000	ns	-	s	0.0000
Goldstein & Price	ns	-	s	0.0000	ns	-	s	0.0147
Martin & Gaddy	ns	-	s	0.0000	ns	--	s	0.0147
Camel Six Hump	ns	-	s	0.0000	ns	0.9045	ns	0.1416
Shekel_4D	ns	-	s	0.0000	ns	-	s	0.0000
Shekel_10D	s	0.0000	s	0.0000	s	0.0000	s	0.0000
Griewank	s	0.0000	ns	-	s	0.0000	ns	-
Langermann	s	0.0078	ns	-	s	0.0000	ns	-

s: statistically significant, ns: not significant
'-': No statistical difference test required

5.4.2 Discussion

Table 5.5: **Comparison of overall performance for *g*BA**

Function	**SBA**			***g*BA**			**SPSO2011**			**qABC**		
	succ.	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**	**succ.**	**acc.**	**perf.**
Easom	100	X		100	X		100	X		100	X	X
Trid	100	X		100	X		100	X	X	0		
Rosenbrock	0			0			21	X		0	X	X
Zakharov	100	X	X	100	X	X	100	X		0		
Schaffer	100	X		100	X	X	100	X		68	X	
Rastrigin	0			0			0			88	X	X
Schwefel	100	X		100	X		100	X		100	X	X
Michaelewicz	88	X		100	X	X	39			100	X	
Goldstein & Price	100	X	X	100	X	X	100	X		100	X	
Martin & Gaddy	100	X		100	X	X	100	X		79	X	
Camel Six Hump	100	X	X	100	X	X	100	X		100	X	X
Shekel_4D	100	X	X	100	X	X	59	X		99	X	
Shekel_10D	41			71	X	X	0			0		
Griewank	0			0			6	X		2		
Langermann	0			0	X	X	2			0		
Total	1029		4	1071		9	927		1	836		4

succ.: successful rate, acc.: accuracy achieved, perf.: overall performance

The findings reported in the previous section demonstrate the advantages of adapting the best solution found so far in order to guide the neighbourhood shrinking for unimproved sites. According to the experimental results obtained in Section 5.4.1, the *g*BA achieved the best overall performance over the standard Bees Algorithm on six multimodal functions (*Schaffer*, *Rastrigin*, *Michaelewicz*, *Shekel_10D*, *Griewank*, and *Langermann*) and three unimodal functions (*Easom*, *Rosenbrock*, and *Martin & Gaddy*). However, the *g*BA performs similarly on four multimodal functions (*Schwefel*, *Goldstein & Price*, *Camel Six Hump*, and *Shekel_4D*) and two unimodal functions (*Trid* and *Zakharov*).

The comparison of results mentioned earlier, demonstrates the effectiveness of the *g*BA dealing with high dimension multimodal functions, such as *Rastrigin*, *Michaelewicz*, *Shekel_10D*, *Griewank*, and *Langermann.* Besides, the *g*BA is also performed fairly effective on the low dimension multimodal function (*Schaffer*).

For the *Rastrigin* and *Griewank* functions, despite the *g*BA not converging, the solutions found are closer to the global optimum. The plot of convergence for the *Rastrigin* function is shown in Figure 5.5, where a better final solution found by the *g*BA. This result is likely because both functions have a rough multimodal surface with unimodal behaviour overall. The standard Bees Algorithm could be easily trapped inside many local optima on this type of function during the neighbourhood shrinking phase. Thus, identifying which neighbourhood size to reduce at the selected index based on the best solution found so far would be an advantage for the function with the overall unimodal behaviour especially at a higher dimension.

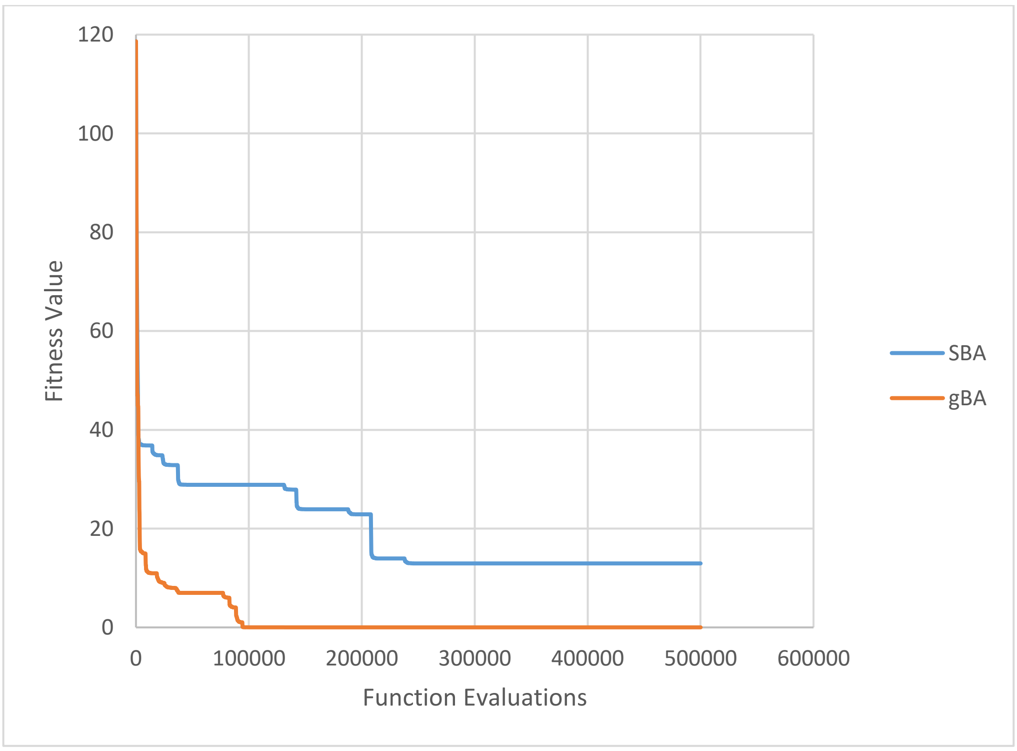

Figure 5.5: Plot of convergence for the *Rastrigin* function

As for the *Shekel_10D* and *Langermann* functions, the *g*BA also shows a better performance in terms of the solutions found. This indicates that reducing the neighbourhood size at the selected index dimension together with new a defined neighbourhood size at the index dimension that satisfied the condition, is beneficial for this type of problem.

For *Michaelewicz*, the valley and ridges of the function could slow down the convergence speed of the standard Bees Algorithm if the bees are placed along the valley and ridges. Thus, adding the guiding neighbourhood shrinking strategy to the standard Bees Algorithm might help the algorithm to escape from those valley and ridges. This behaviour is shown in Figure 5.6, where a plot of convergence for the *Michaelewicz* function clearly demonstrates the ability of the *g*BA to overcome this type of surface landscape. However, this strategy is fairly effective in improving the Bees Algorithm on most low dimension multimodal types of functions, probably because the guided neighbourhood shrinking did not fully take place as the conditions were not satisfied. In this cases, the *g*BA significantly converged faster than the standard Bees Algorithm for the *Schaffer* function.

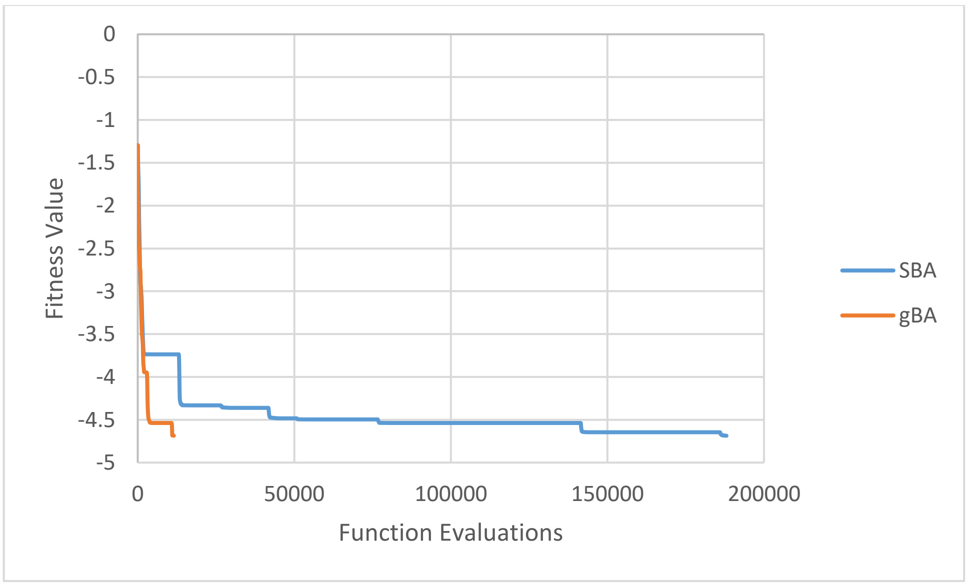

Figure 5.6: Plot of convergence for the *Michaelewicz* function

Regarding the equal performance of the *g*BA with the standard Bees Algorithm on six functions (*Trid*, *Zakharov*, *Schwefel*, *Goldstein & Price*, *Camel Six Hump* and *Shekel_4D*), both algorithms accomplished a 100% success rate but no significant difference in terms of the function evaluations on these functions. These results demonstrate that the proposed strategy could not improve the Bees Algorithm on these functions, which are considered to be easy multimodal functions and unimodal functions. For this reason, the neighbourhood search encounter with unimproved sites is less frequent as the search usually find better solutions. Figure 5.7 illustrates an example of the optimisation progress for the *Camel Six Hump* function. This figure indicates that the *g*BA and the standard Bees Algorithm progressively find better solutions during most of the search progress. Therefore, no significant improvement was observed on these functions.

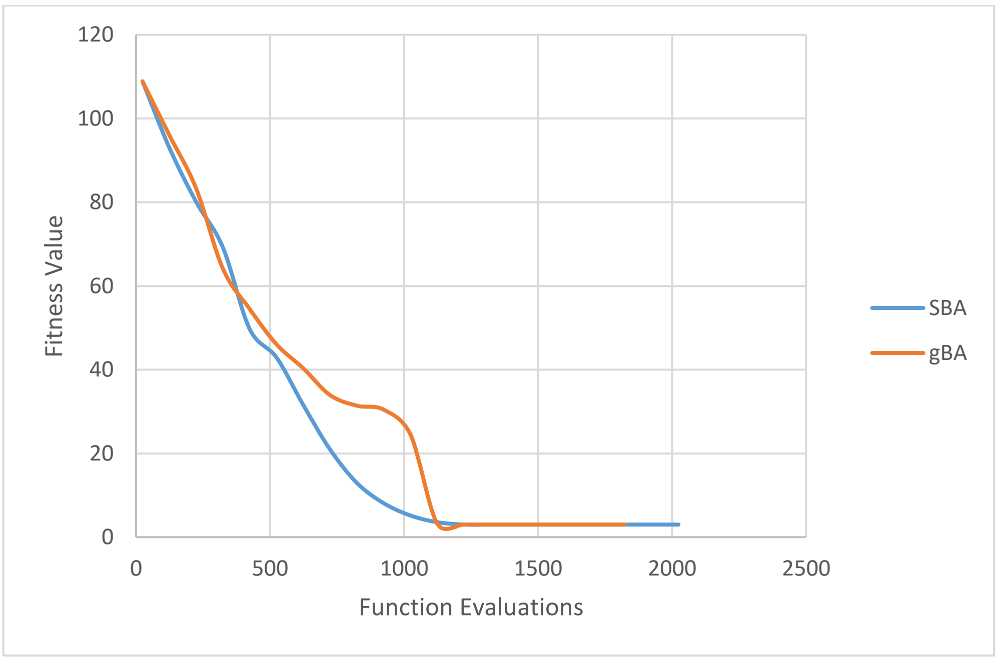

Figure 5.7: Plot of convergence for the *Camel Six Hump* function

When drawing comparison with SPSO2011, where the performance of SPSO2011 is better than of the *g*BA, the results in Table 5.5 show that both algorithms achieved a 100% success rate on the *Trid* function but SPSO2011 required fewer function evaluations. A probable explanation is that the search mechanism of the SPSO2011 algorithm has an advantage over the algorithms based on the foraging behaviour for this type of function.

