A Study of Search Neighbourhood in the Bees Algorithm

A thesis submitted to Cardiff University in the candidature for the degree of

Doctor of Philosophy

By

Siti Azfanizam Ahmad, Bc. Science.

Manufacturing Engineering Centre

School of Engineering

Cardiff University

United Kingdom

January 2012

ABSTRACT

The Bees Algorithm, a heuristic optimisation procedure that mimics bees foraging behaviour, is becoming more popular among swarm intelligence researchers. The algorithm involves neighbourhood and global search and is able to find promising solutions to complex multimodal optimisation problems. The purpose of neighbourhood search is to intensify the search effort around promising solutions, while global search is to enable avoidance of local optima.

Despite numerous studies aimed at enhancing the Bees Algorithm, there have not been many attempts at studying neighbourhood search. This research investigated different kinds of neighbourhoods and their effects on neighbourhood search.

First, the adaptive enlargement of the search neighbourhood was proposed. This idea was implemented in the Bees Algorithm and tested on a set of mathematical benchmarks. The modified algorithm was also tested on single objective engineering design problems. The experimental results obtained confirmed that the adaptive enlargement of the search neighbourhood improved the performance of the proposed algorithm.

Normally, a symmetrical search neighbourhood is employed in the Bees Algorithm. As opposed to this practice, an asymmetrical search neighbourhood was tried in this work to determine the significance of neighbourhood symmetry. In addition to the mathematical benchmarks, the algorithm with an asymmetrical search neighbourhood was also tested on an engineering design problem. The analysis verified that under certain measurements of asymmetry, the proposed

algorithm produced a similar performance as that of the Bees Algorithm. For this reason, it was concluded that users were free to employ either a symmetrical or an asymmetrical search neighbourhood in the Bees Algorithm.

Finally, the combination of adaptive enlargement and reduction of the search neighbourhood was presented. In addition to the above mathematical benchmarks and engineering design problems, a multi-objective design optimisation exercise with constraints was selected to demonstrate the performance of the modified algorithm. The experimental results obtained showed that this combination was beneficial to the proposed algorithm.

ACKNOWLEDGEMENTS

I would like to thank Professor Duc Truong Pham for the countless supervision and help that he offered from day one of my PhD until this thesis is successfully produced. Also, I am in debt to Dr. Marco Castellani and Dr. Ang Mei Choo for reviewing my work and sharing their valuable knowledge.

In addition, without assistance of Dr. Michael Packianather, Dr. Ebu Bekir Koc and Dr. Ji Young Lee, this thesis might not be possible to produce. Not to forget is the rest of the BayBees Team for making my study life in Cardiff a memorable one.

Also, special thanks to Malaysian Government especially Ministry of Higher Education (MOHE) and Universiti Putra Malaysia (UPM) for sponsoring my studies and family while I am on study leave.

Very much appreciation is also presented to my loving parents, Mr. Ahmad Salleh and Mrs. May Kalsom Che Kob and parents-in-law, Mr. Zahari Mohd. Noor and Mrs. Wan Jah Wan Jusoh who were always by my side. Thanks also to my dear brothers; Amirul, Afiq, Azim and Aisar for the jokes and laughter even though we are apart.

Last but not least, I thank my beloved husband, Mohd. Amiri Zahri for always supporting me; mentally, emotionally and physically during this four years journey and our little daughter, Ruby Nafeesa for turning my every long day to a joyful one.

Specially dedicated to my Yayang, Ruby, Maa, Abah, Mak, Ayah, Long, Chik, Ngah and Adik.

With an ocean of thanks and love.

DECLARATION AND STATEMENTS

DECLARATION

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

Signed ……. (Siti Azfanizam Ahmad) Date…….

STATEMENT 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD).

Signed ……. (Siti Azfanizam Ahmad) Date …….

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ……. (Siti Azfanizam Ahmad) Date …….

STATEMENT 3

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

Signed ……. (Siti Azfanizam Ahmad) Date …….

Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

SI	Swarm Intelligence
GA	Genetic Algorithm
AS	Ant System
ACO	Ant Colony Optimisation
TSP	Travelling Salesman Problem
PSO	Particle Swarm Optimisation
SiC-PSO	Simple Particle Swarm Optimisation
CS	Cuckoo Search
GSO	Glowworm Swarm Optimisation
FA	Firefly Algorithm
IWO	Invasive Weed Optimisation
MBO	Marriage in Honey-Bees Optimisation
FMBO	Fast Marriage in Honey-Bees Optimisation
HBO	Honey-Bees Optimisation
HBMO	Honey-Bees Mating Optimisation
VBA	Virtual Bee Algorithm
BCO	Bee Colony Optimisation
MANETs	Mobile Ad Hoc Networks
PI	Path Integration
ABC	Artificial Bee Colony
BA	The Bees Algorithm
n	number of scout bees
m	number of selected sites
nsp	number of bees recruited for each selected site
e	number of elite sites
nep	number of bees recruited for each elite site
ngh	neighbourhood size
EEDP	Environmental/Economic Power Dispatch Problems

JSSP	Job Shop Scheduling Problem
PCB	Printed Circuit Board
kg	kilogram
s	second
NFL	No Free Lunch
TRIZ	Teoriya Resheniya Izobretatelskikh Zadatch
BA-NE	the BA with adaptive neighbourhood enlargement
nsp_{max}	the last forager bee sent to a selected site
nep_{max}	the last forager bee sent to an elite site
EA	Evolutionary Algorithm
Std. Dev.	standard deviation
Avg.	average
D	dimension
S	significant
NS	not significant
UPSOm	Unified Particle Swarm Optimisation (with mutation)
NA	not available
SCA	Social and Civilisation Algorithm
BA-AN	the BA-NE with asymmetrical search neighbourhood
ED	Euclidean Distance
BA-NER	the BA-NE with neighbourhood reduction
ci	consecutive iteration
nr	neighbourhood reduction

LIST OF SYMBOLS

$f(x)$	objective function
x_i	the design parameters
i	index of bee
j	index of axis
ngh_i	neighbouring function
f_{min}	the best solution found
f_{opt}	the known global optimum
$d(p,q);\ d(q,p)$	distance between points p and q
r	distance of final solution from the global optimum
r_{calib}	distance of final solution that captured within the calibrated search space from the global optimum

Chapter 1

INTRODUCTION

1.1. Background

Over the years, swarm intelligence has inspired scientists to develop population-based algorithms to deal with complex optimisation problems. Among the most common population-based optimisation algorithms are the Genetic Algorithm, Particle Swarm Optimisation and Ant Colony Optimisation. The Genetic Algorithm is based on biological evolution and adaptation in nature, while the Particle Swarm Optimisation algorithm imitates the actions of flying agents keeping themselves in the air alongside other members. Ant Colony Optimisation is inspired by the ants' foraging behaviour where they tend to choose the shortest route that links the food source and their nest. In addition to these algorithms, bees-inspired algorithms are being also developed and they are emulating various behaviours of the bees. The Bees Algorithm (Pham et al., 2005), which imitates the foraging behaviour of honey bees, is one of the examples of a bees-inspired algorithm. The algorithm has been widely applied to solve many complex optimisation problems and received a number of improvements.

Despite numerous attempts to improve the performance of the Bees Algorithm, limited attention has been paid to the study of its parameters and how they affect the algorithm. In the standard version, the algorithm employs six parameters that need to be tuned and one of them is the neighbourhood size. Neighbourhood size

refers to the exploitation area of the forager bees. The exploitation area is designed to facilitate the forager bees in finding a more promising food source in a short time.

Like the Bees Algorithm, other swarm-based algorithms also have a set of parameters that need to be carefully tuned, in order to obtain the desired solution of the optimisation problems. (El-Gallad et al., 2002) stated that properly chosen parameter values could positively influence the accuracy of the solutions and time consumed for the search process. In particular, the authors mentioned that the individuals' flying velocities was one of the most important parameters for Particle Swarm Optimisation. The flying velocities could influence the steps taken by the particles. An excessively small step size could cause the particle to get trapped in local optima, while a too large value led to oscillation around a certain position. In the improved version of Particle Swarm Optimisation, the inertia weight affected the behaviour of the algorithm. A larger inertia weight facilitated global search while a smaller one provided fine-tuning (Meissner et al., 2006).

Exploitation is important for a swarm-based algorithm as it is functioning as a 'tool' enabling the population to converge towards a local minimum (MacNish 2007). This exploitation is accomplished by the procedure of neighbourhood search. The quality of the neighbourhood search is highly influenced by the size of the neighbourhood. On one hand, a smaller neighbourhood size could intensify the exploitation effort, while a larger one could reach better solutions quickly. On the other hand, a smaller step (neighbourhood) size might lead to a slow convergence of the algorithm, while a larger one might miss out the spaced peaks

(Krishnanand and Ghose 2009). Furthermore, in (Sundareswaran and Sreedevi 2008), the distance of flight was a key factor and it significantly affected the time for the algorithm to reach the global optimum. Similarly in this research, the neighbourhood size greatly influenced the performance of the algorithm.

1.2. Motivations

Currently, the neighbourhood size employed in the Bees Algorithm is fixed throughout the optimisation. It was anticipated that an adaptive and large search neighbourhood could help the bees to reach the global optimum faster than a fixed and small neighbourhood size. For that reason, the first motivation for this work was to study the adaptive enlargement of the search neighbourhood in the Bees Algorithm.

The first proposed idea did not always succeed in enabling the Bees Algorithm to reach the global optimum and increasing the size of the search neighbourhood actually decreased the probability of finding the global optimum, which is a form of 'contradiction'. The cause of this failure was referred to TRIZ (Altshuller 2001), a theory that provided a list of inventive principles to solve contradictions in problems. The application of TRIZ suggested the adoption of an asymmetrical search neighbourhood. Coincidently, an asymmetrical search neighbourhood had never been employed in the Bees Algorithm. Thus, the second motivation for this work was to investigate the effects of an asymmetrical search neighbourhood on the performance of the Bees Algorithm.

The ‘neighbourhood shrinking’ method was introduced to improve the performance of the Bees Algorithm (Ghanbarzadeh 2007). This method was followed by ‘site abandonment’ if the former procedure failed to yield to improvement. In contrast to the ‘neighbourhood shrinking’ method, the third motivation was to study the combination of the adaptive enlargement and reduction of the search neighbourhood in the Bees Algorithm. Instead of applying ‘site abandonment’, the reduced neighbourhood size was increased back to the initial size if there was no improvement in the solutions.

In choosing the size of the search neighbourhood in the Bees Algorithm, the aim is to achieve a robust algorithm. Robustness can be interpreted in many ways (Beyer and Sendhoff 2006):

- The ability of an optimisation algorithm to adapt to different optimisation scenarios (e.g., different classes of optimisation problems).
- The sensitivity of the algorithm’s performance corresponding to algorithm specific parameter setting.
- Robustness in terms of implementation details.
- Robustness in terms of the solution produced by the algorithm, including insensitivity of the final solution to different initialisation.

1.3. Aim and objectives

The overall aim of this research was to prove the hypothesis that neighbourhood size could influence the performance of the Bees Algorithm.

The objectives of this work were:

1. To develop the adaptive enlargement of the search neighbourhood in the Bees Algorithm.
2. To determine whether an asymmetrical search neighbourhood gave positive influence(s) to the Bees Algorithm.
3. To demonstrate the effects of the combination of adaptive enlargement and reduction of the search neighbourhood in the Bees Algorithm.

It should be emphasized that the purpose of this work was not to produce the best algorithm in solving a wide range of problems but rather to understand how the various types of search neighbourhood served in the Bees Algorithm.

1.4. Methods

In carrying out this research, the following methodologies were adopted:

1. Surveying the previous works including behaviours of the swarm in nature and the developments of other algorithms.
2. Implementing the proposed algorithms in C++.
3. Testing the modified Bees Algorithms on a set of mathematical benchmarks and several engineering design problems in order to validate the modifications. The outcomes of the experiments were analysed and compared to the ones obtained by the Bees Algorithm.

1.5. Outline of the thesis

The remainder of the thesis is organised as follows.

Chapter 2 reviews the definition of optimisation and highlights swarm behaviours. The developments of population-based algorithms are also presented. Examples of optimisation problems that were solved by the Bees Algorithm are highlighted. In addition, the No Free Lunch Theorem and TRIZ are briefly described.

Chapter 3 introduces the adaptive enlargement of the search neighbourhood in the Bees Algorithm. The modification is tested on mathematical benchmarks and several engineering design problems. They are single objective problems, with and without constraints.

Chapter 4 presents the types of asymmetrical search neighbourhood and their influences on the Bees Algorithm. The Bees Algorithm with different kinds of asymmetrical search neighbourhoods is tested on mathematical benchmarks and an unconstrained, single objective engineering design problem.

Chapter 5 elaborates the idea of neighbourhood reduction which is combined with adaptive enlargement. The difference between the 'neighbourhood shrinking' method and the proposed idea is highlighted. In addition to the mathematical benchmarks and stated engineering design problems, this method is also tested on a constrained, multi-objective optimisation problem.

Chapter 6 lists the contributions of this research, summarises the conclusions reached and provides suggestions for further research.

Appendices A, **B** and **C** describe the selected engineering design problems.

Appendix D provides a summary of the experimental results of three types of search neighbourhood in comparison against the Bees Algorithm.

Chapter 2

LITERATURE REVIEW

2.1. Preliminaries

In this chapter, Swarm Intelligence and popular swarm-based algorithms are reviewed. Applications that have been accomplished by the Bees Algorithm are revisited. In addition, the No Free Lunch Theorem is highlighted. The main ideas of TRIZ are explained at the end of this chapter.

2.2. Swarm Intelligence

Swarm Intelligence (SI) is an engineering branch and one developed based on the emergent collective intelligence of groups of simple agents (Bonabeau et al., 1999). An ant colony, a flock of birds and a shoal of fish is regarded as a swarm. These groups miraculously do not have a centralised control system. Rather, they use a decentralised system or self-organisation. That means every task is engaged without any central or hierarchical control to direct the individuals into particular tasks. With this system, every individual responds to simple and local information that allows the whole system to function (Gordon 1996). The exchange of information among the individuals is the most essential component in the formation of collective knowledge (Frisch 1953).

Another significant element in SI is the division of labour or task allocation (Anderson and Ratnieks 1999; Seeley 2002). Division of labour means that different tasks are performed simultaneously by specialised individuals. This behavioural specialisation is able to keep the colony functioning efficiently (Seeley and Buhrman 2001). For instance, in the honey bee colony, the forager bees collect the food and bring it back to the hive. Then, the receiver bees are responsible for the storing (Anderson and Ratnieks 1999). This kind of task allocation enables the swarm to adapt to the environment (Karaboga 2005).