Furthermore, for comparison purposes with the qABC algorithm, the result in Table 5.5 shows that the *g*BA did not perform as well as the qABC on the *Easom*, *Rosenbrock*, *Rastrigin,* and *Schwefel* functions. On the *Easom* and *Schwefel*, functions, both algorithms achieved a 100% success rate but the qABC achieved this with a higher convergence speed. For the *Rosenbrock* function, even though the qABC failed to find the global optimum, the median solution obtained is closer to the global optimum compared to the *g*BA and the difference is significant. This result indicates that the qABC outperformed the *g*BA on the *Rosenbrock* function. On the *Rastrigin* function, the performance of the qABC excelled in finding the global optimum with an 88% success rate, whereas the *g*BA could not locate the global optimum. Even though the performance of the gBA on some of those functions is not as good as in the case of the SPSO2011 or qABC, the superiority of the gBA on a large number of functions shows that the proposed Bees Algorithm is better than other algorithms.

5.5 Mechanical Design Applications

The previous section has demonstrated the performance of *g*BA on a set of unconstrained numerical benchmark functions. It is now necessary to further evaluate the performance of the proposed Bees Algorithm on constrained optimisation problems. For this reason, four well known mechanical design problems were selected: Welded Beam, Pressure Vessel, Tension/Compression Spring and Speed Reducer (Appendix C). The parameter settings used in this experiment is the same as the ones given in Table 3.7 of section 3.3. This experiment also used 30000 function evaluations for stopping criterion and running the algorithm 30 times on each problem. With regards to handling constrained problems, a similar method as used in section 3.3 was adopted in this experiment.

In addition to the problems mentioned above, the *g*BA was applied to another mechanical design problem from the literature known as the Multiple Clutch problem (Rao et al., 2011). The details of the problem are given in Appendix C. The parameter settings used for this problem are given in Table 5.6. For this problem, the maximum function evaluations and number of runs are the same as those used by Rao et al. (2011).

Table 5.6: **Parameter settings for Multiple Clutch Problem**

Parameter	***ns***	***ne***	***nb***	***nre***	***nrb***	***ngh***	***stlim***
Value	10	2	5	10	5	2	10

Table 5.7 and Table 5.8 provide the results of *g*BA obtained from the experiment on problems mentioned above and results of other algorithms. For each problem, the best solution, mean solution and median solution over number of runs were recorded in those tables. As for the results obtained on the Multiple Clutch problem, the number of successful rate (SR) was also recorded. The performance of *g*BA is compared with the standard Bees Algorithm based on the median value. An algorithm is said to be better than another if the median value found is less than another and the difference is significance. If the median value difference is insignificant, both algorithms are said to be comparable. In this experiment, the Mann Whitney Test (at $\alpha = 0.05$) was used to check the significant difference of results between two algorithms.

According to the results reported in Table 5.7 and Table 5.8, the median values found by *g*BA are less than the standard Bees Algorithm and the differences are significant for the Welded Beam, Tension Spring and Multiple Clutch problems. These results indicate the superiority of *g*BA over the standard Bees Algorithm on those problems. For the Speed Reducer problem, the difference between the median value of *g*BA and standard Bees Algorithm are insignificant. For this reason, both algorithms are said to be comparable on this problem. However, for the Pressure Vessel problem, the results showed that the median value obtained by the standard Bees Algorithm is less than *g*BA and the difference is significant. The significant difference for all problems is illustrated in Figures 5.8–5.12. Surprisingly, the addition of the proposed strategy into the standard Bees Algorithm that was meant to improve or at least equal in general performance, showed a performance that was worse on this Pressure Vessel problem.

Table 5.7 **Comparison of *g*BA against others** (Akay and Karaboga , 2012)

Problem	Stats.	Sca[a]	Pso[a]	(μ+λ)-ES	UPSOm	ABC	BA	*g*BA
Welded Beam	Best	NA	NA	1.724852	1.92199	1.724852	1.733247	1.728407
	Mean	NA	NA	1.777692	2.83721	1.741913	1.767923	1.749410
	Median	NA	NA	NA	NA	NA	1.766979	1.745503
	NFE	NA	NA	30,000	100,000	30,000	30000	30000
Pressure Vessel	Best	6171	6059.7143	6059.70161	6544.27	6059.71474	6103.4175	6068.23
	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.3081	6381.4714	7583.8873
	Median	NA	NA	NA	NA	NA	6315.1676	6759.573
	NFE	20000	30000	30000	100000	30000	30000	30000
Tension/ Compresion Spring	Best	0.012669	0.012665	0.012689	0.01312	0.012665	0.012673	0.012666
	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	0.013339	0.012892
	Median	NA	NA	NA	NA	NA	0.013321	0.012762
	NFE	25,167	15000	30000	100000	30000	30000	30000
Speed Reducer	Best	2994.74424	NA	2996.348094	NA	2997.05841	2999.5471	3000.5082
	Mean	3001.758264	NA	2996.34809	NA	2997.05841	3006.3114	3007.2668
	Median	NA	NA	NA	NA	NA	3005.7187	3006.872
	NFE	54,456	NA	30000	NA	30000	30000	30000

NA: Not available, NFE: Number of function evaluations

[a]The welded beam problems are different from the one employed in this study

Table 5.8 **Comparison of *g*BA against others** (Rao et al., 2011) **for Multiple Clutch Problem**

Problem	Stats.	TLBO	ABC	BA	*g*BA
Multiple Clutch Problem	Best	0.313657	0.313657	0.313657	0.313657
	Mean	0.3271662	0.324751	0.328960	0.322020
	Median	NA	NA	0.325419	0.313657
	SR	0.67	0.54	0.08	0.16
	NFE	1000	1000	1,000	1,000

SR: successful rate, NFE: Number of function evaluations, NA: Not available

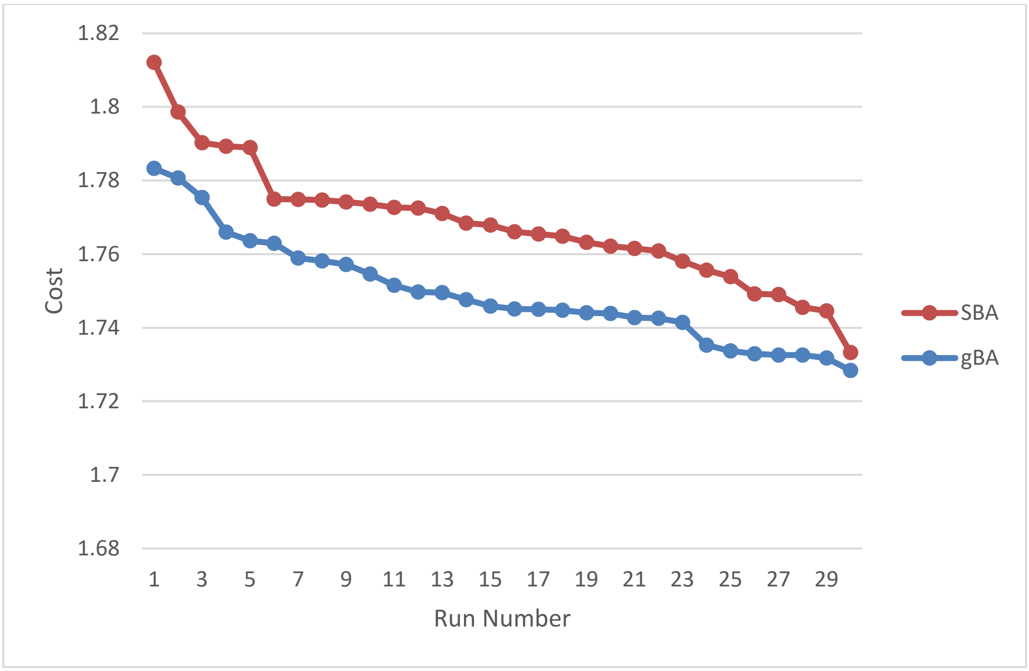

Figure 5.8: Minimum costs found by Standard Bees Algorithm and *g*BA over 30 runs for the Welded Beam problem

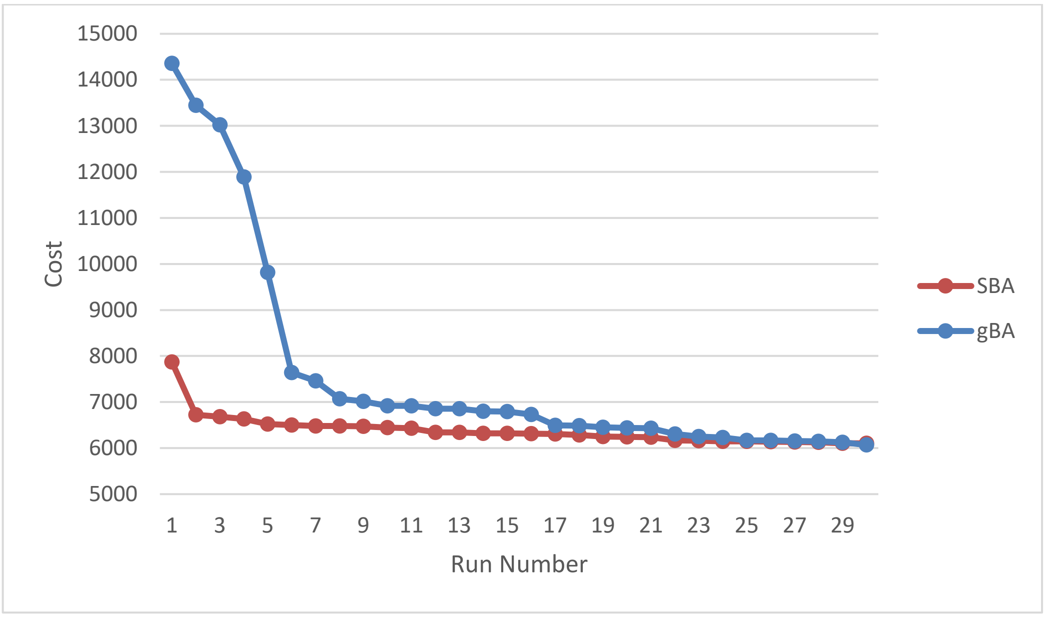

Figure 5.9: Minimum costs found by Standard Bees Algorithm and *g*BA over 30 runs for the Pressure Vessel problem