The following are three of the swarm behaviours that adopt the decentralised system and division of labour.

2.2.1. Dwelling

The queen and about half of the workers in a colony leave their hive and form a cluster on a nearby branch (Passino and Seeley 2006; Passino et al., 2008). Scout bees fly from the cluster to survey for potential dwelling places. Upon returning to the cluster, the scout bees perform a waggle dance (Seeley et al., 2006). The waggle dance is a process of information exchange and other scouts evaluate the quality of the potential new hive by witnessing the dance. The quality of the nest site is characterised by the level of protection against weather, predators and distance from food source (Janson et al., 2007). In addition, there are six attributes that are considered by the honey bees before choosing a new nest; cavity volume, entrance height, entrance area, entrance direction, entrance position relative to the

cavity floor and presence of combs from a previous colony (Seeley and Buhrman 1999).

2.2.2. Evading predators

A flock of starlings flies in perfect unison and sometimes changes and reforms the flying shape to avoid any predator. While in the air, the starlings are always aware of their neighbours. A gull (predator) may dive into the flock in order to grab a starling. To avoid this threat, the starlings work together, for example by making a sudden change of direction in order to wrong-foot the attackers (2011). Also, a shoal of fish makes compression, hourglass, vacuoles or fountain shape as well as flash expansions in order to avoid a predator (Parrish et al., 2002; Sumpter 2006).

2.2.3. Foraging

During the harvesting season, the honey bees optimise the amount of pollen by recruiting a number of scouts to go to the flower field. The amount of nectar in the field is always fluctuating and represents an unpredictable resource. To cope with this uncertainty, changes in the number of workers are necessary to balance the work load of foragers and others in the hive so that the resources can be exploited efficiently (Anderson and Ratnieks 1999). With a decentralised system, the colony sends the scout bees into the field. The scouts search randomly from one flower patch to another, hoping to find a good food source. When the scouts discover the food source, the scouts evaluate and memorise three pieces of information regarding a flower patch (Frisch 1950):

1. The direction in which it was found
2. Its distance from the hive
3. Its quality rating (fitness)

When the scouts return to the hive, they perform a waggle dance on the dance floor. The dance conveys the information of the direction, distance and quality of the flower patch that they found during the searching process. Other bees witness the dance and assess the information that is delivered. This information helps others to locate the flower patch without using guides or maps. When the dancing process is over, other bees follow the scout bees to the discovered flower patch to gather the food. How the other bees decide to follow a particular scout is not well understood, but it is thought that the recruitment among bees is associated with the quality of the food source (Teodorović 2008).

A flower patch with plentiful nectar and near to the hive is regarded as more promising and would attract more followers. On the other hand, long distance scouting is costly and less preferred by others since there is no assurance that a patch will be discovered (Beekman and Ratnieks 2000). Optimising the number of scouts recruited accordingly will optimise the amount of nectar and pollen to be collected. In other words, a maximum amount of pollen and nectar can be gathered with a minimum of effort invested.

2.3. Population-based algorithms

Population-based algorithms are developed based on the interaction and cooperation of members of a swarm. The following are examples of such algorithms:

2.3.1. Genetic Algorithm

Genetic Algorithm (GA) was introduced by John Holland (Goldberg 1989) and developed based on the genetic processes of biological organisms. The idea was inspired by the Darwinian evolutionary concept, which stated "survivability of the fittest species" (Rahmat-Samii 2007). Each species is evolving in adapting with the changes of the environment and only the fittest can survive. The better a species can adapt to the environment, the higher the level of survival of that species will be.

In GA, each generation consists of a population of binary strings which are called chromosomes. Every time evolution takes place, the new attributes are encoded in the chromosomes. The chromosomes with the highest fitness are always being copied into the next generation. In addition, the *Crossover* operator enables an exchange of substrings between two parents, in order to increase the diversity of the perturbed offspring. The *Selection* operator decides whether or not a new solution should be selected to be a member of the next generation. Meanwhile, the random modification of a new configuration is controlled by the *Mutation* operator. (Digalakis and Margaritis 2002) applied GA to a set of benchmark functions. In addition, there are wide applications of GA including pattern

recognition, cellular automata, biology and medicine (Chaiyaratana and Zalzala 1997).

2.3.2. Ant algorithms

(Dorigo et al., 1996) proposed Ant System (AS) that mimicked the foraging behaviour of an ant colony. The ant colony builds a trail that connects the nest and food source. During travelling back to the nest from the food source and vice versa, pheromones are deposited by the ants. Over time, pheromone trails are formed on the ground. Using these pheromone trails, the other ants are able to navigate towards the nest or food. The more ants that follow a trail, the more pheromone accumulates on it. The trail with a high density of pheromones becomes more attractive to other ants.

AS was applied to the Travelling Salesman Problem (TSP), which is a combinatorial optimisation problem. Also, AS had been tested on continuous problems (Bilchev and Parmee 1995; Socha and Dorigo 2008). The study of AS had led to the development of Ant Colony Optimisation (ACO) (Dorigo et al., 1999). Furthermore, (Dorigo et al., 2000) focused on potential models that derived from the behaviour of real ants and how they had inspired other algorithms for the solution of distributed optimisation and control problems.

2.3.3. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was introduced by Kennedy and Eberhart (Eberhart and Kennedy 1995; Kennedy 1997; Kennedy and Eberhart 1995). PSO

is population-based stochastic optimisation technique and is inspired by the behaviour of a flock of birds. The algorithm consists of a swarm of particles moving in a space. Every particle holds a position and velocity vector representing a candidate solution to the problem. In addition, each particle memorises its own best position found so far and a global best position that is obtained through communication with its neighbour.

The early version of PSO had operated in continuous space but was later adapted to operate in discrete binary variables (Kennedy and Eberhart 1997; Zhong et al., 2007). PSO received large amounts of interest from researchers and there are now improved versions of PSO which are detailed in (Angeline 1998; Hu et al., 2004; Kennedy 2000; Shi and Eberhart 1998). Furthermore, the Simple PSO algorithm (SiC-PSO) was developed to cope with constrained optimisation problems (Cagnina et al., 2008).

2.3.4. Cuckoo Search

Cuckoo Search (CS) was in its day a reasonably new metaheuristic that imitates the breeding behaviour of the cuckoo birds (Yang and Deb 2009; 2010). When the breeding time has come, the cuckoo birds tend to lay their eggs in the nest of other birds. The host birds would either throw the eggs left by the cuckoo out of the nest or decide to leave the nest and build a new home at another place. Further to confuse the host birds, some cuckoo birds were able to produce eggs that look similar to the eggs of chosen host birds. This imitation would ensure that these eggs would be cared for by the host birds and thus increase the cuckoo's

productivity. Once the cuckoo eggs hatched, the cuckoo chick would throw out the host birds' eggs of the nest, which were hatched slightly later than cuckoo eggs. Consequently, the cuckoo birds got more chance to be fed as the number of chicks in the nest became less.

The CS adopted three rules from the breeding behaviour of cuckoo birds, which were:

- Each cuckoo laid one egg at a time. The eggs were left in random nests.
- The nest with a high quality of eggs (solutions) would be carried to the next iterations.
- The number of potential hosts was fixed, and the cuckoo eggs could be found with a probability $p_a \in [0, 1]$. If the host bird discovered that the eggs were not hers, the alien eggs could be thrown away or the host birds simply abandon the nest.

The capability of CS was verified by testing it on a set of mathematical benchmarks and a few engineering design problems. In spite of using only two parameters, which were the population size of the cuckoo and p_a, the experimental results showed that the CS produced better solutions compared to the GA and PSO.

2.3.5. Glowworm Swarm Optimisation

Glowworm Swarm Optimisation (GSO) (Krishnanand and Ghose 2009), was another example of a swarm-based algorithm. The GSO was inspired by the behaviour of glowworms in passing the information to the other members of their group. The information regarding the fitness of the current location of glowworms was encoded into a luminescent quantity, which was called *luciferin*. Each glowworm carried *luciferin* along and it was to be broadcast to other members. A glowworm with more *luciferin* would attract more members. By using sensor range, a glowworm recognised its neighbour and computed its movement. Probabilistically, each glowworm selected a neighbour that owned greater *luciferin* value than hers and moved towards this neighbour. According to the authors' analysis, the GSO only needed two parameters, which were the number of glowworms and the maximum radial range. Unlike other optimisation algorithms, the GSO was tested in capturing the peaks in a series of multi-modal test functions. Experimental results demonstrated that the GSO was able to locate more peaks than Niche-PSO within a specified number of runs.

2.3.6. Firefly Algorithm

Flashing light emitted by fireflies inspired Yang to develop an algorithm, called the Firefly Algorithm (FA) (Yang 2009). Whether to attract mating partners or potential prey, the light intensity complied with the inverse square law. It meant that the amount of light that was visible to the mating partners or potential prey was inversely proportional to their distance from the fireflies. The more apart they were, the less the intensity of light from the fireflies would be. Consequently, the fireflies at such distance would be less attractive to other fireflies or potential

prey. On the other hand, the fireflies at a close distance exhibited bright light and would attract more fireflies to them. In the context of the FA, the brightness of the firefly was associated with the landscape of the objective function. In addition, fireflies were assumed as unisex so that the attraction was not restricted to a particular sex. Conversely, each firefly could attract any other fireflies. The capability of the FA was validated by testing it on a number of mathematical optimisation benchmarks. The experimental results showed that the FA produced a low number of evaluations with small standard deviation, compared to the PSO and GA.

2.3.7. Invasive Weed Optimisation

There was also an algorithm that was developed based on colonising weeds (Mehrabian and Lucas 2006). The weeds, which referred to vigorous and invasive plants, caused a threat to desirable and cultivated crops. The adaptation and robustness of weeds against herbicides had inspired the researchers to develop an algorithm that was named Invasive Weed Optimisation (IWO). With nine parameters, the IWO was tested on a set of mathematical benchmarks and the outcomes were promising.

2.3.8. Bees-inspired algorithms

Marriage in Honey Bees optimisation (MBO) was inspired by the evolution of honey bees. The algorithm, which adopted the mating and breeding behaviour of honey bees, started from a *solitary* colony (single queen without a family) to the development of an *eusocial* colony (one or more queens with a family) (Abbas

2001). To produce a family, the queen needed to mate with the drones, which took place in the air and in a probabilistic manner. The mating flight was initialised when the queen performed a dance. Subsequently, the queen flew and was followed by the drones, which then started mating in the air. During the mating flight, the queen flew at different states and at a certain speed. Probabilistically, the queen mated with the drones that she met at each state. At the beginning of the mating flight, the queen owned an amount of energy. As time progressed, the energy supply was gradually reduced. As a result, the probability to mate with a drone was also low. When the queen's energy met a certain threshold or her *spermatheca* was full, the queen returned to the hive. In the hive, the queen started the breeding process by retrieving randomly the mixture of the sperms that accumulated in the *spermatheca*. The breeding process involved the crossover with the queen's genome and mutation, to guarantee that a new and different brood was produced. The workers were improving the fitness of the broods, before updating the fitness of themselves. The brood with the best fitness replaced the least-fitted queen, while the rest of the broods were killed. At the end of this step, another *eusocial* colony was developed and a new mating flight started. (Haddad et al., 2006) applied the same mating and breeding procedure for water resources optimisation and named it the Honey-Bees Mating Optimisation (HBMO) algorithm.

Fast Marriage in the Honey Bees Optimisation (FMBO) was introduced to improve the calculation process and speed of the MBO, as claimed in (Yang et al., 2007). Instead of using a probabilistic mating condition, the FMBO generated a

drone randomly and mated it with a finite number of queens. This procedure was designed to avoid the local optima. In addition to the crossover and mutation operator, the FMBO also adopted a heuristic operator. The experimental results found that these modifications had saved much computation time relative to the MBO.

The Honey-Bees Optimisation (HBO) algorithm was another model that emulated the mating behaviour of bees (Curkovic and Jerbic 2007). This model also adopted the same probabilistic method as in MBO. In HBO, the amount of energy reduction was expressed in the function of size of *spermatheca*, which was not clearly stated in the implementation of MBO and FMBO. The HBO was applied in finding a collision-free path in the work area containing different shapes and distributed obstacles.

Bee System, an improved version of the GA, was the earliest algorithm that mimicked bees' foraging behaviour (Sato and Hagiwara 1997). As it was an enhanced version of the GA, the *Bee System* involved new operations such as concentrated crossover and the Pseudo-Simplex Method. In the system, when a bee found a feed, it then informed the other bees by performing a dance. Then, they were responsible to carry the feed to the hive. After completing this task, each bee tried to find another feed. The purpose of the *Bee System* was to improve the local search while keeping the ability of the global search of the GA. In contrast to the conventional GA, the global search in the *Bee System*, named as *pop_G*, was performed prior the local search. The purpose of the global search

was to search the space as broadly as possible, in order to minimise the chance of falling into a local optimum. The *Superior Chromosome*, which was the best chromosome found at the end of a number of generations, was selected. It was assumed that the *Superior Chromosome* might contain the global optimum. At the end of this procedure, the *pop_G* was initialised again and the search process was repeated.

(Lučić and Teodorović 2002) proposed another bee system and applied it to solve complex problems in traffic and transportation. They named the model the Bees System and aimed to deal with the Travelling Salesman Problem. In their version, a number of nodes were scattered in a network and the hive was located at one of the nodes. The artificial bees needed to collect as much nectar as possible, by flying along a certain link. The amount of nectar was inversely proportional to the length of the link that connected the two nodes. Hence, to maximise the quantity of the nectar, the artificial bees needed to locate the shortest link. After a pre-set period, the hive was moved to another new position. The artificial bees then collected the nectar from this new location and the steps were repeated. One change of the hive position corresponded to one iteration in the searching process. Furthermore, every iteration contained a number of stages. In each stage, the artificial bees flew to nodes, before flying back to the hive. In the hive, a decision making process was performed. After completing this process, the artificial bees would choose whether to abandon the food source or forage in it. If they decided to forage, they might recruit others to the food source or fly back to the source without recruiting nest mates.

The Bee Colony Optimisation (BCO) was proposed by (Teodorović and Dell'Orco 2005) and designed to solve combinatorial optimisation problems (Chong et al., 2006; Wong et al., 2008). Two main elements in the BCO were *forward pass* and *backward pass*. A partial solution was generated when the bees performed a *forward pass*, which was accomplished by the combination of individual exploration and collective experience from the past. A *backward pass* was performed when they returned to the hive. The step was followed by the *decision-making* process, which involved an information *exchange* among the bees. The information regarding the quality of the partial solution was delivered to other members and compared to the one acquired by the individual bee. Based on the quality of the partial solution, the bees decided to commence three different tasks; abandon the food source, forage in it without recruiting others or forage in it and recruiting nest mates, as described in the Bees System. In addition, the *loyalty* parameter was introduced to control the number of bees returning to the previously discovered partial solutions. The bees resumed the process by performing a second *forward pass* and *backward pass* before returning to the hive. If the bees found one or more feasible solutions, the optimisation was considered as having completed one iteration. Since there were a number of stopping criterion that could possibly be made and more questions/ options that must be faced by the bees during the *decision-making* process, the study of BCO led to the development of the Fuzzy Bees System (Teodorović and Dell'Orco 2005).

(Nakrani and Tovey 2004) found that the server allocation to collect revenue in internet housing centres resembled the allocation of foragers to collect nectar in honey bee colonies. For instance, the waggle dance, dance floor, waggle dance duration, flower patch location, following a waggle dance and waggle dancing resembling advertisement, advert board, advertisement duration, web-site identifier, reading an advertisement and posting an advertisement, respectively. Besides these similar key features, they also claimed that the rapid change of request stream, the significant downtime cost of reallocation and the distributed nature of the process were parallel to the performance strengths of honey bee foraging. Due to this fact, they employed the honey bee forager algorithm to work on the server allocation problem.

AntNet was proposed by Di Caro and Dorigo (Di Caro and Dorigo 1998). In AntNet, there were two ant agents involved; *Forward_Ant* and *Backward_Ant*. The *Forward_Ant* was responsible for estimating queuing time without waiting inside data packet queues and they had a stack memory that recorded the address and entrance time of each node along its path. Meanwhile, the *Backward_Ant* which was created by the *Forward_Ant*, visited the same nodes as the *Forward_Ant*, in reverse order. Later on, BeeHive was developed by Wedde and his colleagues (Wedde and Farooq 2005; Wedde et al., 2004). In contrast to the AntNet, this model, which was inspired by the communication activities of honey bees, did not need to be equipped with a stack memory to perform the tasks. In addition, this model eliminated the use of backward moving agents. The forward moving agents were able to calculate the travel duration from the source to a

given node. There were two types of agents in the BeeHive; *short distance bee agents* and *long distance bee agents*. The former agents gathered and distributed routing information in the neighbourhood of their source node. Meanwhile, the latter agents were responsible for gathering and distributing the information to all nodes of the network. A network was split into two; *Foraging Regions* and *Foraging Zones*. Each node was restricted to only one *Foraging Region* and each *Foraging Region* has a representative node. Meanwhile, a *Foraging Zone* of a node comprised all the nodes from which a replica of an agent could get in touch with this node (Wedde and Farooq 2005).

BeeAdHoc, another routing algorithm was proposed by (Wedde et al., 2005) and motivated by the success of the BeeHive. This algorithm was introduced for energy efficient routing in mobile ad hoc networks. The challenge in the Mobile Ad Hoc Networks (MANETs) domain was to design a routing algorithm that was energy efficient as well as having the same performance as that of other algorithms. This model consisted of four types of agents; *packers*, *scouts*, *foragers*, and *swarms*. The *packers* played a role as a food-storer bee, while the *scouts* were looking for new routes from their initial position to a destination node. *Foragers* accepted the data packets from *packers* and brought them to their destination. Meanwhile, the *swarm* acted as problem solver when a protocol could not provide an implicit return to its source node. In addition, each node in MANETs contained a hive. Each hive consisted of three sections; *packing floor*, *entrance* and *dance floor*. The *packing floor* and *entrance* were an interface to the

higher and lower levels respectively, while the *dance floor* received routing decisions.

The Virtual Bee Algorithm (VBA) was inspired by a swarm of virtual bees where it began with bees wandering randomly in the search space (Yang 2005). For function optimisation, the VBA initially created a population of virtual bees, where each bee was associated with a memory bank. Then, the functions of optimisation (objectives) were converted into virtual food. The direction and distance of the virtual food were then defined. The bees updated a population of individuals to new positions for virtual food searching and the direction by performing a waggle dance. The solution of the function optimisation was associated with the highest mode in the number of virtual bees or intensity (frequency) of visiting bees. The results obtained were then decoded to match the solution to the problem.

(Lemmens et al., 2007; Lemmens et al. 2008) introduced a non-pheromone-based algorithm which combined the recruitment and navigation strategies of the biological bees. The recruitment strategies were employed to communicate the search experiences to the rest of the bees in the colony. These strategies involved the dancing process, where the information of distance and direction towards a destination were delivered to other members. Meanwhile, the navigation strategies were to find undiscovered worlds. Instead of pheromone, the bees used a strategy named Path Integration (PI), which adopted a simple approximation for navigation. By PI, they were able to figure out their current position from the previous trajectory continuously. As a result, they managed to navigate back to

their initial point by selecting the direct route, rather than retracing their outbound path. The algorithm consisted of three functions; ManageBeesActivity(), CalculateVectors() and DaemonActions(). The ManageBeesActivity() dealt with agents' activity, with each activity corresponding to any of six internal states; 'AtHome', 'StayAtHome', 'Exploitation', 'Exploration', 'HeadHome' and 'CarryingFood'. An agent with the state 'AtHome' indicated that the agent was located in the hive and to decide which new state it would embark. Meanwhile, 'StayAtHome' implied that the agent stayed at home if there was no previous search experience available. An agent with the state 'Exploitation' and 'Exploration' indicated that the agent was exploiting previous search experiences and exploring its environment, respectively. An agent with the state 'HeadHome' indicated that the agent was returning home without food. In contrast to the 'HeadHome' state, an agent with the state 'CarryingFood' indicated that the agent was bringing the food back to the nest. The function of CalculateVector() was to manage the PI vectors for each agent. The third function, the DaemonActions() was an optional one and used to implement centralised actions such as collection of global information. For example, this information was used by the agents to decide whether to dance or not. Experiments on this algorithm were conducted in a simulation environment, named BeeHave. The experimental results found that the non-pheromone-based algorithms were not only more efficient when foraging, but also required less computation time to complete the task and were less adaptive than ant-inspired algorithms.

Karaboga and Basturk introduced the Artificial Bee Colony (ABC) algorithm (Karaboga 2005; Karaboga and Basturk 2007a, 2008). The algorithm, which mimicked the foraging behaviour of bees, comprising employed bees, onlooker bees and scouts. Employed bees flew to a field and returned to the hive with a piece of memory. The memory, which contained the information regarding the food source, was delivered to the onlookers who were waiting on the dance area. Based on information received, the onlooker bees decided whether to follow the employed bees. Meanwhile, the scouts were the ones performing random search. In this algorithm, half of the colony was set as employed bees and the other half were onlookers. Furthermore, one food source was associated with one employed bee. Once the food source got exhausted, the employed bee that was associated with it became a scout. The capability of the ABC had been tested on an artificial neural networks problem (Karaboga et al., 2007). A modified ABC algorithm was introduced later on to adapt to constrained optimisation problems (Karaboga and Akay 2011; Karaboga and Basturk 2007a).

More highlights on the bees-inspired algorithm are available in (Bitam et al., 2010; Karaboga and Akay 2009).

2.4. The Bees Algorithm

The Bees Algorithm (BA) was developed by a group of researchers at the Manufacturing Engineering Centre, Cardiff University (Pham et al. 2006a). This

algorithm emulated the behaviour of honey bees in foraging for pollen and nectar. The algorithm required six parameters, namely the number of scout bees (n), number of selected sites (m), number of top-ranking (elite) sites among the m selected sites (e), number of bees recruited for each non-elite site (nsp), number of bees recruited for each elite site (nep), and neighbourhood size (ngh). The optimisation process started with n scout bees randomly spread across the solution space. Each scout bee was associated with a possible solution to the problem. The solutions were evaluated and ranked in descending order of the fitness, and the best m sites were selected for neighbourhood search.

In the neighbourhood search procedure, more forager bees were sent in the neighbourhood of the elite (e) sites, and fewer bees around the non-elite (m-e) sites. According to this strategy, the foraging effort was concentrated on the very best (i.e., elite) solutions. That is, nep bees were sent to forage around the elite sites, while the area around the non-elite locations was exploited by nsp bees. Within the given neighbourhood area (i.e., flower patch size), some of the newly generated solutions were expected to be better than that found by the scout bees.

In the global search procedure, the unselected scout bees (n-m) were used to explore at random the solution space. This kind of search was to avoid bees being trapped at local optima. At the end of each cycle, a new list of scout bees was formed, comprising the fittest solutions from each neighbourhood (neighbourhood search results), and the new randomly generated solutions (global search results). This list would be sorted in the next iteration and used for a new phase of optimisation. The combination of exploitative (neighbourhood) and explorative

(global) search would be able to capture the best solution quickly and efficiently. These steps were repeated until the stopping criterion was met. The pseudocode of the BA is shown in Figure 2.1 and the algorithm flowchart as in Figure 2.2.

Since the algorithm was introduced, there have been several attempts to improve the performance of the BA, for instance, interpolation and extrapolation, 'neighbourhood shrinking' and 'sites abandonment' (Ghanbarzadeh 2007).

2.5. Applications

The BA was tested on various types of problems which could be categorised into two groups: continuous and combinatorial problems. Below are examples of each group:

2.5.1. Continuous

- Mathematical benchmarks (Ghanbarzadeh 2007; Koc 2010; Sholedolu 2009)
- Mechanical design (Ang et al. 2009; Pham and Ghanbarzadeh 2007)
- Wood defect classification (Pham and Haj Darwish 2010; Pham et al., 2007c; Pham et al., 2006b).
- Environmental/Economic Power Dispatch Problems (EEDP) (Lee and Haj Darwish 2008)
- Chemical engineering process (Pham et al., 2008)

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

Figure 2.1 Pseudocode of the BA (Ghanbarzadeh 2007)

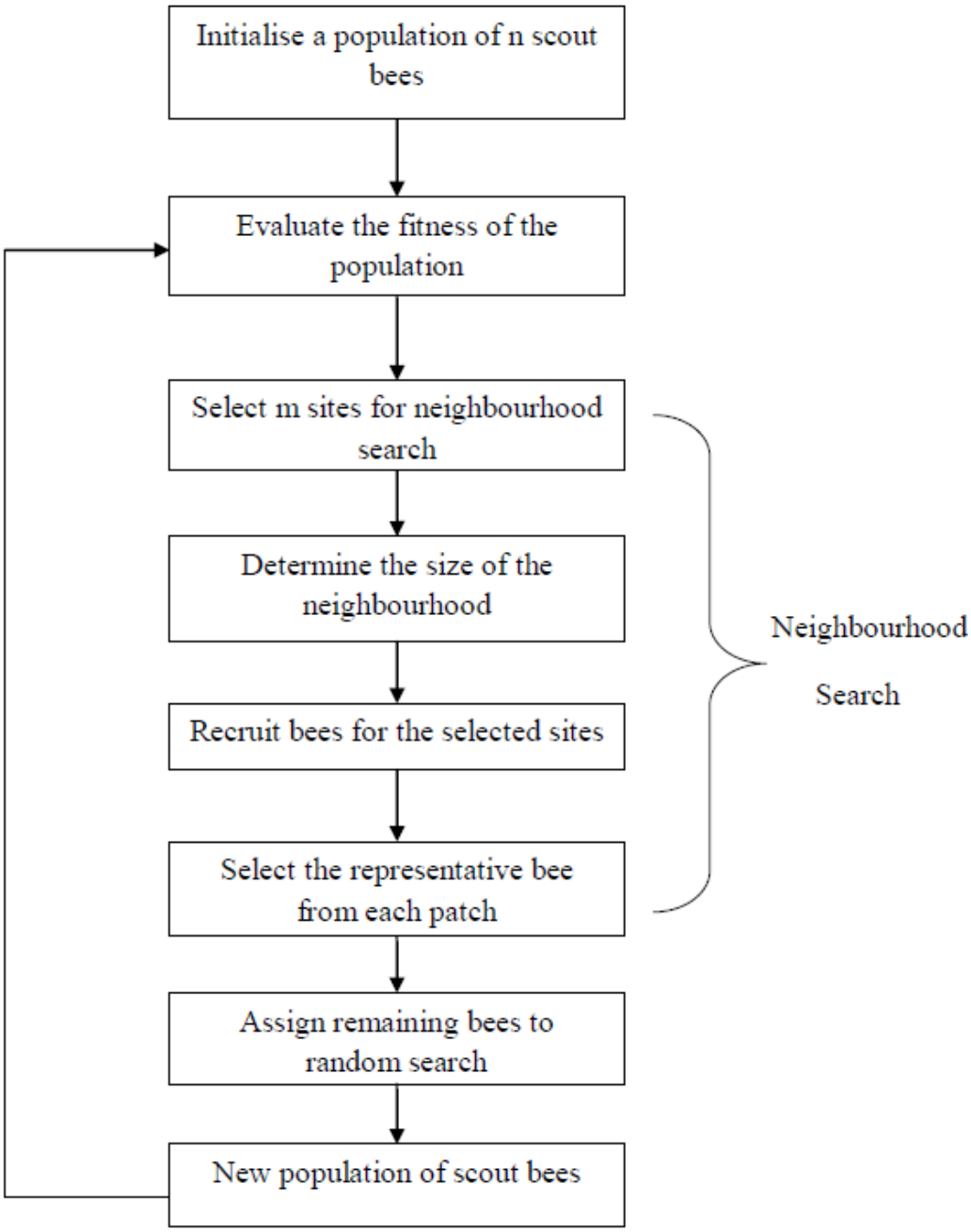


Figure 2.2 Flowchart of the BA

2.5.2. Combinatorial

- Job Shop Scheduling Problem (JSSP) (Pham et al., 2007b)
- Manufacturing Cell Formation (Pham et al., 2007a)
- Printed Circuit Board (PCB) problem (Ang et al., 2010)

2.6. No Free Lunch Theorem

The No Free Lunch (NFL) Theorem was introduced by (Wolpert and Macready 1997). The theory stated that there was no such algorithm that performed better than others in solving all classes of problems (e.g., multimodal, unimodal). In other words, if algorithm A performed better than algorithm B in some class of problems, then algorithm B performed better than algorithm A in some other class of problems. On average, each algorithm produced similar performance in respect to other algorithms. In addition, the performance of an algorithm on a set of benchmarking problems did not guarantee giving similar performance on a different class of problems (MacNish 2007).