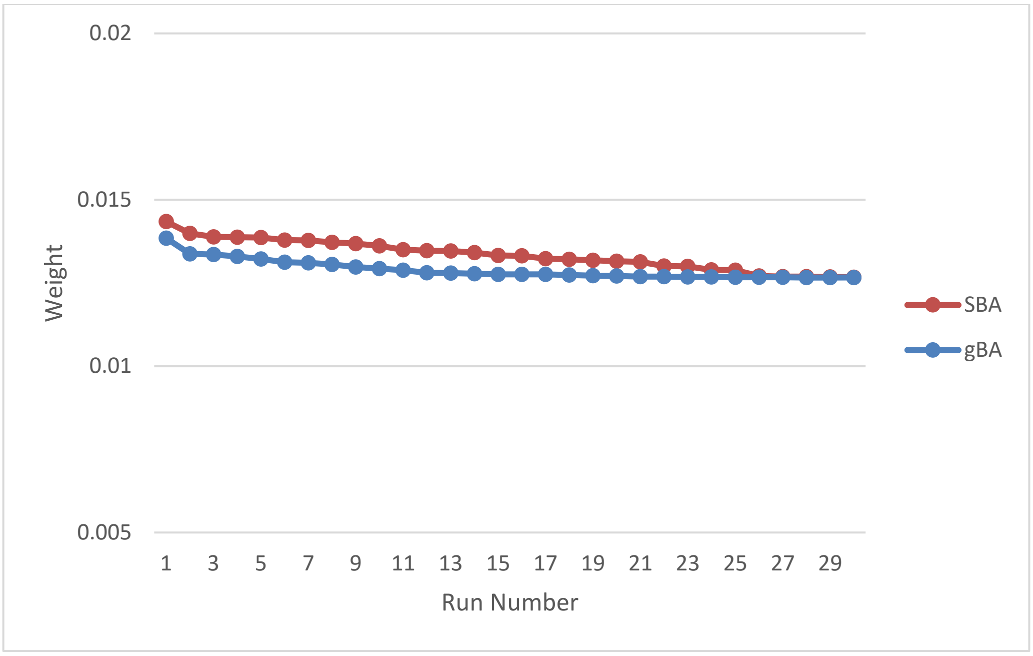

Figure 5.10: Minimum weights found by Standard Bees Algorithm and *g*BA over 30 runs for the Tension/Compression Spring problem

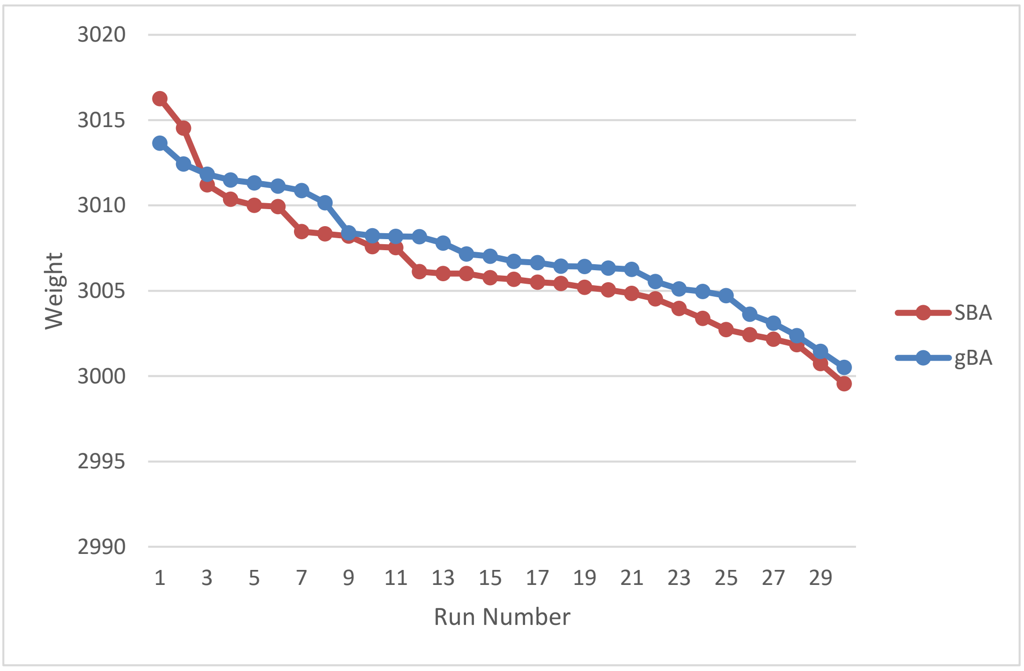

Figure 5.11: Minimum weights found by Standard Bees Algorithm and *g*BA over 30 runs for the Speed Reducer problem

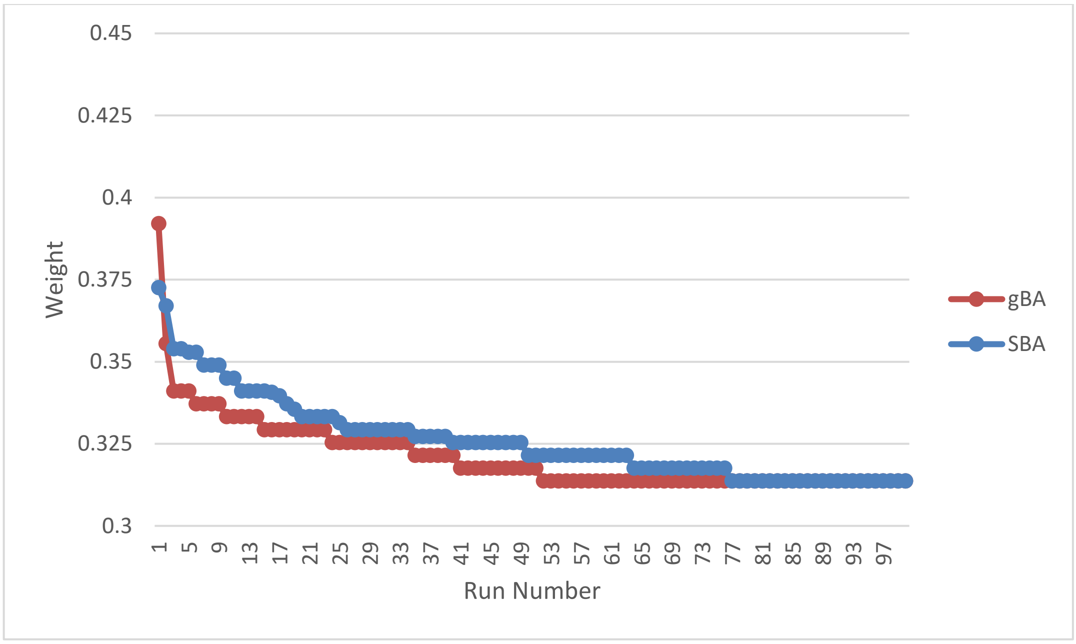

Figure 5.12: Minimum weights found by Standard Bees Algorithm and *g*BA over 100 runs for the Multiple Clutch Problem

In the case of comparison with other algorithms, *g*BA is compared based on the best solution found. The results in Table 5.7 showed that the best solution was found by (μ+λ)-ES (ABC found the same best), (μ+λ)-ES, PSO (ABC found the same best), and Sca for the Welded Beam, Pressure Vessel, Tension Spring and Speed Reducer problems respectively. Generally, the results found by *g*BA are comparable to other algorithms, even though other algorithms found better solution in terms of best solution because those values are relatively close compared to best solution found by others. As for the Multiple Clutch problem, all algorithms found the same best solution. Overall, these results show the benefits of using the global best to guide the neighbourhood shrinking on most of the problems. The design variables value of the best solutions found by *g*BA, standard Bees Algorithm and other algorithms are given in Tables 5.9–5.12.

Table 5.9: **Parameter and constraint values of the best solutions obtained by *g*BA and others for Welded Beam**

	(μ+λ)ES	ABC	BA	*g*BA
x_1	0.20573	0.20573	0.202064	0.204795
x_2	3.470489	3.470489	3.569718	3.499854
x_3	9.036624	9.036624	9.0364	9.040141
x_4	0.205729	0.20573	0.205836	0.205786
g_1	0	0	-55.8952	-32.5063
g_2	0.000002	−0.000002	-14.0521	-31.5751
g_3	0	0	-0.00377	-0.00099
g_4	−3.432984	−3.432984	-3.42349	-3.42936
g_5	−0.080730	−0.080730	-0.07706	-0.07979
g_6	−0.235540	−0.235540	-0.23555	-0.23556
g_7	−0.000001	0	-9.23549	-6.48493
$f(x)$	1.724852	1.724852	1.733247	1.728407

Table 5.10: **Parameter and constraint values of the best solutions Obtained by *g*BA and others for Pressure Vessel**

	SCA	PSO	(μ+λ)ES	ABC	BA	*g*BA
x_1	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375
x_3	41.9768	42.098446	42.098446	42.098446	41.811295	42.095279
x_4	182.2845	176.636052	176.636596	176.636596	180.260978	176.677711
g_1	−0.0023	0	0	0	-0.005542	-5.44E-05
g_2	−0.0370	−0.035881	0.03588	−0.035881	-0.038620	-0.035908
g_3	−23420.5966	0	0	−0.000226	-183.1446	-99.92888
g_4	−57.7155	−63.363948	−63.363404	−63.363404	-59.73902	-63.31052
$f(x)$	6171	6059.70161	6059.7143	6059.714339	6103.417489	6068.229994

Table 5.11: **Parameter and constraint values of the best solutions obtained by *g*BA and others for Tension Spring**

	SCA	PSO	(μ+λ)ES	ABC	BA	*g*BA
x_1	0.0521602	0.05169	0.052836	0.051749	0.052312	0.051413
x_2	0.368159	0.35675	0.384942	0.358179	0.371889	0.350089
x_3	10.648442	11.287126	9.807729	11.203763	10.452475	11.690453
g_1	0	0	−0.000001	−0.000000	-1.91E-05	-1.53E-05
g_2	0	0	0	−0.000000	-2.44E-05	-5.19E-06
g_3	−4.075805	−4.053827	−4.106146	−4.056663	-4.082842	-4.04814
g_4	−0.719787	−0.727706	−0.708148	−0.726713	-0.717198	-0.729777
$f(x)$	0.012669	0.012665	0.012689	0.012665	0.012673	0.012666

Table 5.12: **Parameter and constraint values of the best solutions obtained by *g*BA and others for Speed Reducer**