Even though (Yang 2005) was not directly discussing the NFL Theorem, he mentioned that parameter tuning could be difficult for any considered problem. Despite the fact that numerous evolutionary-based algorithms had been developed, the best choice of algorithm still depended on the type and characteristics of the problem concerned. (MacNish 2007) added that as well as comparing the algorithms proposed, it was also beneficial to understand what

properties of the algorithms were most successful so that improved algorithms could be developed.

Moreover, (Kennedy and Spears 1998) stated that experimental results might not be generalised from the test problems used. This was because an algorithm might be carefully tuned in order to beat other algorithms on particular problems. On the other hand, the algorithm might produce poor performance on other problems. Complying with the NFL Theorem and claims above, this work was to study the effect of the proposed idea and identify its strengths and weaknesses, rather than producing an algorithm that is capable of solving any kind of problem.

2.7. TRIZ

TRIZ is the acronym for 'Teoriya Resheniya Izobretatelskikh Zadatch' in Russian, whereas in English it means 'Theory of Inventive Problem Solving'. It was developed by Genrich Altshuller starting in 1946 and is now being used extensively in various fields such as engineering (Ang et al., 2010), service quality (Su and Lin 2008) and software development (Mann 2008). (Kim et al., 2009) stated that TRIZ was a problem solving method that was not based on intuition but one relying on logic and data. In other words, it was a theory that considered problems and proposed solutions that were derived from previous successful design solutions.

To solve problems with TRIZ, users needed to identify the improving and worsening features before mapping those features onto the TRIZ contradiction matrix. The classical TRIZ contradiction matrix listed 39 improving and worsening features, and proposed 40 inventive principles to solve design problems. Each suggested inventive principle, however, was not subject to a single definition. On the other hand, the interpretation of the solution was very abstract, ambiguous and subjective (Ang et al., 2010).

Over the years, the classical TRIZ has been studied and extended to embrace broader features and inventive principles. This study led to the development of a new TRIZ matrix, which contained up to 48 features, with 77 inventive principles (Mann et al., 2003). Several years later, the study of TRIZ had brought to another matrix, which comprised 50 features, with 82 solutions (Mann 2009). The increasing number of features and inventive solutions did not necessarily guarantee a solution to a problem. On the other hand, it did minimise the number of trials and error of the solution finding process (Duran-Novoa et al., 2011). Considering the problem from a different perspective might also derive a system in different and stronger ways. Furthermore, the higher the level of the disagreement, the more obvious it would be and become easier to remove it (Altshuller 2001).

The BA was not a technical engineering problem. It was a virtual tool that was used to solve optimisation problems that often occur in the engineering world. Since the idea of TRIZ now covered a wide range of fields, it was reasonable to look at the problem faced by the algorithm from a TRIZ perspective.

2.8. Summary

This chapter briefly described swarm-based and population-based optimisation algorithms. It also highlighted the No Free Lunch Theorem and TRIZ. This study provided information and background to the contents of subsequent chapters. Nevertheless, none of the survey especially those are related to the bees-inspired algorithm studied and discussed the search neighbourhood. For this reason, the contents of the subsequent chapters primarily focused on this issue.

Chapter 3

ADAPTIVE ENLARGEMENT IN THE SEARCH NEIGHBOURHOOD IN THE BEES ALGORITHM

3.1. Preliminaries

The Bees Algorithm (BA) involves global and neighbourhood search. In the global search procedure, a number of bees are employed to explore at random the solution space. This kind of search is crucial as it enables the bees to escape from local optima. Meanwhile, neighbourhood search concentrates exploitation around promising solutions. The combination of global and neighbourhood search in population-based algorithms may locate solutions that gradually come closer to an optimal one. This iterative method makes this kind of algorithm more efficient than other conventional optimisation methods.

This work presents a modification of the neighbourhood search procedure in the BA. The proposed modification consists of manipulating the neighbourhood size by enlarging it dynamically as the optimisation progresses. The BA with the adaptive enlargement of the search neighbourhood, named BA-NE (which stands for the Bees Algorithm- neighbourhood enlargement), was tested on ten mathematical benchmarks with various characteristics and dimensionality.

The chapter is organised as follows: Section 3.2 explains the BA-NE, its experimental setup, and the results obtained, followed by a discussion of the

outcomes. The BA-NE was also tested on several engineering problems, with and without constraints and the results are presented in Section 3.3. Section 3.4 summarises the achievements.

3.2. BA-NE

This work proposes an adaptive neighbourhood enlargement procedure. Like the BA, this new algorithm required the same six parameters (n, m, e, nsp, nep, ngh). Initially, a number of bees (n) were sent randomly to the search space. Each bee was associated with one solution. The solutions representing the fitness of individual bees were then ranked in descending order. The top m solutions were regarded as selected sites. Of m sites, a number of top e site(s) were considered as elite one(s). Each of non-elite (m-e) and elite (e) sites respectively received nsp and nep forager bee(s) to exploit the discovered food source.

At present, when the neighbourhood search procedure fails to improve the current solution, the size of a flower patch is gradually shrunk until either a better solution is produced or the patch is abandoned (Ghanbarzadeh 2007; Pham et al., 2006a). In contrast to this practice, the idea proposed in this work was to enlarge the neighbourhood size if the neighbourhood search procedure progressed, and to keep it unchanged if the neighbourhood search brought no improvement. This proposed technique increased the range of the neighbourhood search around the promising solutions, aiming to speed up the optimisation of the fitness landscape. The size of the new neighbourhood was enlarged independently and depended on

how far the distance of the new best solution was from the current best solution of each particular patch. The farther the solution was found to be from the current best solution, the larger the new neighbourhood would be and vice versa.

For each dimension, the size of the new neighbourhood (ngh_j) was calculated according to Equation 3.1:

ngh_j=

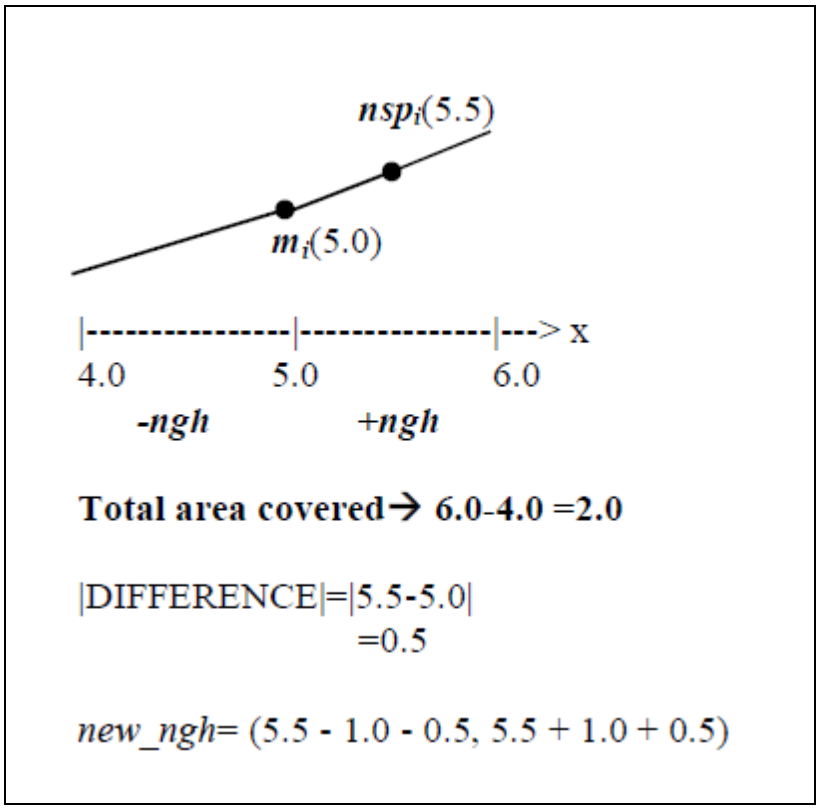
(x_j of nsp_i – ngh – DIFFERENCE, x_j of m_i + ngh + DIFFERENCE) (3.1)

where,

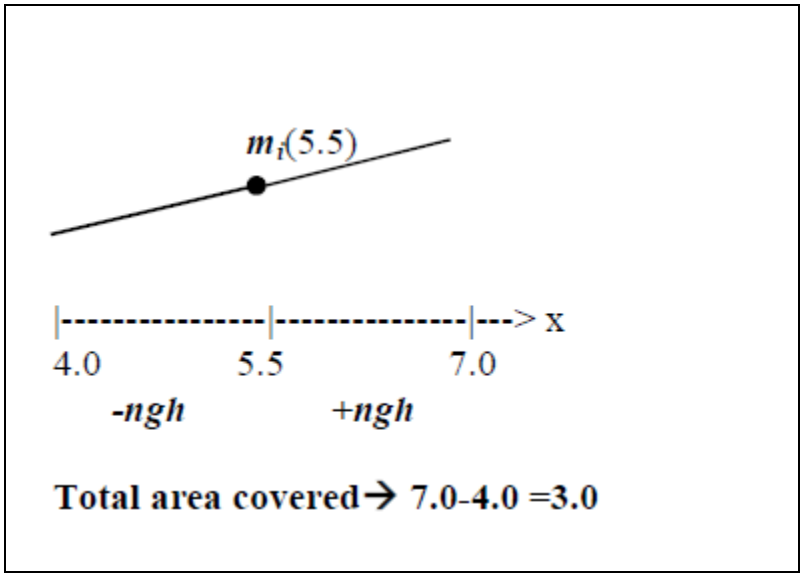
DIFFERENCE=| x_j of nsp_i- x_j of m_i|,

i=1, 2, 3, … m_{max} (nsp_{max}), where i was the index of selected and forager bee, j=1, 2, 3,…D, where j was the index of axis, D was the dimension of the problem and x was the point in that particular axis.

Figure 3.1 illustrates an example of applying the proposed method to the neighbourhood search procedure in a one-dimensional flower patch. The neighbourhood size was set as 1.0, a selected bee (m_i) was at position (5.0), a forager bee (nsp_i) landed at (5.5), and the optimisation problem was a maximisation one. Figure 3.1 (a) represents the neighbourhood search process around a given selected bee (before the enlargement procedure), while Figure 3.1



(a)



(b)

Figure 3.1 The neighbourhood size (a) before and (b) after the enlargement procedure

(b) shows the case when a forager found a better solution than the selected bee (after the enlargement procedure).

In this example, the total neighbourhood size was increased from 2.0 to 3.0. That is to say that the neighbourhood size was enlarged from 1.0 to 1.5 on the right and left sides of the current selected solution (m_i). In the event that no improvement took place, the neighbourhood size was kept equal to ngh. The advantage of using this kind of enlargement was that it allowed the bees to adjust independently and adaptively the neighbourhood size, with the aim of speeding up the search process.

As long as the stopping criteria were not satisfied, the neighbourhood search procedure was individually adjusted for each of the selected solutions. Figure 3.2 presents the flowchart of the neighbourhood search performed following this new approach, where i was the index of forager bees (nsp and nep) and i=1,2,3,..., nsp_{max} (nep_{max}). The stopping criteria in the flowchart were either that the solution obtained met the preset threshold or the index $i > nsp_{max}$ ($i > nep_{max}$).

After the neighbourhood search procedure had been completed, the BA-NE performed the global search as in the BA. The number of unselected bees (n-m) explored the search space randomly and hopefully better solutions would be found. Upon completion of the random search procedure, there were two groups of solutions; one group were solutions obtained by the neighbourhood search procedure, while the other group were the solutions that had been captured during

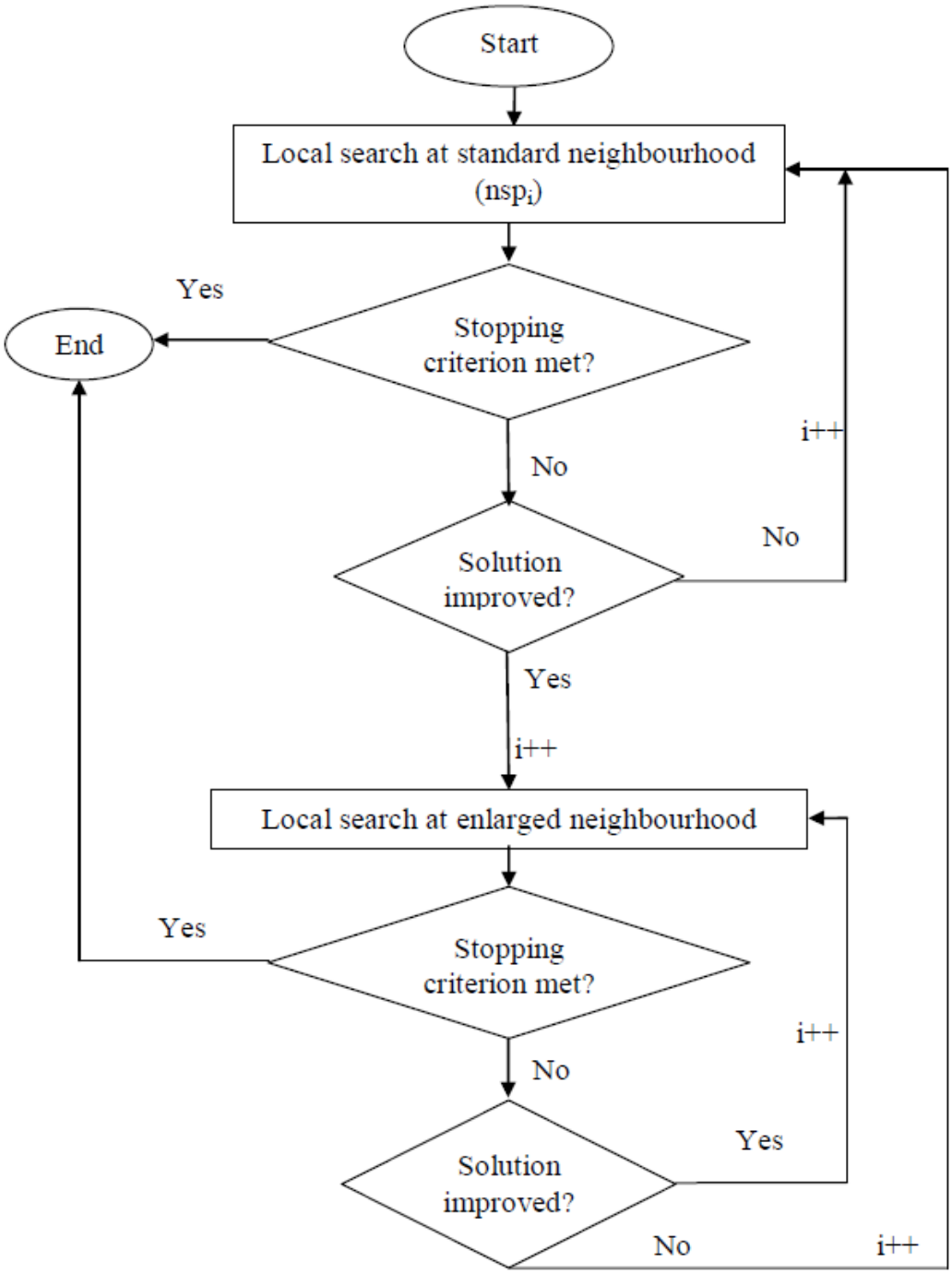


Figure 3.2 Flowchart of the neighbourhood search procedure in the BA-NE

the global search procedure. As long as the stopping criteria had not been met, the algorithm resumed the next iteration by sorting and ranking the solutions obtained from the previous iteration. Figure 3.3 shows the pseudocode of the BA-NE.