	SCA	(μ+λ)ES	ABC	BA	*g*BA
x_1	3.5	3.5	3.5	3.50148	3.502
x_2	0.7	0.7	0.7	0.70012	0.70003
x_3	17	17	17	17	17
x_4	7.327602	7.3	7.3	7.38838	7.540459
x_5	7.715321	7.8	7.8	7.84469	7.803575
x_6	3.350267	3.35022	3.35022	3.35054	3.353023
x_7	5.286655	5.28668	5.2878	5.28702	5.287158
g_1	−0.073915	−0.073915	−0.073915	-0.0746	-0.07453
g_2	−0.197999	−0.197998	−0.197999	-0.1986	-0.19853
g_3	−0.493501	−0.499172	−0.499172	-0.9695	-0.96764
g_4	−0.904644	−0.901472	−0.901555	-0.9941	-0.9942
g_5	0	0	0	-0.0001	-0.0021
g_6	0.000633	0	0	-0.0002	-0.00027
g_7	−0.7025	−0.702500	−0.7025	-0.7024	-0.70249
g_8	0	0	0	-0.0002	-0.00052
g_9	−0.583333	−0.583333	−0.583333	-0.5832	-0.58312
g_{10}	−0.054889	−0.051325	−0.051326	-0.0626	-0.08102
g_{11}	0	−0.010852	−0.010695	-0.0164	-0.01124
$f(x)$	2994.744241	2996.35	2997.06	2999.55	3000.51

5.6 Summary

This chapter introduced two new strategies into the standard Bees Algorithm. The first strategy implements self-adaptive bee recruitment mechanism on selected sites. It is designed to reduce the number of parameters of existing standard Bees Algorithm, which will result in lesser efforts in fine tuning of the parameters. The second strategy uses the best solution found so far to guide the neighbourhood shrinking strategy. This strategy aimed at making the search strategy bias towards the best solution found so far. Experimental results showed that this new version of Bees Algorithm excelled the standard Bees Algorithm on various unconstrained numerical benchmark functions considered particularly, the high dimensional functions.

Furthermore, the *g*BA is also applied on five constrained mechanical design problems. The experimental results produced by *g*BA were significantly better than those of the standard Bees Algorithm on three of the problems.

CHAPTER 6

Conclusion and Future Work

A summary of the main contributions, together with the conclusion of this research is provided in this chapter. It also provides suggestions for future work.

6.1 Contributions

The main contributions of this research are:

i. Providing guiding direction to the neighbourhood search to improve exploitation strategies of the algorithm, improving its overall performance.

ii. Employing a new operator between the selected sites and abandoned sites to produce better solutions closer to the local or global optima.

iii. Reduction in number of parameters to be tuned by introducing new bee recruitment mechanisms, eliminating two parameters which are number of recruit bees for elite sites (*nep*) and number of recruit bees for best sites (*nsp*).

iv. Development of a new neighbourhood shrinking strategy to deal with slow convergence search near global optima more efficiently.

v. The improved versions of the Bees Algorithm are developed without addition of any extra parameters.

vi. The use of a similar set of parameter values for testing on benchmark functions, showing the robustness of the algorithm.

6.2 Conclusions

The objectives mentioned in the first chapter have been achieved.

Three different improved versions of the Bees Algorithm have been presented in this thesis. All proposed algorithms were tested on unconstrained numerical benchmark functions and constrained mechanical design problems. Results obtained are provided in the associated chapters. The conclusions are as follows:

1. A guiding direction was provided for the neighbourhood search, forming a new improved Bees Algorithm named the Nelder and Mead Bees Algorithm (NMBA). This algorithm performed better than the standard Bees Algorithm, specifically in terms of speed on most of the problems tested. The NMBA performed moderately on the functions with flat surface like landscape and noisy characteristic surface because this type of surface might have no directions to guide the neighbourhood search. The algorithm also showed fair performance on applications of constrained mechanical design problems. This addresses Objective (i).
2. The second improvement was utilising the recombination operator between selected sites (solutions) and abandoned sites (solutions) to enable sharing of information. The recombination-based Bees Algorithm (*r*BA) utilises recombination operator to produce new good solutions closer to the local or global optima. Thus, this allows the neighbourhood search to start at a good initial position, which is an advantage for the algorithm. Similar problems were also tested for this proposed algorithm. The results obtained on most of the problems showed better performance in terms of solution found and convergence speed. However, the algorithm gives similar performance on non-separable type of functions. This result might be due to unsuitability of the

recombination operator for this type of function as it could produce solutions away from the current solution. This addresses Objective (ii).

3. Lastly, a new self-adaptive bee recruitment mechanism was introduced into the Bees Algorithm, reducing the number of parameters to be tuned. It is also more nature-based in terms of honey bee foraging behaviour where the priority of selection considers other criteria instead of fully based on fitness value. Furthermore, a new modified neighbourhood shrinking strategy was also developed to deal with slow convergence and stagnation during search near the global optima. The neighbourhood shrinking based on best solution found so far (*g*BA) performed at least similar to the standard Bees Algorithm or better on most of the problems tested with lesser number of parameters. This addresses Objectives (iii and iv).

6.3 Future Work

Future research might proceed in the following directions:

1. Developing an algorithm that requires fewer parameters or a fully adaptive Bees Algorithm. This would certainly benefit users of the algorithm in terms of ease of application
2. Investigating the performance of the proposed algorithms on other real world problems
3. Exploring the performance of the proposed algorithms on higher dimensions and different classes of numerical functions

APPENDICES

Appendix A – List of Benchmark Functions

No.	Function	Dimension	Function	Search range	Minimum
f_1	Schwefel	2	$f(x) = x_1 sin(\sqrt{\lvert x_1 \rvert} - x_2 sin(\sqrt{\lvert x_2 \rvert}$	$[-500, 500]^D$	-837.9658
f_2	Easom	2	$f(x) = -\cos x_1 + \cos x_2\, e^{[(x_1-\pi)^2-(x_2-\pi)^2]}$	$[-100,100]^D$	-1
f_3	Goldstein & Price	2	$A(x) = 1+ (x_1 + x_2 + 1)^2 (19-14x_1+3x_1^2-14x_2 +6x_1x_2 +3x_2)$ $B(x) = 30+ (2x_1 - 3x_2)^2 (18-32x_1 + 12x_1^2 +48x_2 - 36x_1x_2 + 27x_2^2)$ $f(x) = AB$	$[-2, 2]^D$	3
f_4	Martin & Gaddy	2	$f(x)= (x_1 - x_2)^2 + \frac{(x_1-x_2-10)^2}{3}$	$[-20, 20]^D$	0
f_5	Schaffer	2	$f(x)= 0.5 + \frac{(sin\sqrt{x_1^2+x_2^2})^2 - 0.5}{[1+0.001(x_1^2+x_2^2)]^2}$	$[-100, 100]^D$	0
f_6	Camel six hump	2	$f(x)= (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (4x_2^2 - 4)x_2^2$	$[-5, 5]^D$	-1.0316
f_7	Shekel	4	$f(x)= - \sum_{i=1}^{10} \frac{1}{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i}$	$[0, 10]^D$	-10.5364
f_8	Michaelewicz	5	$f(x) = \sum_{i=1}^{D} (\sin x_i)\, (\sin(i\, x_i^2))^{2m}$	$[0, \pi]^D$	-4.687
f_9	Trid	6	$f(x)= \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=2}^{D} x_i\, x_i - 1$	$[-D^2, D^2]^D$	-50
f_{10}	Shekel	10	$f(x)= - \sum_{j=1}^{30} \frac{1}{c_i \sum_{j=1}^{D}(x_j - a_{ij})^2}$	$[0, 10]^D$	-10.2028
f_{11}	Griewank	10	$f(x) = \frac{1}{4000}\sum_{i=1}^{D}(x_i - 100)^2 \prod_{i=1}^{D} cos(\frac{x_i-100}{\sqrt{i+1}})+1$	$[-600, 600]^D$	0
f_{12}	Langermann	10	$f(x)= c_i \sum_{i=1}^{30} e^{-\frac{1}{\pi}\sum_{j=1}^{D}(x_j-a_{ij})^2} cos(\pi \sum_{j=1}^{D}(x_j - a_{ij})^2)$	[0, 10]	-1.4

*f*13	Rosenbrock	10	$f(x) = x_1 sin(\sqrt{\lvert x_1 \rvert} - x_2 sin(\sqrt{\lvert x_2 \rvert}$	$[-50, 50]^D$	0
*f*14	Zakharov	10	$f(x) = -\cos x_1 + \cos x_2\, e^{[(x_1 - \pi)^2 - (x_2 - \pi)^2]}$	$[-5, 5]^D$	0
*f*15	Rastrigin	10	$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	0

Appendix B – Characteristics of Benchmark Functions

	Unimodal	Multimodal	Minima on origin	Minima on grid	Separable	Non-separable	Wavelike	Flat Surface	Dimensionality
Easom	X			X	X			X	
Trid	X		X			X			X
Rosenbrock	X		X			X		X	X
Zakharov	X		X			X			X
Martin & Gaddy	X			X		X			
Schaffer		X	X			X	X		
Rastrigin		X	X		X		X		X
Schwefel		X		X	X				
Michaelewicz		X		X	X			X	X
Goldstein & Price		X		X		X			
Camel Six Hump		X		X		X			
Shekel_4D		X		X		X			X
Shekel_10D		X		X		X			X
Griewank		X		X		X	X		X
Langermann		X		X		X			X

Appendix C – List of Mechanical Design Problems

Problem 1: **Welded Beam Design**

Welded beam design illustrated in Figure B.1 minimizes the cost of the beam subject to constraints on shear stress, τ , bending stress in the beam, σ , buckling load on the bar, Pc, end deflection of the beam, δ, and side constraints. There are four design parameters (x_1, x_2, x_3 and x_4) for this problem as shown in Figure B.1.

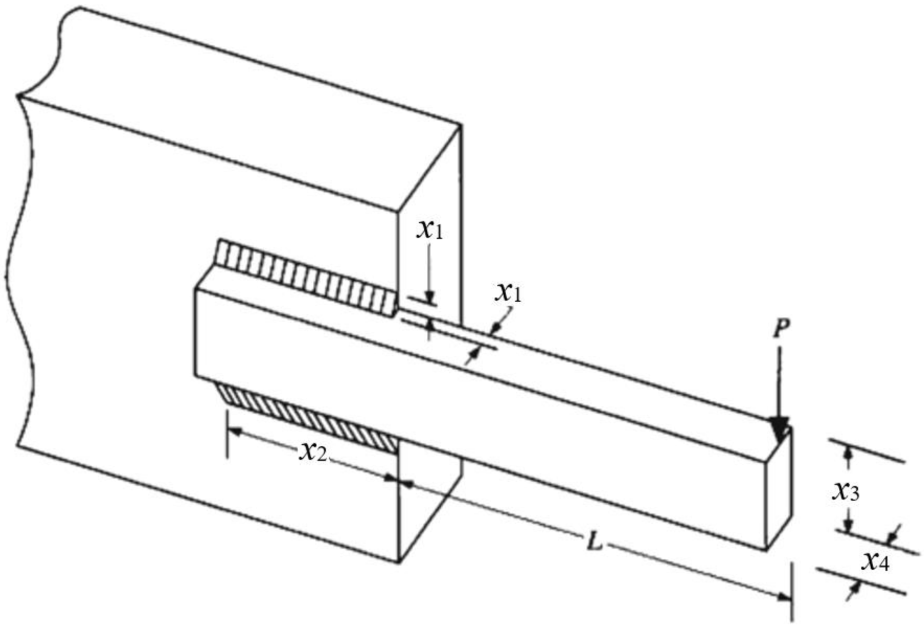

Figure B.1: Welded Beam Design (Akay and Karaboga, 2012)

Minimise: $f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4 (14.0+x_2)$.