3.2.1. Experimental setup

Based on their characteristics and popularity, ten mathematical benchmarks were selected (see Table 3.1). The mathematical formulation of the test functions was referred to (Adorio 2005; Pham and Castellani 2009). These test functions spanned different dimensionalities and modalities that were able to challenge the performance of the algorithm. A function was called unimodal if the global optimum was the only optimum, whereas a multimodal function was the one that had two or more local optima. For multimodal functions, the search process should be able to avoid the local optima, which often cause premature convergence and stagnation.

The Goldstein & Price function represents an easy and multimodal function (Pham and Castellani 2009). Most algorithms easily locate the global optimum on this test function. The Schwefel, a multimodal function, possesses a great number of peaks and valleys. The function has the second best minimum far from the global minimum where many search algorithms were trapped (Digalakis and Margaritis 2002; Dimopoulos 2007). The Schaffer, another example of a multimodal function, was chosen because its surface could cause difficulty to the

1. Initialise population with random solutions.
2. Evaluate and rank of the fitness of the population.
3. While (stopping criterion not met).

   // Forming new population.
4. Select the m solutions.
5. Select the e solutions.
6. Recruit bees and evaluate their fitness.
7. Select the fittest bee from each patch.
8. Update new neighbourhood size.
9. Assign bees to search randomly and evaluate their fitness.
10. End While.

Figure 3.3 Pseudocode of the BA-NE

Table 3.1 Test functions

No.	Function	Interval	Equation	Minimum
1	Goldstein & Price (2D)	[-2,2]	$A(x_1,x_2)=1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)$ $B(x_1x_2)=30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)$ $f(x_1,x_2)=AB$	$\vec{x}=(0,-1)$ $f(\vec{x})=3$
2	Schwefel (2D)	[-500,500]	$f(x_1,x_2)=-x_1\sin(\sqrt{\lvert x_1\rvert})-x_2\sin(\sqrt{\lvert x_2\rvert})$	$\vec{x}=(420.97,420.97)$ $f(\vec{x})=-837.97$
3	Schaffer (2D)	[-100,100]	$f(x_1,x_2)=0.5+\frac{(\sin\sqrt{x_1^2+x_2^2})^2-0.5}{[1.0+0.001(x_1^2+x_2^2)]^2}$	$\vec{x}=(0,0)$ $f(\vec{x})=0$
4	Rosenbrock (10D)	[-1.2,1.2]	$f(\vec{x})=\sum_{i=1}^{10}100(x_{i+1}-x_i^2)^2+(1-x_i)^2$	$\vec{x}=(\vec{1})$ $f(\vec{x})=0$
5	Sphere (10D)	[-5.12,5.12]	$f(\vec{x})=\sum_{i=0}^{10}x_i^2$	$\vec{x}=(\vec{0})$ $f(\vec{x})=0$
6	Ackley (10D)	[-32,32]	$f(\vec{x})=20-20e^{-0.2\sqrt{(1/10)\sum_{i=1}^{10}x_i^2}}-e^{(1/10)\sum_{i=0}^{10}\cos(2\pi x_i)}+e$	$\vec{x}=(\vec{0})$ $f(\vec{x})=0$
7	Rastrigin (10D)	[-5.12,5.12]	$f(\vec{x})=\sum_{i=1}^{i=10}[(x_i)^2-10\cos(2\pi x_i)+10]$	$\vec{x}=(\vec{0})$ $f(\vec{x})=0$
8	Martin & Gaddy (2D)	[-20,20]	$f(x_1,x_2)=(x_1-x_2)^2+[\frac{(x_1+x_2-10)}{3}]^2$	$\vec{x}=(5,5)$ $f(\vec{x})=0$
9	Easom (2D)	[-100,100]	$f(x_1,x_2)=-\cos(x_1)\cos(x_2)e^{[(x_1-\pi)^2-(x_2-\pi)^2]}$	$\vec{x}=(\pi,\pi)$ $f(\vec{x})=-1$
10	Griewank (10D)	[-600,600]	$f(\vec{x})=\frac{1}{4000}\sum_{i=0}^{10}(x_i-100)^2-\prod_{i=0}^{i=10}\cos\left(\frac{x_i-100}{\sqrt{i+1}}\right)+1$	$\vec{x}=(\vec{100})$ $f(\vec{x})=0$

algorithms. Reaching the global optimum was not easy since it was surrounded by a high number of local minima (Zhao et al. 2009). Furthermore, the closer a candidate solution was to the global optimum, the longer the peak that must be overcome to move from one local minimum to the other (MacNish 2007).

The Rosenbrock function is frequently used to assess the performance of optimisation algorithms. The classical Rosenbrock, which is 2-dimensional, is regarded as a unimodal function. Over the years, researchers found that the Rosenbrock with n-dimensional (n=4~30) was a multimodal function (Shang and Qiu 2006). In this work, the 10-dimensional Rosenbrock was used. Besides its multimodality, a nonlinear deep valley with the shape of a parabola that led to the global minimum and nonlinear interactions among the variables was the challenge that was offered by the Rosenbrock (Akay and Karaboga 2010).

The Sphere represented a convex, symmetric and unimodal test function. On the Ackley test function, the exponential term produces numerous local minima and valleys that spread over its problem domain. In addition, its optima are regularly distributed (Akay and Karaboga 2010). The Rastrigin, a multimodal function, was constructed from a sphere but having a modulator term. The challenge offered by this function was that an algorithm could be easily trapped in its million local optima on its way towards the global optimum (Karaboga and Basturk 2008). In addition, its contour was made up of numerous local minima that were evenly spaced and their value raised with the distance to the global minimum (Karaboga and Basturk 2007a). The Martin & Gaddy was an example of simple unimodal

function (Pham and Castellani 2009). The Easom, a unimodal and nonlinear function, represented a problem of flat surfaces (Pham and Castellani 2009). Flat surfaces were a hindrance for optimisation algorithms. This was because there was no variable step size that could give information on which direction would lead to a better solution (Digalakis and Margaritis 2002). Also, the global minimum on the Easom was located in a very narrow hole. The Griewank, a multimodal function, has a product term that introduces interdependence among the variables (Akay and Karaboga 2010). A parabola was produced by the terms of summation, where the local optima were located above the parabola level (Digalakis and Margaritis 2002).

Parameter selection highly influenced the performance of the algorithms in terms of solution quality and execution time (Chai-ead et al. 2011). In this work, the solution quality was represented by the accuracy of the solution, while the execution time was denoted by the number of evaluations. To achieve the best quality and execution time, a large number of experiments were conducted on the BA in order to determine the best parameter setting for this algorithm. For each test function, parameters values were optimised by trial and error since there was no defined procedure to guide the user in selecting the most suitable set of parameters (Dereli and Das 2010). The parameter setting that served best in the BA was then employed in the BA-NE. The BA-NE with corresponding parameter setting was tested on the benchmarks. For every benchmark, the BA and BA-NE were run for 100 times, so a meaningful statistical analysis could be performed.

Table 3.2 presents the parameter setting that was employed in the BA and BA-NE.

The optimisation was terminated when the number of iterations reached 5000, or an acceptable solution was found. A solution was an acceptable one if the difference between the solution found and the global optimum was less than or equal to 0.001. This threshold is illustrated in Equation 3.2:

$$f_{min} - f_{opt} \leq 0.001 \quad (3.2)$$

where f_{min} was the best solution found and f_{opt} was the known global optimum of the problem considered (Ali and Kaelo 2008).

3.2.2. Experimental results

The average of accuracy and number of evaluations obtained by the BA, BA-NE and other state-of-art algorithms are presented in Table 3.3 (a) and (b), respectively. The results of the PSO, Evolutionary Algorithm (EA) and ABC were extracted from (Pham and Castellani 2009). When the difference between the final solution and global optimum was less than 0.001, the accuracy was denoted as 0.0000. It should be noted that the parameters of the PSO, EA and ABC were tailored to be equivalent to 100 evaluations in one cycle (i.e., iteration). Conversely, parameters that were adopted in this work were not subjected to 100 evaluations per iteration. For that reason, the number of cycles that were denoted by the 'speed' in (Pham and Castellani 2009) was converted to the number of

Table 3.2 Parameter setting in the BA and BA-NE

No.	Function	n	m	nsp	e	nep	ngh
1	Goldstein & Price (2D)	10	3	2	1	13	0.005
2	Schwefel (2D)	10	2	5	1	6	0.5
3	Schaffer (2D)	100	4	10	2	30	3
4	Rosenbrock (10D)	15	8	10	5	30	0.0015
5	Sphere (10D)	10	7	20	1	30	0.05
6	Ackley (10D)	100	8	10	1	20	0.7
7	Rastrigin (10D)	100	3	20	1	40	0.01
8	Martin & Gaddy (2D)	10	5	10	1	30	0.1
9	Easom (2D)	100	4	10	2	30	0.5
10	Griewank (10D)	100	40	10	20	30	1.5

Table 3.3 Comparison on (a) accuracy and (b) average number of evaluations against other algorithms

(a)

No.	Functions	PSO		EA		ABC		BA		BA-NE	
		Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.
1	Goldstein & Price (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0003	0.0000	0.0003
2	Schwefel (2D)	4.7376	23.4448	4.7379	23.4448	0.0000	0.0000	0.0000	0.0005	0.0000	0.0005
3	Schaffer (2D)	0.0000	0.0000	0.0009	0.0025	0.0000	0.0000	0.0000	0.0003	0.0000	0.0003
4	Rosenbrock (10D)	0.5998	1.0436	61.5213	132.6307	0.0965	0.0880	44.3210	112.29	0.0508	0.0337
5	Sphere (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0002
6	Ackley (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.2345	0.3135	1.2297	0.2383
7	Rastrigin (10D)	0.1990	0.4924	2.9616	1.4881	0.0000	0.0000	24.8499	8.3306	23.3201	9.1703
8	Martin & Gaddy (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0003
9	Easom (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	2.0096	0.0000	0.0003	0.0000	0.0003
10	Griewank (10D)	0.0008	0.0026	0.0210	0.0130	0.0052	0.0078	0.3158	0.1786	0.1912	0.1024

(b)

No.	Functions	PSO		EA		ABC		BA		BA-NE	
		Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.
1	Goldstein & Price (2D)	3,262	822	2,002	390	2,082	435	504	211	384	168
2	Schwefel (2D)	84,572	90,373	298,058	149,638	4,750	1,197	1,140	680	1,140	701
3	Schaffer (2D)	28,072	21,717	219,376	183,373	21,156	13,714	121,088	174,779	132,176	157,520
4	Rosenbrock (10D)	492,912	29,381	500,000	0	497,728	16,065	935,000	0	935,000	0
5	Sphere (10D)	171,754	7,732	36,376	2,736	13,114	480	285,039	277,778	325,125	252,987
6	Ackley (10D)	236,562	9,119	50,344	3,949	18,664	627	910,000	0	910,000	0
7	Rastrigin (10D)	412,440	67,814	500,000	0	207,486	57,568	885,000	0	885,000	0
8	Martin & Gaddy (2D)	1,778	612	1,512	385	1,498	329	600	259	450	187
9	Easom (2D)	16,124	15,942	36,440	28,121	1,542	201	5,280	6,303	4,576	3,344
10	Griewank (10D)	290,466	74,501	490,792	65,110	357,438	149,129	4,300,000	0	4,300,000	0

evaluations, by multiplying the ‘speed’ with 100.

In terms of number of evaluations, the BA and BA-NE outperformed the PSO, EA and ABC in the case of the Goldstein & Price, Schwefel and Martin & Gaddy. In other cases, the performance of the BA and BA-NE were comparable to other algorithms.

In addition, a test to determine whether the performance of the BA and BA-NE were statistically different was conducted. Before performing this test, the normality test was performed on every data set. If both data sets which were generated by the BA and BA-NE were normally distributed, the Student’s t-test at 95% level of confidence was performed to identify whether the results were statistically different. If the t-value generated by this test was more than 1.98, the performance of the BA and BA-NE in that particular problem was significantly different and vice versa. On the other hand, if either or both of the data sets were not normally distributed, the Mann Whitney test was applied. When the significant value generated by the Mann Whitney test was less than 0.05, the performance of two algorithms was significantly different and vice versa. In this thesis, a significant difference was denoted by ‘S’, while a non-significant difference was represented by ‘NS’. Furthermore, the superiority of the algorithm was indicated by a hierarchical method, where the comparison on accuracy was performed prior to that of the number of evaluations (Pham and Castellani 2009). It should be noted that this statistical and hierarchical procedure was applied throughout the subsequent two chapters.

The significance of the difference between the BA and BA-NE is presented in Table 3.4. The comparisons on the number of evaluations on the Rosenbrock, Ackley, Rastrigin and Griewank were omitted since they were consistent in all runs. According to the table, the BA-NE outperformed the BA in the case of the Goldstein & Price, Rosenbrock, Ackley, Martin & Gaddy and Griewank. On the other hand, the BA did not outperform the BA-NE in any case.

3.2.3. Discussions

Compared to the BA, the BA-NE produced a better performance on the Rosenbrock, Ackley and Griewank, which were all multimodal and 10-dimensional problems. It was because the enlarged neighbourhood promoted the BA-NE to converge faster at an early stage of iteration. The fast convergence at an early stage of optimisation was visualized by the steepness of the graph. The steeper the graph, the faster the optimisation progressed. For example, Figure 3.4 shows the optimisation progress of the BA-NE and BA on the Rosenbrock.