Subject to: $g_1(x) = \tau(x)-\tau_{max} \leq 0$,

$g_2(x) = \sigma(x)-\sigma_{max} \leq 0$,

$g_3(x) = x_1 - x_4 \leq 0$,

$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0+x_2)-5.0 \leq 0$,

$g_5(x) = 0.125-x_1 \leq 0$,

$g_6(x) = \delta(x)-\delta_{max} \leq 0$,

$g_7(x) = P - Pc\ (x) \leq 0,$

where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x^2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{P}{x_1 x_2 \sqrt{2}} \quad , \tau'' = \frac{MR}{J}, \quad M = P(L + \frac{x_2}{2})$$

$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2} \quad , J = 2\left[\sqrt{2}x_1 x_2 \left\{\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right\}\right], \sigma(x) = \frac{6PL}{x_4 x_3^2},$$

$$\delta(x) = \frac{4PL^3}{E x_4 x_3^3}, \; P_C = \frac{4.013E\sqrt{\frac{x_4^6 x_3^2}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

P = 6000lb, L = 14in., E = 30e6 psi, G = 12e6 psi, τ_{max}= 13 600 psi, σ_{max} = 30 000 psi, δ_{max}= 0.25 in.

$0.1 \leq x_1 \leq 2.0$, $0.1 \leq x_2 \leq 10.0$,

$0.1 \leq x_3 \leq 10.0$, $0.1 \leq x_4 \leq 2.0$.

Problem 2: **Design of Pressure Vessel**

Second example is minimization of the total cost comprising of material, forming and welding costs of a cylindrical vessel as shown in Figure B.2. The four design variables are x_1 (thickness of the shell), x_2 (thickness of the head), x_3 (inner radius) and x_4 (length of the cylindrical section of the vessel, not including the head). x_1 and x_2 are to be in integral multiples of 0.0625 inch which are the available thicknesses of rolled steel plates. The radius x_3 and the length x_4 are continuous variables.

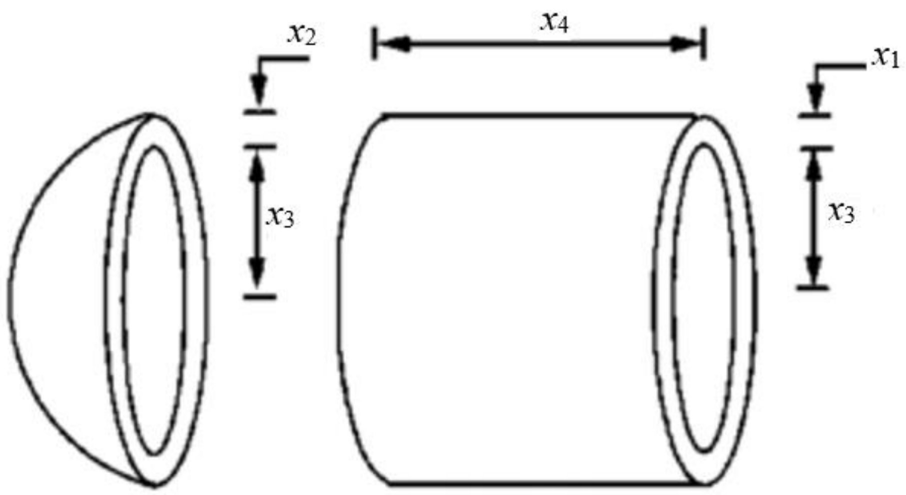

Figure B.2: Design of Pressure Vessel (Akay and Karaboga, 2012)

Minimise: $f(x) = 0.6224x_1x_3x_4+1.7781x_2x_3^2 +3.1661x_1^2\, x^4 +19.84x_1^2\, x_3$

Subject to: $g_1(x) = -x_1+0.0193x_3 \leq 0$,

$g_2(x) = -x_2+0.00954x_3 \leq 0$,

$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3+1296\,000 \leq 0$,

$g_4(x) = x_4-240 \leq 0$,

where

$0 \leq x_1 \leq 99$, $0 \leq x_2 \leq 99$,

$10 \leq x_3 \leq 200$, $10 \leq x_4 \leq 200$.

Problem 3: **Tension/Compression spring problem**

The tension/compression problem deals with the minimisation of the weight of the tension/compression spring shown in Figure B.3, subject to constraints on the minimum deflection, shear stress, surge frequency, diameter and design variables. The design variables are the wire diameter, *d*, the mean coil diameter, *D*, and the number of active coils, *N*. The problem is formulated as:

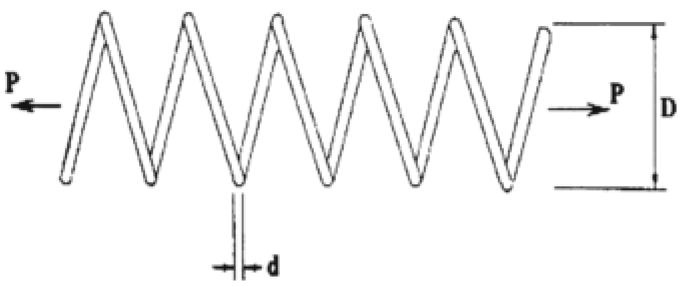

Figure B.3: Tension/Compression Spring problem (Akay and Karaboga, 2012)

Minimise: $f(x) = (N+2)Dd^2$

Subject to: $g_1(x) = 1 - \frac{D^3 N}{71785 d^4} \leq 0,$

$g_2(x) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108 d^2} - 1 \leq 0,$

$g_3(x) = 1 - \frac{140.45 d}{D^2 N} \leq 0,$

$g_4(x) = \frac{D+d}{1.5} - 1 \leq 0.$

$X = (d, D, N)^T$, $0.05 \leq d \leq 2.0$, $0.25 \leq D \leq 1.3$, $2.0 \leq N \leq 15.0$

Problem 4: **Speed Reducer Design**

The aim of the speed reducer design shown in Figure B.4 is to minimise the weights of the speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. Design parameters of the speed reducer problem, the face width (x_1), module of teeth (x_2), number of teeth in the pinion (x_3), length of the first shaft between bearings (x_4), length of the second shaft between bearings (x_5) and the diameter of the first shaft (x_6) and second shaft (x_7). This is an example of a mixed integer programming problem. The third variable (number of teeth) is of integer value while all other variables are continuous.

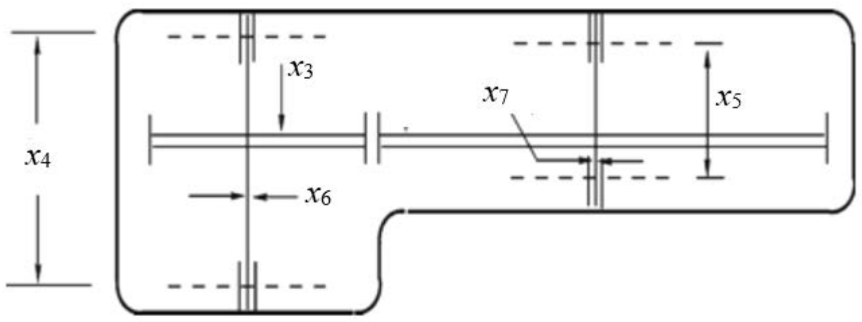

Figure B.4: Speed Reducer Design (Akay and Karaboga, 2012)

Minimise: $f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2)$
$+7.4777(x_6^3 + x_7^3)$

Subject to: $g_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0,$

$g_2(x) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0,$

$g_3(x) = \frac{1.93x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0,$

$g_4(x) = \frac{1.93x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0,$

$g_5(x) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right)^{1/2}}{110.0x_6^3} - 1 \leq 0,$

$g_6(x) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 157.5 \times 10^6\right)^{1/2}}{85.0x_7^3} - 1 \leq 0,$

$g_7(x) = \frac{x_2 x_3}{40} - 1 \leq 0,$

$g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0,$

$g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0,$

$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0,$

$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0,$

where $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, $5.0 \leq x_7 \leq 5.5$.

Problem 5: **Multiple Disc Clutch Brake**

Figure B.5 shows a multiple disc clutch brake. The objective is to minimize the mass of the multiple disc clutch brake using five discrete variables: inner radius (r_i=60, 61, 62, . . . , 80), outer radius (r_o= 90, 91, 92, . . . , 110), thickness of discs (t = 1, 1.5, 2, 2.5, 3), actuating force (F = 600, 610, 620, . . . , 1000) and number of friction surfaces (Z = 2, 3, 4, 5, 6, 7, 8, 9).

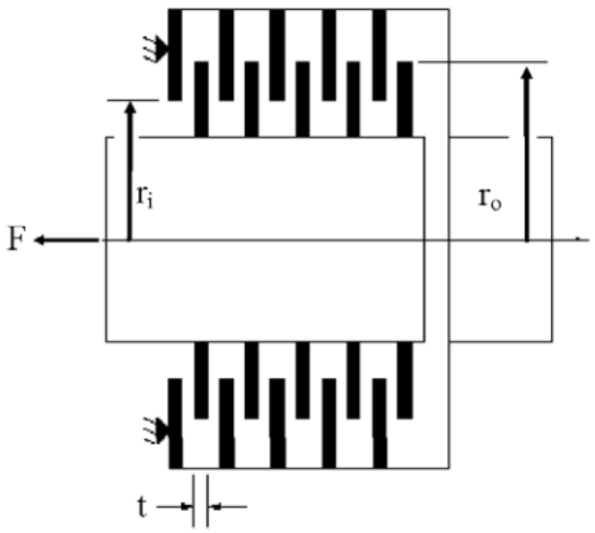

Figure B.5: Multiple disc clutch brake (Akay and Karaboga, 2012)

Minimise: $f(x) = \pi(r_o^2 - r_i^2)t(Z+1)\rho$

Subject to: $g_1(x) = r_o - r_i - \Delta r \geq 0,$

$g_2(x) = l_{max} - (Z+1)(t+\delta) \geq 0,$

$g_3(x) = p_{max} - p_{rz} \geq 0,$

$g_4(x) = p_{max}v_{srmax} - p_{rz}v_{sr} \geq 0,$

$g_5(x) = v_{srmax} - v_{sr} \geq 0,$

$g_6(x) = T_{max} - T \geq 0,$

$g_7(x) = M_h - sM_s \geq 0,$

$g_8(x) = T \geq 0,$

where $M_h = \frac{2}{3}\mu FZ\frac{r_o^3 - r_i^3}{r_o^2 - r_i^2}$, $p_{rz} = \frac{F}{\pi(r_o^2 - r_i^2)}$, $v_{sr} = \frac{2\pi n(r_o^3 - r_i^3)}{90(r_o^2 - r_i^2)}$, $T = \frac{I_z\pi n}{30(M_h + M_f)}$,

Δr = 20mm, t_{max} = 3mm, t_{min} = 1.5mm, l_{max} = 30mm, Z_{max} = 10, v_{srmax} = 10 *m/s*, μ= 0.5, *s* = 1.5, M_s = 40 N m, M_f= 3 N m, *n* = 250 rpm, p_{max} =1MPa, I_z = 55 kg mm^2, T_{max} = 15 s, F_{max} = 1000N, $r_{i\mathrm{min}}$ = 55mm, $r_{o\mathrm{max}}$ = 110mm.