However, the BA-NE failed to reach the global optimum on the Rosenbrock, Ackley, Rastrigin and Griewank. Despite its multimodality, the global optimum that was located in a deep valley might cause the difficulty on the Rosenbrock. Meanwhile, the highly multi-pocketed landscape that was created by a cosinusoidal noise component (Pham and Castellani 2009) might have contributed to the poor optimisation progress on the Ackley, Rastrigin and Griewank. It could be said that the enlarged neighbourhood could not accommodate the bees to locate the global optimum on the problems that exhibited such characteristics. In other

Table 3.4 Significance difference between the BA and BA-NE

No.	Function	Accuracy	Number of Evaluations
1	Goldstein & Price (2D)	NS	S
2	Schwefel (2D)	NS	NS
3	Schaffer (2D)	NS	NS
4	Rosenbrock (10D)	S	-
5	Sphere (10D)	NS	NS
6	Ackley (10D)	S	-
7	Rastrigin (10D)	NS	-
8	Martin & Gaddy (2D)	NS	S
9	Easom (2D)	NS	NS
10	Griewank (10D)	S	-

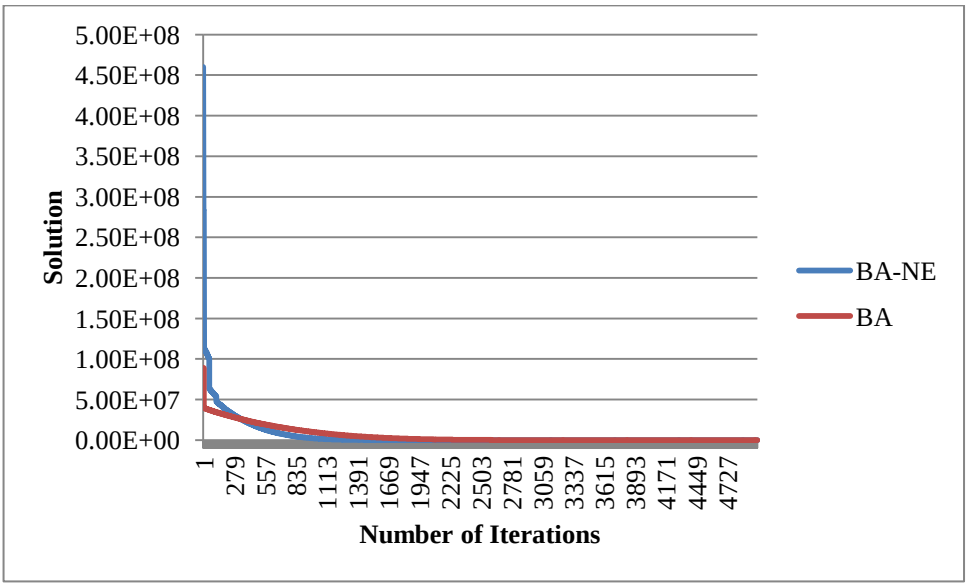


Figure 3.4 Optimisation progress on the Rosenbrock

words, the enlarged neighbourhood reduced the capability of the algorithm to intensify the search, where this kind of search was crucial to solve problems with multi-pocketed surfaces.

On easy functions such as the Goldstein & Price and Martin & Gaddy, the solution improvement might occur in every iteration. This impressive progress nevertheless might risk an ‘explosion’. It meant that the neighbourhood size became larger and larger as the number of iterations increased. Due to this uncontrolled neighbourhood size, the role of neighbourhood search might conflict with that of global search, where the size of the search area equalled the total search space. Since they were easy functions, the global optimum could still be located with a large neighbourhood search.

The performance of the BA and BA-NE was similar on the Schwefel, Schaffer, Sphere, Rastrigin, and Easom. Therefore, it could be said that the modified algorithm had no superiority over the BA when solving these problems.

The Schwefel and Sphere were also examples of easy functions. The BA and BA-NE could easily converge and locate the global optimum quickly on these functions. Figure 3.5 shows the similar steepness produced by the BA and the BA-NE during the early stage of optimisation on the Sphere. The similar steepness indicated that both of the algorithms converged equally at the same speed when solving this problem.

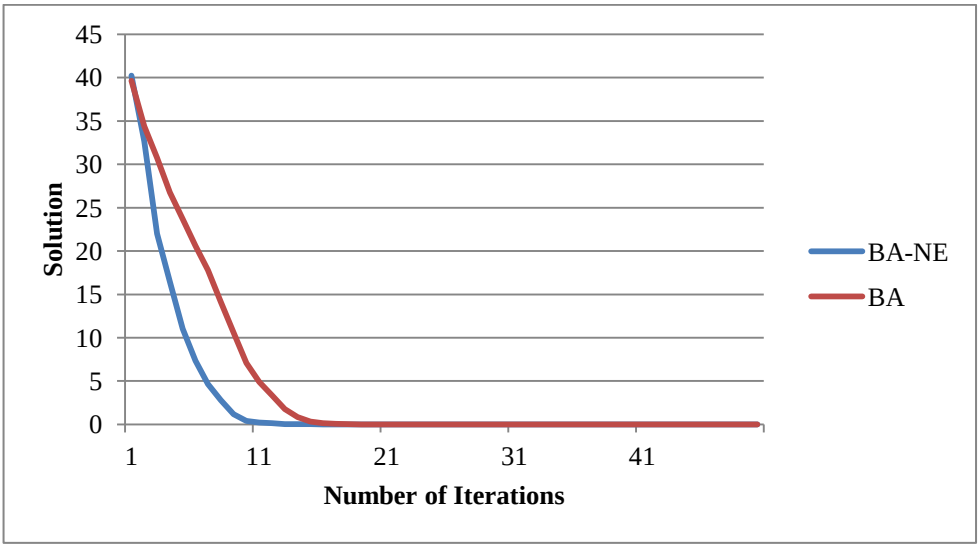


Figure 3.5 Optimisation progress on the Sphere

The vast number of extreme peaks on the Schaffer and Rastrigin meant that the BA-NE hardly made an improvement. Therefore, the adaptive enlargement of the search neighbourhood rarely happened. On the other hand, the BA-NE often performed the neighbourhood search by using the fixed neighbourhood size. However, it was expected that the enlarged search neighbourhood alone was also not able to help the algorithm to perform better than the BA on these functions as they were featuring multi-peak landscape. This landscape required an intensive neighbourhood search which was served less in the BA-NE.

Meanwhile, on the Easom, the flat surface caused the similar performance of the algorithms. As long as the bees landed on the flat surface, the procedure of adaptive enlargement of the search neighbourhood would not take place. Consequently, the BA-NE produced a similar performance to that of the BA. This is shown by the similar straight lines at the early stage of optimisation (see Figure 3.6).

3.3. Applications

3.3.1. Single objective problem without constraints

The effectiveness of the adaptive enlargement of the search neighbourhood was verified by testing the proposed algorithm on the gear train problem. The objective of this problem was to design a compound gear train such that the gear ratio between the driver and driven shafts was as close as possible to 1/6.931 (or 0.1442793) (Kannan and Kramer 1994). The closer the gear ratio to this value, the

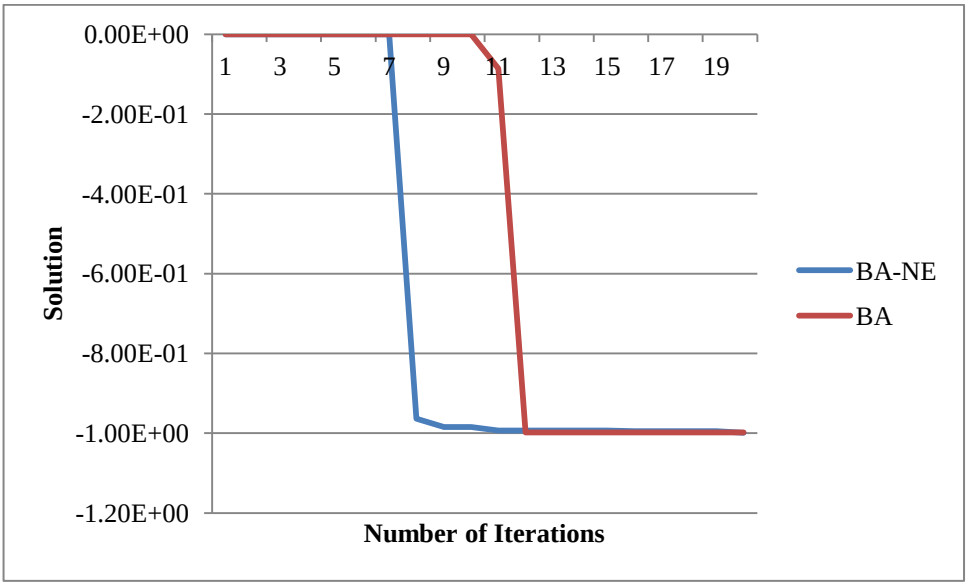


Figure 3.6 Optimisation progress on the Easom

smaller the % error would be. The gear ratio and % error were calculated by using Equations 3.3 and 3.4, respectively:

$$\text{Gear ratio}=\frac{x_2 x_3}{x_1 x_4} \tag{3.3}$$

$$\%\ \text{error} = \frac{Gear\ ratio-(1/6.931)}{(1/6.931)} \times 100\% \tag{3.4}$$

The number of teeth which was the design variable of the problem was to be an integer between 12 and 60. The diagram and mathematical formulation of this problem is provided in Appendix A.

By trial and error, a set of parameters that worked best on the BA was captured (see Table 3.5). With this parameter setting, 30 independent experiments were run on the BA. The optimisation was interrupted when the number of evaluations was more than 60 or the solution obtained was equal to or smaller than 2.700857E-12. This value was the best reached by the ABC in (Akay and Karaboga 2010). With the same parameter setting and stopping criteria, the experiments were conducted on the BA-NE. The experimental results of the BA and BA-NE were then compared against Unified Particle Swarm Optimisation (UPSOm) (Parsopoulos and Vrahatis 2005) and ABC (Akay and Karaboga 2010). According to Table 3.6, the average solution and standard deviation obtained by the UPSOm and ABC were better than the ones obtained by the BA and BA-NE. The % error produced by the ABC was also smaller than that of the BA and BA-NE.

Table 3.5 Parameters for gear train problem

Parameter	Value
Number of scout bees, n	5
Number of selected bees, m	2
Number of forager bees for each selected bee, nsp	1
Number of elite bees, e	1
Number of forager bees for each elite bee, nep	2
Neighbourhood size, ngh	2.0

Table 3.6 Comparison against other algorithms on the gear train problem

	UPSOm	ABC	BA	BA-NE
Avg. Solution	3.80562E-08	3.641339E-10	6.84E-05	2.12E-06
Std. Deviation	1.09631E-07	5.525811E-10	0.366946E-04	4.43477E-06
Best Solution	2.70085E-12	2.700857E-12	9.92158E-10	1.54505E-10
x_1	NA	49	47	43
x_2	NA	16	12	13
x_3	NA	19	26	21
x_4	NA	43	46	44
Gear ratio	NA	0.144281	0.144311	0.144292
% error	NA	0.001%	0.022%	0.009%

Using the Mann Whitney Test, the average solution produced by the BA-NE was better than the one obtained by the BA. In addition, the % error produced by the BA-NE was less than half of the % error obtained by the BA. This experimental result suggested that the adaptive enlargement in the search neighbourhood encouraged the optimisation progress in the BA-NE.

3.3.2. Single objective problem with constraints

A set of four well-known engineering design problems were chosen. An engineering design problem is normally large and comprised nonlinear objective problem(s) and constraints that must not be violated (Akay and Karaboga 2010). In this work, it should be noted that the BA and BA-NE were not tailored to adapt to constrained problems. On the other hand, they adopted the idea of (He et al. 2004), where the solution searching was only conducted inside the feasible region. In this method, a produced solution was checked as to whether it satisfied all constraints. If it did, it would be regarded as a feasible solution. A feasible solution was put into a solution list, while an infeasible one was discarded. The solution searching was resumed until a required number of feasible solutions were captured.

The chosen engineering design problems were the welded beam, pressure vessel, tension/ compression spring and speed reducer (Akay and Karaboga 2010). Minimising the cost was the objective of the welded beam and pressure vessel design problem. Meanwhile, the objective of the tension/ compression spring and

speed reducer problem was to minimise the design weight. These design problems had various dimensionalities, which was indicated by the number of design variables, x_i. Meanwhile, the details of these problems, including mathematical formulation and constraints, are described in Appendix B (Akay and Karaboga 2010).

A number of experiments were carried out on the BA with different parameter settings. The parameter setting that produced the best performance on the BA was chosen and used for the BA-NE. Table 3.7 shows the parameter setting that was employed to solve these engineering design problems. The BA and BA-NE were run 30 times on every problem, with 30,000 evaluations.

In addition to the UPSOm and ABC, the performance of the BA and BA-NE were also compared against results that were obtained by the Society and Civilisation Algorithm (SCA) (Ray and Liew 2003), PSO (He et al., 2004) and (μ+λ)-ES, which was a version of Evolutionary Strategies (Mezura-Montes and Coello Coello 2005) (see Table 3.8). In general, the average of the solutions produced by the BA and BA-NE was comparable to other algorithms.

The significance of the difference between the BA and BA-NE is shown in Table 3.9. It implies that the BA-NE gave a better performance than the BA on the welded beam and pressure vessel problem, while both of the algorithms performed equally on the tension/ compression spring and speed reducer problem. These experimental results proved that the adaptive enlargement of the search

Table 3.7 Parameter setting for design problems

No.	Problem	n	m	nsp	e	nep	ngh
1	Welded Beam (4D)	10	5	2	2	4	0.08
2	Pressure Vessel (4D)	10	5	2	3	6	0.2
3	Tension/ comp. spring (3D)	6	5	5	1	8	0.001
4	Speed Reducer (7D)	35	15	5	5	15	0.01

Table 3.8 Comparison against other algorithms on design problems

Problem	Stats.	SCA	PSO	(μ + λ)-ES	UPSOm	ABC	BA	BA-NE
Welded Beam	Best	NA	NA	1.724852	1.92199	1.724852	1.734783	1.731916
	Mean	NA	NA	1.777692	2.83721	1.741913	1.768855	1.749546
	Std. Dev.	NA	NA	0.088	0.68	0.031	0.040	0.013
	Evaluations	NA	NA	30000	100000	30000	30000	30000
Pressure Vessel	Best	6171.00	6059.7143	6059.701610	6544.27	6059.714736	6289.745562	6283.130775
	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.308144	6853.349849	6749.722776
	Std. Dev.	NA	310	210	996	205	609	542
	Evaluations	20000	30000	30000	100000	30000	30000	30000
Tension/ Compression spring	Best	0.012669	0.012665	0.012689	0.0131200	0.012665	0.00988	0.00988
	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	0.01036	0.01027
	Std. Dev.	0.00059	0.000041	0.00039	0.0072	0.013	0.00048	0.00048
	Evaluations	25167	15000	30000	100000	30000	30000	30000
Speed Reducer	Best	2994.744241	NA	2996.348094	NA	2997.058412	2997.843904	2998.348453
	Mean	3001.758264	NA	2996.348094	NA	2997.058412	3005.295876	3003.358497
	Std. Dev.	4.0	NA	0	NA	0	3.2	3.1
	Evaluations	54456	NA	30000	NA	30000	30000	30000

Table 3.9 Significance of the difference of the BA and BA-NE on the engineering design problems

Problems	Significance
Welded Beam Problem	S
Pressure Vessel	S
Tension/ Compression Spring	NS
Speed Reducer	NS

neighbourhood had the potential to encourage the bees to reach the solution faster than using the fixed and small search neighbourhood. Furthermore, the BA did not outperform the BA-NE in any case. The x_i and g_i values of each problem obtained by the BA and BA-NE are provided in Table 3.10 (a)-(d).