REFERENCES

Abbass, H. A. 2001. MBO: Marriage in Honey Bees Optimization-A Haplometrosis Polygynous Swarming Approach. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001). Seoul, May 2001,* pp.207-214.

Ahmad Farhan, A., & Bilal, S. 2011. A Novel Fast and Robust Digital Image Watermarking Using Bee Algorithm. In *Multitopic Conference (INMIC), 2011 IEEE 14th International, Karachi, 2011,* pp. 82–86.

Ahmad, S. A. 2012. *A study of Search Neighbourhood in The Bees Algorithm*. Cardiff University.

Ahmad, S. A., Pham, D. T., & Faieza, A. A. 2014. Combination of Adaptive Enlargement and Reduction in the Search Neighbourhood in the Bees Algorithm. *Applied Mechanics and Materials*, *564*, 614–618.

Ahmad, S. A., Pham, D. T., Ng, K. W., & Ang, M. C. 2012. TRIZ-inspired Asymmetrical Search Neighborhood in the Bees Algorithm. *Sixth Asia Modelling Symposium*, *May 2012*, pp.29-33

Akay, B., & Karaboga, D. 2012. Artificial Bee Colony Algorithm for Large-scale Problems and Engineering Design Optimization. *Journal of Intelligent Manufacturing*, *23*(4), 1001–1014.

Akbari, R., Hedayatzadeh, R., Ziarati, K., & Hassanizadeh, B. 2012. A Multi-Objective Artificial Bee Colony Algorithm. *Swarm and Evolutionary Computation*, *2*, 39–52.

Akpinar, Ş., & Baykasoğlu, A. 2014a. Modeling and Solving Mixed-Model Assembly Line Balancing Problem with Setups. Part I: A Mixed Integer Linear Programming Model. *Journal of Manufacturing Systems*, *33*(1), 177–187.

Akpinar, Ş., & Baykasoğlu, A. 2014b. Modeling and Solving Mixed-model Assembly Line Balancing Problem with Setups. Part II: A Multiple Colony Hybrid Bees Algorithm. *Journal of Manufacturing Systems*.

Anantasate, S., & Bhasaputra, P. 2011. A Multi-objective Bees Algorithm for Multi-objective Optimal Power Flow Problem. In *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI), Association of Thailand-Conference 2011*, *May 2011*, pp. 852-856.

Ang, M. C., Pham, D. T., & Ng, K. W. 2009. Minimum-Time Motion Planning for a Robot Arm Using the Bees Algorithm. *In 7th IEEE International Conference on industrial informatics (INDIN 2009), June 2009*, pp. 487-492

Ang, M. C., Pham, D. T., Anthony, J. S., & Ng, K. W. 2010. PCB Assembly Optimisation Using the Bees Algorithm Enhanced with TRIZ Operators. *In IECON 2010-36th Annual Conference on IEEE industrial Electronics Society, November 2010*, pp. 2708-2713.

Ang, M. C., Ng, K. W., & Pham, D. T., 2013. Combining the Bees Algorithm and shape grammar to generate branded product concepts. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *227*(12), 1860–1873.

Attaran, B., & Ghanbarzadeh, A. 2014. Bearing Fault Detection Based on Maximum Likelihood Estimation and Optimized ANN Using the Bees Algorithm. *Journal of Applied and Computational Mechanics*, *1*(1), 35–43.

Attaran, B., Ghanbarzadeh, A., Zaeri, R., & Moradi, S. 2011. Intelligent Fault Diagnosis of Rolling Bearing Based on Optimized Complementary Capability Features and RBF Neural Network by Using the Bees Algorithm. *The 2nd International Conference on Control, Instrumentation and Automation*, *Shiraz*, *2011*, pp. 764–769.

Aydogdu, I., & Akijn, A. 2011. Bees Algorithm Based Optimun Design of Open Canal Sections. *International Journal of Engineering and Applied Sciences*, *3*(4), 21–31.

Azzeh, M. 2011. Adjusted Case-Based Software Effort Estimation Using Bees Optimization Algorithm. In *ES'11 Proceedings of the 15th international conference on Knowledge-based and intelligent information and engineering systems*, *2011*, pp. 315–324.

Bahamish, H. A. A., Abdullah, R., & Salam, R. A. 2008. Protein Conformational Search Using Bees Algorithm. *In IEEE Second Asia International Conference on Modeling & Simulation (AICMS 08), May 2008*, pp. 911–916.

Bahrainian, S. S., Mehrdoost, Z., & Ghanbarzadeh, A. 2013. The Application of Bees Agorithm in Finding the Neutral Stability Curve for Plane Poiseuille Flow. *Meccanica*, *48*(9), 2255–2261.

Battiti, R., & Tecchiolli, G. 1994. The Reactive Tabu Search. *ORSA Journal of Computing*, *6*(2), 126–140.

Beasley, D., Bull, D. R., & Martin, R. R. 1993a. An Overview of Genetic Algorithms : Part 1, Fundamentals. *University Computing*, *2*(15), 58 – 69.

Beasley, D., Bull, D. R., & Martin, R. R. 1993b. An Overview of Genetic Algorithms : Part 2 ,Research Topics. *University Computing*, *15*(4), 170 – 181.

Beni, G. 2005. From Swarm Intelligence to Swarm Robotics. *Swarm Robotics*, pp. 1–9.

Biegler-könig, F. 2013. Artificial Bee Colony Algorithm for Power Plant Optimization. In *Proceedings 27th European Conference on Modelling and Simulation*, pp. 778–793.

Blum, C., & Roli, A. (2003). Metaheuristics In Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, *35*(3), 189–213.

Bonab, M. B., Zaiton, S., Hashim, M., Erne, N., Bazin, N., Khalaf, A., & Alsaedi, Z. 2015. An Effective Hybrid of Bees Algorithm and Differential Evolution Algorithm in Data Clustering. *Mathematical Problems in Engineering, vol. 2015, Article ID 240419,* pp. 1-17.

Boumazouza, D., Sefouane, Y., Djeddi, M., Khelouat, B., & Benatchba, K. 2013. Bees for Block Matching. In *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society* (pp. 2390–2394).

Boussaïd, I., Lepagnot, J., & Siarry, P. 2013. A Survey on Optimization Metaheuristics. *Information Sciences*, *237*, 82–117.

Castellani, M., Pham, Q. T., & Pham, D. T. 2012. Dynamic Optimisation by a Modified Bees Algorithm. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *226*(7), 956–971.

Claus Bendtsen. 2012. PSO: Particle Swarm Optimization. Retrieved from http://cran.r-project.org/package=pso

Corne, D. W., Reynolds, A. P., & Bonabeau, E. 2012. Swarm Intelligence. In G. Rozenberg, T. Bäck, & JoostN (Eds.), *Handbook of Natural Computing* (pp. 1599–1622). Kok, New York: Springer.

Daoud, S., Yalaoui, F., Amodeo, L., Chehade, H., & Duperray, P. 2012. A Hybrid Bees Algorithm for Solving a Robotic Assembly Line Balancing Problem. In *Uncertainty Modeling in Knowledge Engineering and Decision Making* (pp. 1275–1280).

Das, S., Mullick, S. S., & Suganthan, P. N. 2016. Recent Advances in Differential Evolution-An Updated Survey. *Swarm and Evolutionary Computation*, *27*, 1–30.

Dereli, T., & Das, G. S. 2011. A Hybrid “Bee(s) Algorithm” for Solving Container Loading Poblems. *Applied Soft Computing*, *11*(2), 2854–2862.

Diwold, K., Beekman, M., & Middendorf, M. 2010. Bee Nest Site Selection as an Optimization Process. *In Proceedings of the 12th Alife Conference*, *Odense*, *2010*, pp.626-633.

Dorigo, M., Maniezzo, V., & Colorni, A. 1996. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *26*(1), 29 – 41.

Dorigo, M., & Stutzle, T. 2010. Ant Colony Optimization : Overview and Recent Advances. In *Handbook of Metaheuristics* (pp. 227–263).

Durongdumrongchai, P., Sangiamvibool, W., Aurasopon, A., & Pothiya, S. 2014. Robust And Optimal Fuzzy Logic Proportional Integral Derivative Controllers Design By Bee Algorithm For Hydro-Thermal System. *Rev. Roum. Sci. Techn. – Électrotechn. et Énerg*, *59*(2), 193–203.

Ebrahimzadeh, A., Addeh, J., & Ranaee, V. 2013. Recognition of Control Chart Patterns Using an Intelligent Technique. *Applied Soft Computing*, *13*(5), 2970–2980.

Eldukhri, E. E., & Kamil, H. G. 2013. Optimisation of Swing-up Control Parameters for A Robot Gymnast Using The Bees Algorithm. In *Proceedings of 8th International Symposium on Intelligent and Manufacturing Systems (IMS 2012)*, *Turkey*, pp. 456–466.

Ercin, O., & Coban, R. 2011. Comparison of the Artificial Bee Colony and the Bees Algorithm for PID Controller Tuning. In *International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, pp. 595–598.

Fahmy, A. A., Kalyoncu, M., & Castellani, M. 2011. Automatic Design of Control Systems for Robot Manipulators Using the Bees Algorithm. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *226*(4), 497–508.

Fister, I., Yang, X. S., & Brest, J. 2013. A Comprehensive Review of Firefly Algorithms. *Swarm and Evolutionary Computation*, *13*, 34–46.

Forsati, R., Keikha, A., & Shamsfard, M. 2015. An Improved Bee Colony Optimization Algorithm With An Application To Document Clustering. *Neurocomputing*, *159*(1), 9–26.

Gao, W. F., Liu, S. Y., & Huang, L. L. 2013. A Novel Artificial Bee Colony Algorithm with Powell's Method. *Applied Soft Computing Journal*, *13*(9), 3763–3775.

Gholipour, R., Khosravi, A., & Mojallali, H. 2015. Multi-objective Optimal Backstepping Controller Design for Chaos Control in a Rod-type Plasma Torch System Using Bees algorithm. *Applied Mathematical Modelling*, *39*, 4432–4444.

Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, *13*(5), 533–549.

Guney, K., & Onay, M. 2007. Amplitude-only Pattern Nulling of Linear Antenna Arrays With the Use of Bees Algorithm. *Progress in Electromagnetic Research*, *70*, 21–36.