3.4. Summary

Adaptive enlargement of the search neighbourhood was proposed. This method was intended to speed up the optimisation. The landscape of the surface highly influenced the performance of the algorithm. A smooth surface encouraged the algorithm to capture the global optimum in a short time, while a rough and noisy surface caused more time to be needed by the algorithm to find the global optimum.

A number of experiments were carried out on the BA, where the parameter values were obtained by trial and error. The parameter settings that worked best in the BA were then used for the BA-NE. The experimental results produced by the BA-NE were better than the ones obtained by the BA in five of ten mathematical benchmarks and in three of five single objective design problems. On the other hand, the BA never outperformed the BA-NE in any case. This result proved that the proposed algorithm was robust since its good performance did not depend on its own parameter-tuning.

Table 3.10 The x_i and g_i values of the best solution obtained by the BA and BA-NE on (a) welded beam, (b) pressure vessel, (c) tension/ compression spring and (d) speed reducer problem

(a)

Variables and Constraints	BA	BA-NE
x_1	0.206526	0.203612
x_2	3.48281	3.52698
x_3	9.004	9.02774
x_4	0.207398	0.206293
g_1	-52.4761	-21.0005
g_2	-25.2893	-23.0144
g_3	-0.00087	-0.00268
g_4	-3.42486	-3.42527
g_5	-0.08153	-0.07861
g_6	-0.2355	-0.23554
g_7	-132.54	-45.5134

(b)

Variables and Constraints	BA	BA-NE
x_1	0.808553	0.803735
x_2	0.399348	0.402705
x_3	41.6614	41.4947
x_4	196.083	198.152
g_1	-0.00449	-0.00289
g_2	-0.0019	-0.00685
g_3	-2.09E+08	-2.11E+08
g_4	-43.917	-41.848

(c)

Variables and Constraints	BA	BA-NE
x_1	0.050041	0.050011
x_2	0.375376	0.374656
x_3	8.51061	8.53865
g_1	-7.6E-05	-2E-05
g_2	-4.4E-06	-5.2E-05
g_3	-4.86073	-4.86041
g_4	-0.71639	-0.71689

(d)

Variables and Constraints	BA	BA-NE
x_1	3.50086	3.50193
x_2	0.700063	0.700005
x_3	17	17
x_4	7.31302	7.31539
x_5	7.80287	7.80578
x_6	3.3526	3.35355
x_7	5.28682	5.28685
g_1	-0.07431	-0.07443889
g_2	-0.19834	-0.19845198
g_3	-0.49796	-0.498
g_4	-0.90138	-0.90127
g_5	-0.00211	-0.00296
g_6	-7.7E-05	-9.4E-05
g_7	-0.70247	-0.7024979
g_8	-0.00016	-0.00054
g_9	-0.58327	-0.58311
g_{10}	-0.05253	-0.05264
g_{11}	-0.01119691	-0.0115613

Chapter 4

ASYMMETRICAL SEARCH NEIGHBOURHOOD IN THE BEES ALGORITHM

4.1. Preliminaries

According to the experimental results from Section 3.2.2, the BA-NE did not reach the global optimum after 5000 iterations on the Rosenbrock, Ackley, Rastrigin and Griewank. Despite its multimodality, the global optimum that was located in a deep valley might have caused the difficulty on the Rosenbrock. Meanwhile, the highly multi-pocketed landscape that was created by a cosinusoidal noise component (Pham and Castellani 2009) might have contributed to the poor optimisation progress on the Ackley, Rastrigin and Griewank.

This work aims to investigate the effect of using an asymmetrical search neighbourhood and to study its rationale in the context of the BA, as opposed to the standard practice of using a symmetrical one. An algorithm with an asymmetrical search neighbourhood, named BA-AN (which stands for the Bees Algorithm- asymmetrical neighbourhood), was tested on the same ten test functions and the gear train problem. The experimental results were compared to those obtained by the BA and BA-NE.

This chapter is organised as follows: Section 4.2 describes the symmetrical search neighbourhood, which was normally used in the BA and section 4.3 is about TRIZ and how it leads to the use of an asymmetrical search neighbourhood. The BA-NE

with an asymmetrical search neighbourhood is described at Section 4.4. Section 4.5 states the application of the proposed algorithm on an engineering problem. Lastly, Section 4.6 summarises the work.

4.2. Symmetrical search neighbourhood

Symmetry is defined as how half of an object can be a reflection of another half and the reflection can be upon axes or planes. The object is symmetrical if it can produce an exact reflection about any plane (Ball et al., 2011). Complying with this definition, the search neighbourhood that was used for the BA and BA-NE was symmetrical. Since the neighbourhood size was defined and implemented in a 1-dimensional space, the reflection of the neighbourhood was made about a point, instead of an axis or plane (see Figure 4.1). The point was the position of a scout bee. During the neighbourhood search procedure, the forager bees searched randomly within this symmetrical search neighbourhood.

A symmetrical search neighbourhood had been used for the BA ever since its introduction. Nevertheless, the rationale for making the search neighbourhood symmetrical was not well-justified. In addition, the failure to find the global optimum on the Ackley, Rastrigin and Griewank demanded a special approach by the BA-NE. As stated earlier, these three functions shared the same characteristic, which was featuring multi-pocketed surfaces created by a cosinusoidal noise component (Pham and Castellani 2009). This noise component might cause difficulty to the algorithm in finding the global optimum. For this reason, this noise component should be reduced or eliminated from the problems.

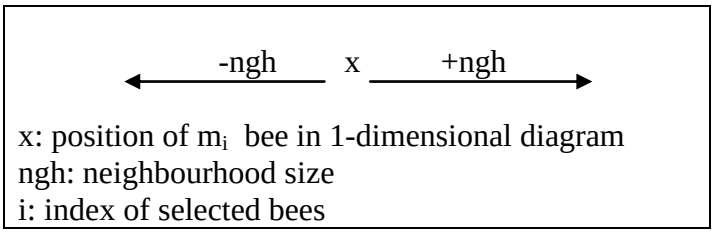


Figure 4.1 Symmetrical search neighbourhood

However, it was not possible to remove the noise feature from the problems as these benchmarks were deliberately built to exhibit this characteristic. Alternatively, the noise component could be treated partially, more precisely by reducing the neighbourhood size so that the noisy surface would no longer appear as it was. On the other hand, it then became a smooth surface. However, reducing the neighbourhood size would somehow slow down the optimisation progress. In other words, it may slow down the speed of the algorithm. This conflict yielded to a number of solutions, as proposed by TRIZ.

4.3. Solutions from TRIZ perspective

TRIZ was developed by Genrich Altshuller in 1946. In (Altshuller 2001), he stated that changing one part of the system might cause drawback(s) in the system's other parts. This technical contradiction could be solved by applying the inventive solution. An inventive solution always has two conditions that must be met:

1. Improving a single part or characteristic of the system without,
2. Worsening other parts or characteristics of the system or adjacent systems.

The classical TRIZ contradiction matrix comprised 39 improving and worsening features and proposed 40 inventive principles. However, none of the features was explicitly concerned with the problems that were associated with noise. Darrell Mann was the individual who was responsible for the evolution of TRIZ, when he added nine new features into the matrix (Mann et al., 2003), of which one was noise.

According to (Mann et al., 2003; Mann 2009), noise is not only produced by an object, system or surrounding but also could be received by those entities. In this work, the noise problem is received by the algorithm. Noise also sometimes could be either useful or harmful. In respect to the noise problem in the Ackley, Rastrigin and Griewank benchmarks, a noisy surface could be regarded as a harmful one since it traps the bees and thus might cause stagnation of the solution. Due to this fact, reducing the noise surface by means of reducing the neighbourhood size in the BA-NE was regarded as improving the system.

As stated earlier, a small neighbourhood size might reduce the amount of noisy surface but at the same time it could slow down the optimisation progress. A fast optimisation progress was one of the characteristics of a good optimisation algorithm. Hence, slowing down the optimisation progress is regarded as worsening the system.

The system, improving and worsening features, as well as proposed solutions (Mann et al., 2003; Mann 2009) are listed in Table 4.1. To solve the stated conflict, there were seven possible inventive solutions suggested. Each solution could be implemented differently, based on an individual's interpretation. Because of this, the solutions suggested by the TRIZ matrix were not subjected to a rigid and single definition. On the contrary, they were abstract, ambiguous and subjective (Ang et al., 2010).

Table 4.1 Problem analysis in the BA by the TRIZ approach

System	Improving Feature	Worsening Feature	Proposed Solutions
Neighbourhood size in the Bees Algorithm	Noise	Speed	1. Segmentation 3. Local Quality 4. Asymmetry 14. Curvature 24. Intermediary 31. Porous Materials 39. Inert Atmosphere

In this work, an asymmetrical solution was chosen. There were a number of ways to implement the asymmetry solution and according to (Mann et al., 2003; Mann 2009), this solution could be implemented by:

A – Where an object or system is symmetrical or contains lines of symmetry, introduce asymmetries.

B - Change the shape of an object or system to suit external asymmetries (e.g., ergonomic features).

C - If an object or system is already asymmetrical, increase the degree of asymmetry.

Of these three suggestions, instance A was selected because the current neighbourhood in the BA-NE was symmetrical. An asymmetrical search neighbourhood was introduced and coincidently its effect(s) on the BA had never been studied. To produce an asymmetrical search neighbourhood, the sum of old neighbourhood size and enlargement were kept on one side of the current best solution, while only the old neighbourhood size was on the other side (Note that the old neighbourhood size was referred to the neighbourhood size presented in Table 3.2). This setting formed an asymmetrical search neighbourhood. An asymmetrical search neighbourhood was only used when there was an improvement in the neighbourhood search. In the case that no improvement had been made, the neighbourhood size was symmetrical and fixed, as applied in the BA.

4.4. BA-AN

Similarly to the BA, BA-AN started by sending the bees randomly around the search space. Each bee which was associated with a solution that had been obtained from initialisation was ranked in descending order. The top m solution(s) were selected for neighbourhood search. Of m solution(s), the top e solution(s) which were regarded as elite sites received nep forager bee(s) to exploit the e discovered site(s). Meanwhile, the remaining selected sites (m-e), received nsp forager bee(s) for neighbourhood search.

During neighbourhood search, solution evaluation in the BA-AN was performed in a serial way. Serial evaluation meant that any solution obtained by a forager bee (nsp_i) was compared straight away to the current best solution. The solution of nsp_i was kept if it was better than the current best solution, otherwise it would be discarded. Then, the neighbourhood search was carried out by the nsp_{i+1} and the comparison against the current best solution was again performed. The step was repeated until nsp solutions were obtained and compared to the current best solution (see Figure 4.2).

In conjunction with serial evaluation, four different assessments were carried out. For each current best solution in an assessment, there were two sides; former and latter. The former side was regarded as the first attempt, while the latter (written in the parenthesis) was the second attempt (see Table 4.2). In the assessment-i and assessment-ii, if there was improvement in the first attempt, the neighbourhood

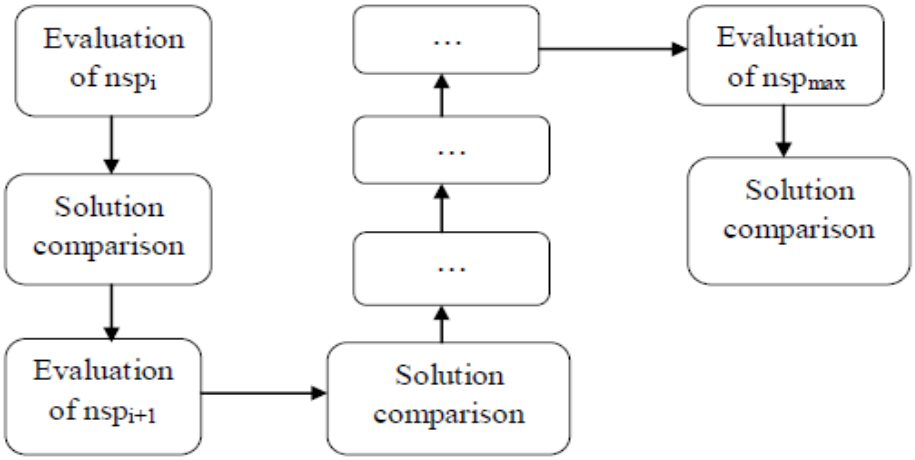
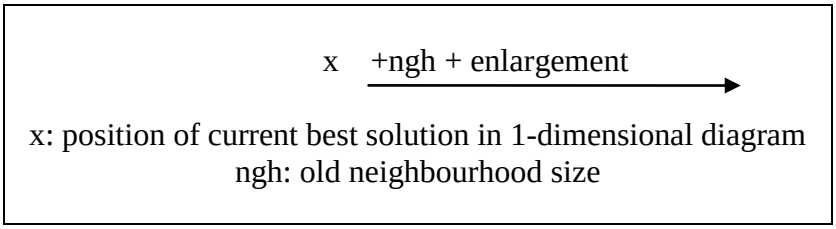