Guney, K., & Onay, M. 2008. Bees Algorithm for Design of Dual-Beam Linear Antenna Arrays with Digital Attenuators and Digital Phase Shifters. *International Journal of RF and Microwave Computer Aided Engineering*, *18*(4), 337–347.

Guney, K., & Onay, M. 2010. Bees Algorithm for Interference Suppression of Linear Antenna Arrays by Controlling the Phase-only and Both the Amplitude and Ahase. *Journal of Communications Technology and Electronics*, *58*(12), 1147–1156.

Guney, K., & Onay, M. 2011. Synthesis of Thinned Linear Antenna Arrays Using Bees Algorithm. *Microwave and Optical Technology Letters*, *53*(4), 795–799.

Haddad, O. B., Afshar, A., & Mariño, M. A. 2006. Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. *Water Resources Management*, *20*(5), 661–680.

Hazli, M., Zabil, M., & Zamli, K. Z. 2013. Implementing a T -Way Test Generation Strategy Using Bees Algorithm. *International Journal Advance Soft Compu. Appl*, *5*(3), 116–126.

Hedayatzadeh, R., Hasanizadeh, B., Akbari, R., & Ziarati, K. 2010. A Multi-Objective Artificial Bee Colony for Optimizing Multi-Objective Problems. *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), Chengdu, 2010*, pp. 277-281.

Hooke, R., & Jeeves, T. A. 1961. "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the ACM*, *8*(2), 212–229.

Idris, R. M., Khairuddin, A., & Mustafa, M. W. 2009. A Multi-objective Bees Algorithm for Optimum Allocation of FACTS Devices for Restructured Power System. In *TENCON 2009 - 2009 IEEE Region 10 Conference*, *Singapore, 2009*, pp. 1–6.

Idris, R. M., Kharuddin, A., & Mustafa, M. W. 2010. Available Transfer Capability Determination Using Bees Algorithm. In *Universities Power Engineering Conference (AUPEC), 2010 20th Australasian*, pp. 1–6.

Ilka, R., Gholamian, S. A., & Addeh, J. 2013. Optimun Design of A Five-Phase Surface-Mounted Permanent Magnet Syncronous Motor Using Bees Algorithm. *Journal of Electrical Engineering*, *13*(1), 146–153.

Jamil, M., & Yang, X. S. 2013. A Literature Survey of Benchmark Functions for Global Optimisation Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*, 150–194.

Jamil, M., Yang, X. S., & Zepernick, H. J. D. 2013. Test Functions for Global Optimization: A Comprehensive Survey. In *Swarm Intelligence and Bio-Inspired Computation* (pp. 193–222).

Jones, K. O., & Bouffet, A. 2008. Comparison of Bees Algorithm, Ant Colony Optimisation and Particle Swarm Optimisation for PID Controller Tuning. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing-CompSysTech '08* (p. IIIA.9). New York, New York, USA: ACM Press.

Kang, F., Li, J., & Ma, Z. 2011. Rosenbrock Artificial Bee Colony Algorithm for Accurate Global Optimization of Numerical Functions. *Information Sciences*, *181*(16), 3508–3531.

Karaboga, D. 2005. *An Idea Based On Honey Bee Swarm For Numerical Optimization*: Technical Note TR06, Erciyes Univ Press, Erciyes.

Karaboga, D., & Akay, B. 2009. A Comparative Study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132.

Karaboga, D., & Basturk, B. 2007. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) algorithm. *Journal of Global Optimization*, *39*, 459–471.

Karaboga, D., & Gorkemli, B. 2012. A Quick Artificial Bee Colony-qABC-Algorithm for Optimization Problems. In *Innovations in Intelligent Systems and Applications (INISTA),* *2012 International Symposium* (pp. 1–5).

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. 2012. A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications. *Artificial Intelligence Review*, *42*(1), 21–57.

Kennedy, J., & Eberhart, R. 1995. Particle Swarm Optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on, Perth, WA* (Vol. 4, pp. 1942–1948).

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680.

Kolda, T., Lewis, R., & Torczon, V. 2003. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, *45*(3), 385–482.

Koza, P. J. 1994. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. *MIT Press* (Vol. 33).

Leeprechanon, N., & Polratanasak, P. 2010. Multiobjective Bees Algorithm with Clustering Technique for Environmental/Economic Dispatch. In *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference*, pp. 621–625.

Lewis, R. M., Torczon, V., & Trosset, M. W. 2000. Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, *124*(1-2), 191–207.

Lien, L. C., & Cheng, M. Y. 2014. Particle Bee Algorithm for Tower Crane Layout with Material Quantity Supply and Demand Optimization. *Automation in Construction*, *45*, 25–32.

Long, V. T., & Nhan, N. V. 2012. Bees Algorithm-based Optimization of Component Size and Control Strategy Parameters for Parallel Hybrid Electric Vehicles. *International Journal of Automotive Technology*, *13*(7), 1177–1183.

Lucic, P., & Teodorović, D. 2002. Transportation Modeling: An Artificial Life Approach. In *14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings.* (pp. 216 – 223).

Maneechote, T., & Luangpaiboon, P. 2010. An Exploration of Bees Parameter Settings via Modified Simplex and Conventional Design of Experiments. In *Proceedings of National Operations Research Co-operative Research Network Conference, Bangkok, Thailand,* pp. 11–16.

Marini, F., & Walczak, B. 2015. Particle Swarm Optimization (PSO). A Tutorial. *Chemometrics and Intelligent Laboratory Systems*, *149*, 153–165.

Mastrocinque, E., Yuce, B., Lambiase, A., & Packianather, M. S. 2013. A Multi-objective Optimization for Supply Chain Network Using the Bees Algorithm. *International Journal of Engineering Business Management*, *5*, 1–11.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. *Journal Chemical Physics*, *21*(6), 1087–1092.

Ming, H., Baohui, J., & Xu, L. 2010. An Improved Bee Evolutionary Genetic Algorithm. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Xiamen, 2010*, pp. 372-374.

Mirsadeghi, E., & Shariat Panahi, M. 2012. Hybridizing Artificial Bee Colony with Simulated Annealing. *International Journal of Hybrid Information Technology*, *5*(4), 11–18.

Mirzakhani Nafchi, A., Moradi, A., Ghanbarzadeh, A., Rezazadeh, A., & Soodmand, E. 2011. Solving Engineering Optimization Problems Using the Bees Algorithm. In *2011 IEEE Colloquium on Humanities*, *Science and Engineering, Penang*, *2011*, pp. 162-166.

Moayedikia, A., Jensen, R., Wiil, U. K., & Forsati, R. 2015. Weighted Bee Colony Algorithm For Discrete Optimization Problems With Application To Feature Selection. *Engineering Applications of Artificial Intelligence*, *44*, 153–167.

Mollabakhshi, N., & Eshghi, M. 2013. Combinational Circuit Design Using Bees Algorithm. In *IEEE Conference Anthology, China, 2013,* pp. 1-4.

Moradi, A., Nafchi, A. M., Ghanbarzadeh, A., & Soodmand, E. 2011. Optimization of Linear and Nonlinear Full Vehicle Model for Mmproving Ride Comfort vs. Road Holding With the Bees Algorithm. In *2011 IEEE Colloquium on Humanities, Science and Engineering, Penang, 2011,* pp. 17-22.

Moradi, S., Fatahi, L., & Razi, P. 2010. Finite element model updating using bees algorithm. *Structural and Multidisciplinary Optimization*, *42*(2), 283–291.

Moradi, S., & Kargozarfard, M. H. 2013. On multiple crack detection in beam structures. *Journal of Mechanical Science and Technology*, *27*(1), 47–55.

Moradi, S., Razi, P., & Fatahi, L. 2011. On The Application of Bees Algorithm to The Problem of Crack Detection of Beam-Type Structures. *Computers & Structures*, *89*(23-24), 2169–2175.

Muhamad, Z., Mahmuddin, M., Nasrudin, M. F., & Sahran, S. 2011. Local Search Manoeuvres Recruitment in The Bees Algorithm. In *Proceedings of The 3rd International Conference on Computing and informatics, ICOCI 2011, 8-9 June, 2011 Bandung, Indonesia,* pp.43-48

Nelder, J. A., & Mead, R. 1964. A Simplex Method for Function Minimization. *Computer Journal*, *7*, 308–313.

Özbakir, L., Baykasoğlu, A., & Tapkan, P. 2010. Bees Algorithm for Generalized Assignment Problem. *Applied Mathematics and Computation*, *215*(11), 3782–3795.

Packianather, M. S., Fruggiero, F., Mastrocinque, E., Holloway, R., & Lambiase, A. 2014. Novel Genetic Bees Algorithm Applied to Single Machine Scheduling Problem. In *2014 World Automation Congress (WAC), Waikoloa, HI, 2014,* pp. 906-911.

Packianather, M. S., & Kapoor, B. 2015. A Wrapper-based Feature Selection Approach Using Bees Algorithm for a Wood Defect Classification System. In *2015 10th System of Systems Engineering Conference (SoSE), San Antonio, TX, 2015,* pp. 498-503.

Parsa, H. R., AsgharGholamian, S., & Abbasi, M. 2013. Design and Optimization of Eddy Current Testing Probe Using Bees Algorithm and Finite Element Analysis. *International Journal of Modern Education and Computer Science*, *5*(12), 40–46.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. 2006a. The Bees Algorithm–A Novel Tool for Complex Optimisation Problems. In *Intelligent Production Machines and Systems - 2nd I*PROMS Virtual International Conference*,pp. 454–459.

Pham, D. T., Soroka, A. J., Ghanbarzadeh, A., Koç, E., Otri, S., & Packianather, M. 2006b. Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm. In *006 4th IEEE International Conference on Industrial Informatics, Singapore, 2006,* pp. 1346-1351.

Pham, D. T., Afify, A. A., & Koç, E. 2007a. Manufacturing Cell Formation Using the Bees Algorithm. In *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*,pp. 523–528.

Pham, D. T., & Castellani, M. 2009. The Bees Algorithm: Modelling Foraging Behaviour to Solve Continuous Optimization Problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *223*(12), 2919–2938.

Pham, D. T., & Castellani, M. 2013. Benchmarking and Comparison of Nature-inspired Population-based Continuous Optimisation Algorithms. *Soft Computing*, *18*(5), 871–903.

Pham, D. T., & Castellani, M. 2015. A Comparative Study of the Bees Algorithm as a Tool for Function Optimisation. *Cogent Engineering*, *2*(1), 1091540.

Pham, D. T., Darwish, A. H., & Eldukhri, E. E. 2009a. Optimisation of a Fuzzy Logic Controller Using the Bees Algorithm. *International Journal of Computer Aided Engineering and Technology*, *1*(2), 250.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Muhamad, Z. 2005. *Bee Algorithm - A Novel Approach to Function Optimisation, Technical Report, MEC 0501*. Manufacturing Engineering Centre, Cardiff University, Cardiff, UK.

Pham, D. T., Ghanbarzadeh, A., Otri, S., & Koç, E. 2009b. Optimal Design of Mechanical Components Using the Bees Algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *223*(5), 1051–1056.

Pham, D. T., & Haj Darwish, A. 2010. Using the Bees Algorithm with Kalman Filtering to Train an Artificial Neural Network for Pattern Classification. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *224*(7), 885–892.

Pham, D. T., & Kalyoncu, M. 2009. Optimisation of A Fuzzy Logic Controller for A Flexible Single-Link Robot Arm Using the Bees Algorithm. In *2009 7th IEEE International Conference on Industrial Informatics, Cardiff, Wales, 2009,* pp. 475-480.

Pham, D. T., & Koç, E. 2010. Design of a Two-dimensional Recursive Filter Using the Bees Algorithm. *International Journal of Automation and Computing*, *7*(3), 399–402.

Pham, D. T., Koç, E., Lee, J. Y., & Phrueksanant, J. 2007b. Using the Bees Algorithm to Schedule Jobs for a Machine. In *Proceedings of Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance, June 2007,* pp.430-439.

Pham, D. T., Suarez-Alvarez, M. M., & Prostov, Y. I. 2011. Random Search with k-prototypes Algorithm for Clustering Mixed Datasets. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *467*(2132), 2387–2403.

Pham, Q. T., Pham, D. T., & Castellani, M. 2012. A Modified Bees Algorithm and a Statistics-based Method for Tuning its Parameters. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *226*(3), 287–301.

Phonrattanasak, P. 2011. Optimal Placement of Wind Farm on the Power System Using Multiobjective Bees Algorithm. In *Proceedings of the World Congress on Engineering 2011*, *Vol. II*, pp. 4–8.

Phonrattanasak, P., Miyatake, M., & Sakamoto, O. 2013. Optimal Location and Sizing of Solar Farm on Japan East Power System Using Multiobjective Bees Algorithm. In *2013 IEEE Energytech, Cleveland, OH, 2013,* pp. 1-6.

Poli, R., Langdon, W. B., & McPhee, N. F. 2008. *A Field Guide to Genetic Programing*. *Wyvern*. Retrieved from http://www.essex.ac.uk/wyvern/2008-04/Wyvern April 08 7126.pdf

Polratanasuk, P., Mesacharoenwong, P., Anantasate, S., & Leeprechanon, N. 2010. Solving Optimal Power Flow Problem Using Parallel Bee Algorithm. In *6th WSEAS International Conference on Remote Sensing (Remote'10)*, pp. 60–64.

Rao, R. V., Savsani, V. J., & Vakharia, D. P. 2011. Teaching–learning-based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems. *Computer-Aided Design*, *43*(3), 303–315.

Sato, T., & Hagiwara, M. 1997. Bee System: Finding Solution by a Concentrated Search. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, 1997,* pp. 3954-3959.

Sayadi, F., Ismail, M., Misran, N., & Jumari, K. 2009. Multi-Objective Optimization Using the Bees Algorithm in Time-Varying Channel for MIMO MC-CDMA Systems. *European Journal of Scientific Research*, *33*(3), 411–428.

Sayarshad, H. R. 2009. Using Bees Algorithm for Material Handling Equipment Planning in Manufacturing Systems. *The International Journal of Advanced Manufacturing Technology*, *48*(9-12), 1009–1018.

Seeley, T. D. 1995. *The Wisdom of The Hive: The Social Physiology of The Honey Bee Colonies*. Harvard University Press.

Seeley, T. D., & Visscher, P. K. 2004. Group Decision Making in Nest-Site Selection by Honey Bees. *Apidologie*, *35*(2), 101–116.

Shafia, M. A., Rahimi, M. R., & Tavakolian, R. 2011. A Hybrid Algorithm for Data Clustering Using Honey Bee Algorithm , Genetic Algorithm and K-Means Method. *Journal of Advanced Computer Science and Technology Research*, *1*, 110–125.

Shatnawi, N., Faidzul, M., & Sahran, S. 2013b. Optimization of Multilevel Image Thresholding Using the Bees Algorithm. *Journal of Applied Sciences*, *13*(3), 458–464.

Shatnawi, N., Sahran, S., & Mohammad Faidzul. 2013a. A Memory Based Bees Algorithm: An Enhancement. *Journal of Applied Sciences*, *13*(3), 497–502.

Shi, Y., & Eberhart, C. R. 1998. A Modified Particle Swarm Optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pp. 69–73.

Socha, K., & Dorigo, M. 2008. Ant Colony Optimization for Continuous Domains. *European Journal of Operational Research*, *185*(3), 1155–1173.

Spendley, W., Hext, G. R., & Himsworth, F. R. 1962. Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics*, *4*(4), 441–461.

Storn, R., & Price, K. 1997. Differential Evolution–A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Stutzle, T. G. 1998. *Local Search Algorithms for Combinatorial Problem: Analysis, Improvements, and New Applications*. *Darmstadt University of Technology*.

Sumpavakup, C., Srikun, I., & Chusanapiputt, S. 2012. A Solution to Multi-Objective Optimal Power Flow Using Hybrid Cultural-Based Bees Algorithm. *In 2012 Asia-Pacific Power and Energy Engineering Conference, Shanghai, 2012*, pp. 1-4.

Sung, H. J. 2003. Queen-Bee Evolution for Genetic Algorithms. *Electronics Letters*, *39*(6), 575.

Tapkan, P., Özbakır, L., & Baykasoğlu, A. 2011. Bees Algorithm for Constrained Fuzzy Multi-objective Two-sided Assembly Line Balancing Problem. *Optimization Letters*, *6*(6), 1039–1049.

Teodorovic, D., & Dell'Orco, M. 2005. Bee Colony Optimization - A Cooperative Learning Approach To Complex Transportation Problems. In *Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT Advanced OR and AI Methods in Transportation*, pp. 51–60.

Teodorović, D., Lucic, P., Markovic, G., & Dell'Orco, M. 2006. Bee Colony Optimization: Principles and Applications. In *8th Seminar on Neural Network Applications in Electrical Engineering*, pp. 151–156.

Tsai, H-C. 2013. Integrating Artificial Bee Colony and Bees Algorithm for Solving Numerical Function Optimization. *Neural Computing and Applications*, *25*(3-4), 635–651.

Tsai, H-C. 2014a. Integrating the Artificial Bee Colony and Bees Algorithm to Face Constrained Optimization Problems. *Information Sciences*, *258*, 80–93.

Tsai, H-C. 2014b. Novel Bees Algorithm: Stochastic Self-Adaptive Neighborhood. *Applied Mathematics and Computation*, *247*, 1161–1172.

Vejdannik, M., & Sadr, A. 2016. Automatic Microstructural Characterization and Classification Using Dual Tree Complex Wavelet-based Features and Bees Algorithm. *Neural Computing and Applications*.

Wang, S. 2009. Solving Aircraft-Sequencing Problem Based on Bee Evolutionary Genetic Algorithm and Clustering Method. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, Chengdu, 2009,* pp. 157-161.

Wang, X., Contreras, A. F. G., Ceberio, M., Hoyo, C. Del, Gutierrez, L. C., & Virani, S. 2012. Interval-based Algorithms to Extract Fuzzy Measures for Software Quality Assessment. In *2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), Berkeley, CA, 2012,* pp. 1-6.

Wang, X., Cummins, J., & Ceberio, M. 2011. The Bees Algorithm to Extract Fuzzy Measures For Sample Data. In *2011 Annual Meeting of the North American Fuzzy Information Processing Society, El Paso, TX, 2011,* pp. 1-6.

Wolpert, D. H., & Macready, W. G. 1997. No Free Lunch Theorems For Optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Xu, S., Yu, F., Luo, Z., Ji, Z., Pham, D. T., & Qiu, R. 2011. Adaptive Bees Algorithm-Bioinspiration from Honeybee Foraging to Optimize Fuel Economy of a Semi-Track Air-Cushion Vehicle. *The Computer Journal*, *54*(9), 1416–1426.

Xu, W., Tian, S., Liu, Q., Xie, Y., Zhou, Z., & Pham, D. T. 2015. An Improved Discrete Bees Algorithm for Correlation-Aware Service Aggregation Optimization in Cloud Manufacturing. *International Journal Advance Manufacturing Technology*, *82*(422), 1–12.

Xu, W., Zhou, Z., Pham, D. T., Liu, Q., Ji, C., & Meng, W. 2012. Quality of Service in Manufacturing Networks: A Service Framework and Its Implementation. *The*

International Journal of Advanced Manufacturing Technology, *63*(9-12), 1227–1237.

Yang, C., Chen, J., & Tu, X. 2007. Algorithm of Fast Marriage in Honey Bees Optimization and Convergence Analysis. In *2007 IEEE International Conference on Automation and Logistics, Jinan, 2007*, pp. 1794-1799.

Yang, X. S. 2010. Firefly Algorithm. In *Engineering Optimization: An Introduction with Metaheuristic Applications* (pp. 221–230). John Wiley & Sons, Inc.

Yang, X. S. 2005. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. In *In Artificial intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Lecture Notes in Computer Science,* Vol.3562. Berlin Heidelberg: Springer-Verlag, pp.317-323

Yuce, B., Mastrocinque, E., Lambiase, A., Packianather, M. S., & Pham, D. T. 2014. A Multi-Objective Supply Chain Optimisation Using Enhanced Bees Algorithm With Adaptive Neighbourhood Search and Site Abandonment Strategy. *Swarm and Evolutionary Computation*, *18*, 71–82.

Yuce, B., Packianather, M., Mastrocinque, E., Pham, D. T., & Lambiase, A. 2013. Honey Bees Inspired Optimization Method: The Bees Algorithm. *Insects*, *4*(4), 646–662.

Yuce, B., Pham, D. T., Packianather, M. S., & Mastrocinque, E. 2015. An Enhancement to the Bees Algorithm with Slope Angle Computation and Hill Climbing Algorithm and its Applications on Scheduling and Continuous-type Optimisation Problem. *Production & Manufacturing Research*, *3*(1), 3–19.

Zaeri, R., Ghanbarzadeh, A., Attaran, B., & Zaeri, Z. 2011. Fuzzy Logic Controller Based Pitch Control of Aircraft Tuned With Bees Algorithm. In *The 2nd International Conference on Control, Instrumentation and Automation, Shiraz, 2011,* pp. 705-710.

Zambrano-Bigiarini, M., Clerc, M., & Rojas, R. 2013. Standard Particle Swarm Optimisation 2011 at CEC-2013: A Baseline for Future PSO Improvements. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pp. 2337–2344.

Zarea, H., Moradi Kashkooli, F., Mansuri Mehryan, A., Saffarian, M. R., & Namvar Beherghani, E. 2013. Optimal Design of Plate-Fin Heat Exchangers By a Bees Algorithm. *Applied Thermal Engineering*, *69*(1-2), 267–277.

Zhang, J., Zhang, Z. H., Lin, Y., Chen, N., Gong, Y. J., Zhong, J. H., & Shi, Y. H. 2011. Evolutionary Computation Meets Machine Learning: A Survey. *IEEE Computational Intelligence Magazine*, *6*(4), 68–75.

Zhou, Z., Xie, Y., Pham, D. T., Kamsani, S., & Castellani, M. 2015. Bees Algorithm for Multimodal Function Optimisation. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *0*(0), 1–18.