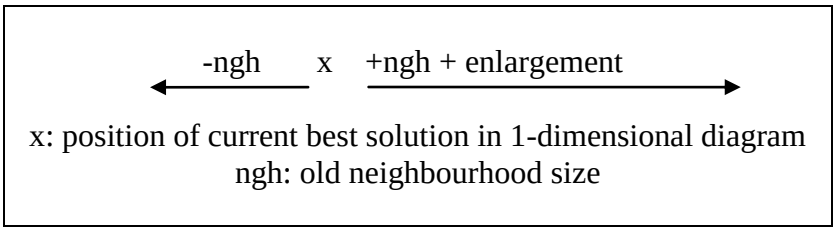
Figure 4.2 Serial evaluation

Table 4.2 Assessments and its descriptions

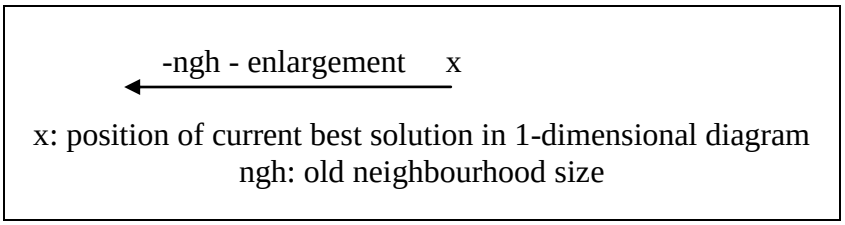
Assessment	Description
i	Right first (then Left) with position of current best solution as lower (upper) bound. (Figure 4.3 (a))
ii	Right first (then Left) with standard neighbourhood as lower (upper) bound. (Figure 4.3 (b))
iii	Left first (then Right) with position of current best solution as upper (lower) bound. (Figure 4.3 (c))
iv	Left first (then Right) with standard neighbourhood as upper (lower) bound. (Figure 4.3 (d))



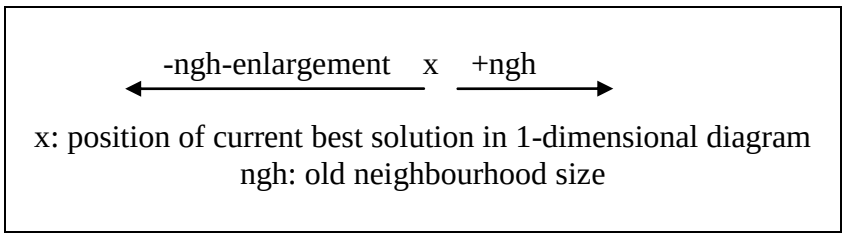
(a)



(b)



(c)



(d)

Figure 4.3 Visualisation of (a) Assessment-i, (b) Assessment-ii, (c) Assessment-iii and (d) Assessment-iv

search would be continued on the right side of the current best solution. In the case that no improvement had been made, the neighbourhood search would be performed on the left side of the current best solution (second attempt).

Meanwhile, in assessment-iii and assessment-iv, if there was an improvement in the first attempt, the neighbourhood search would be continued on the left side of the current best solution. In the case that no improvement had been made, the neighbourhood search would be performed on the right side of the current best solution (second attempt). There was nothing special in each assessment other than to observe whether the evaluation on the right side prior to the left side was superior to doing the left side prior to the right side. Also, it was to observe whether making the position of the current best solution or the old neighbourhood as the boundary produced a different performance by the BA-AN.

If the second attempt of any assessment did not yield a better solution, the neighbourhood search was performed in the symmetrical search neighbourhood as employed in the BA. The flowchart in Figure 4.4 depicts the process of the neighbourhood search in the BA-AN, where i was the index of forager bees (nsp and nep), and i=1,2,3,..., nsp_{max} (nep_{max}). The stopping criteria in the flowchart were when either the solution obtained met the preset threshold or the index i > nsp_{max} (i > nep_{max}).

After neighbourhood search had been completed, the BA-AN resumed the optimisation by performing a global search as in the BA-NE. The process was

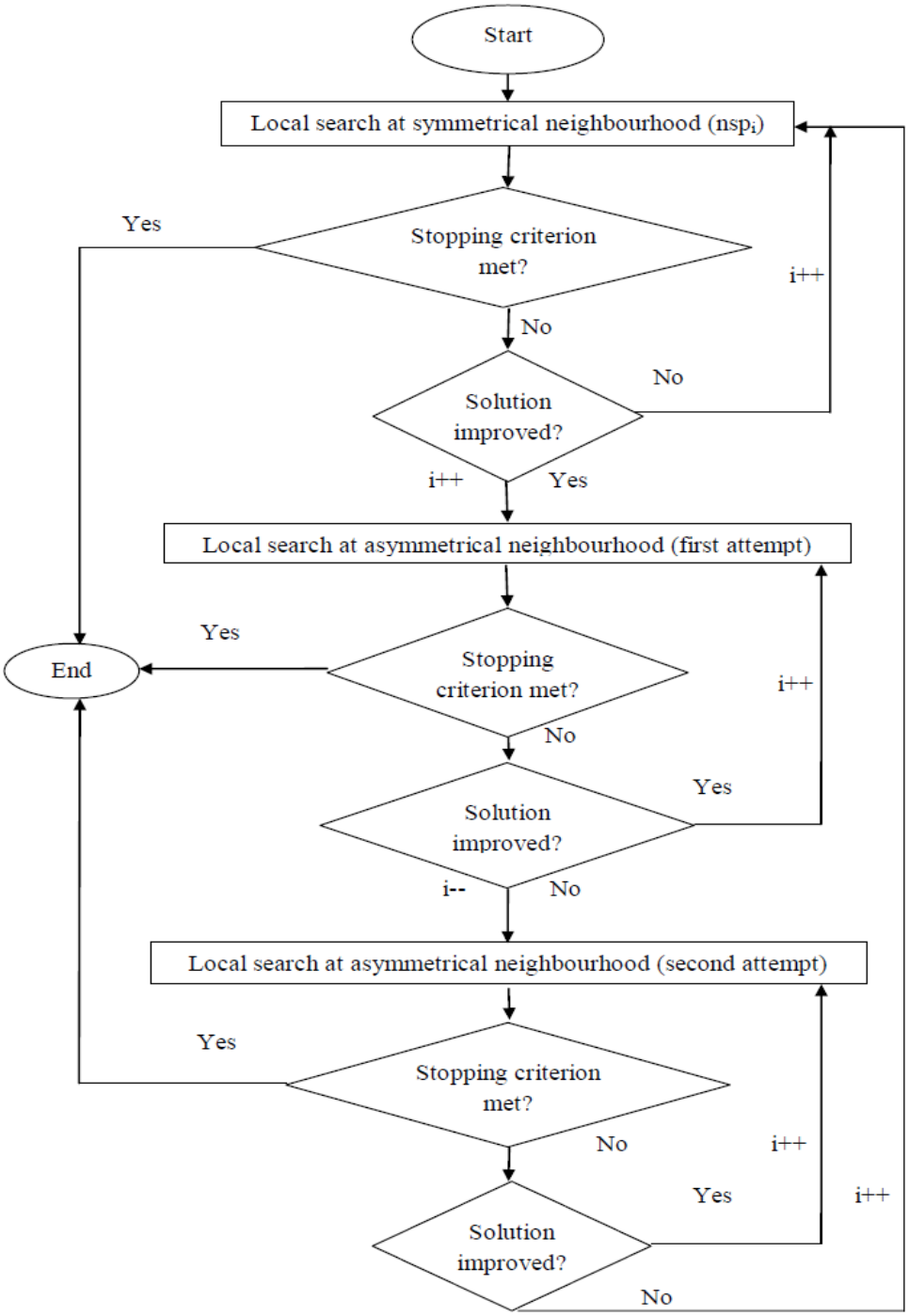


Figure 4.4 Flowchart of the neighbourhood search procedure in the BA-AN

repeated until a stopping criterion had been met. The pseudocode of the BA-AN is given in Figure 4.5.

4.4.1. Experimental setup

In the C++ programme, a counter was placed at the first attempt, second attempt and symmetrical search neighbourhood. Counter A and Counter B counted the number of solutions found at the first and second attempt, respectively. Counter C recorded the number of solutions found in the symmetrical neighbourhood. Counter C accumulated when there was no better solution found at the first and second attempt. With this setting, the BA-AN was tested on the ten test functions (see Table 3.1) and used the same parameter setting as in Table 3.2. For every test function, 100 independent trials were carried out. The average of accuracy, average number of evaluations, standard deviation and average value that were counted by Counter A, B and C were recorded.

4.4.2. Experimental results

On average, the performance of assessment-ii and assessment-iv was better than the performance of assessment-i and assessment-iii in all cases (see Table 4.3 (a)-(d)). Furthermore, the performances of assessment-i and assessment-iii were similar (see third and fourth column of Table 4.4). The difference between assessment-i, assessment-iii and assessment-ii, assessment-iv was only that the first two assessments used the current best solution as the neighbourhood boundary. The search neighbourhood of the latter two assessments was slightly larger since they used the old neighbourhood as the neighbourhood boundary.

1. Initialise population with random solutions.

2. Evaluate and rank of the fitness of the population.

3. While (stopping criterion not met).

    // Forming new population.

4. Select the m solutions.

5. Select the e solutions.

6. Recruit bees for assessment and evaluate their fitness.

7. Select the fittest bee from each patch.

8. Update new neighbourhood size.

9. Update assessment.

10. Assign bees to search randomly and evaluate their fitness.

11. End While.

Figure 4.5 Pseudocode of the BA-AN

Table 4.3 Results of (a) assessment-i, (b) assessment-ii, (c) assessment-iii and (d) assessment-iv

(a)

No.	Function	Avg. accuracy	Std. Dev.	Avg. number of evaluations	Std. Dev.	Avg. Counter A	Avg. Counter B	Avg. Counter C
1	Goldstein & Price (2D)	0.0000	0.0003	504	316.43	133	228	42
2	Schwefel (2D)	0.0000	0.0005	1,539	1,061.94	252	554	81
3	Schaffer (2D)	0.0011	0.0019	406,912	339,280.03	53	166,135	18,789
4	Rosenbrock (10D)	13.7874	71.1378	935,000	0.00	436,174	460,589	3,237
5	Sphere (10D)	0.0013	0.0003	676,413	208,520.83	7,144	654,915	87,941
6	Ackley (10D)	1.986	0.8678	910,000	0.00	431	428,328	21,241
7	Rastrigin (10D)	24.6274	7.6059	885,000	0.00	517	394,450	5,033
8	Martin & Gaddy (2D)	0.0000	0.0003	675	365.62	244	311	47
9	Easom (2D)	0.0000	0.0003	31,856	50,348.48	133	13,217	1,111
10	Griewank (10D)	0.4499	0.1342	4,300,000	0.00	91,816	1,758,171	2,150,013

(b)

No.	Function	Avg. accuracy	Std. Dev.	Avg. number of evaluations	Std. Dev.	Avg. Counter A	Avg. Counter B	Avg. Counter C
1	Goldstein & Price (2D)	0.0000	0.0003	357	214.37	119	142	34
2	Schwefel (2D)	0.0000	0.0005	1,102	648.90	2,239	325	91
3	Schaffer (2D)	0.0000	0.0003	146,784	365,960.40	61	18,635	48,047
4	Rosenbrock (10D)	0.3228	0.4152	935,000	0.00	364,907	531,134	3,959
5	Sphere (10D)	0.0000	0.0003	377,298	300,237.08	1,844	93,720	274,310
6	Ackley (10D)	1.2515	0.2587	910,000	0.00	645	167,027	282,328
7	Rastrigin (10D)	25.6963	8.6835	885,000	0.00	514	337,719	61,767
8	Martin & Gaddy (2D)	0.0000	0.0003	525	240.58	218	242	39
9	Easom (2D)	0.0000	0.0003	5,280	6,768.14	137	1,157	1,099
10	Griewank (10D)	0.2581	0.0902	4,300,000	0.00	27,121	719,863	3,253,017

(c)

No.	Function	Avg. accuracy	Std. Dev.	Avg. number of evaluations	Std. Dev.	Avg. Counter A	Avg. Counter B	Avg. Counter C
1	Goldstein & Price (2D)	0.0000	0.0003	462	248.41	136	193	41
2	Schwefel (2D)	0.0000	0.0004	1,615	1,071.56	260	589	84
3	Schaffer (2D)	0.0015	0.0022	482,416	357,969.46	54	197,183	22,045
4	Rosenbrock (10D)	66.3468	297.81	935,000	0.00	334,053	561,911	4,036
5	Sphere (10D)	0.0013	0.0003	678,096	209,469.37	9,728	656,889	6,851
6	Ackley (10D)	1.8466	0.6759	910,000	0.00	434	426,039	23,527
7	Rastrigin (10D)	22.5664	8.4194	885,000	0.00	558	394,401	5,041
8	Martin & Gaddy (2D)	0.0000	0.0003	600	364.82	235	306	50
9	Easom (2D)	0.0000	0.0003	35,552	49,882.02	134	14,897	1,095
10	Griewank (10D)	0.3964	0.1658	4,300,000	0.00	33,840	1,754,822	2,211,338

(d)

No.	Function	Avg. accuracy	Std. Dev.	Avg. number of evaluations	Std. Dev.	Avg. Counter A	Avg. Counter B	Avg. Counter C
1	Goldstein & Price (2D)	0.0000	0.0003	378	215.18	123	154	35
2	Schwefel (2D)	0.0000	0.0005	1,083	690.71	214	332	79
3	Schaffer (2D)	0.0000	0.0003	120,208	166,848.12	61	16,680	37,889
4	Rosenbrock (10D)	0.3888	0.3661	935,000	0.00	358,950	533,833	7,217
5	Sphere (10D)	0.0000	0.0002	345,933	291,791.3	1,841	91,538	245,749
6	Ackley (10D)	1.2360	0.1912	910,000	0.00	656	178,352	270,992
7	Rastrigin (10D)	25.3088	7.7443	885,000	0.00	515	335,527	63,959
8	Martin & Gaddy (2D)	0.0000	0.0003	525	221.73	213	243	39
9	Easom (2D)	0.0000	0.0003	4,928	7,593.43	135	1,068	1,032
10	Griewank (10D)	0.2550	0.1285	4,300,000	0.00	27,341	740,776	323,188

Table 4.4 Significance of the difference

No.	Function	between assessment-i and assessment-iii		between assessment-ii and assessment-iv	
		Accuracy	Number of evaluations	Accuracy	Number of evaluations
1	Goldstein & Price (2D)	NS	NS	NS	NS
2	Schwefel (2D)	NS	NS	NS	NS
3	Schaffer (2D)	NS	NS	NS	NS
4	Rosenbrock (10D)	NS	NS	NS	NS
5	Sphere (10D)	NS	NS	NS	NS
6	Ackley (10D)	NS	NS	NS	NS
7	Rastrigin (10D)	NS	NS	NS	NS
8	Martin & Gaddy (2D)	NS	NS	NS	NS
9	Easom (2D)	NS	NS	NS	NS
10	Griewank (10D)	NS	NS	NS	NS

Because of this, it could be said that the poor performance of assessment-i and assessment-iii was caused by the limited neighbourhood search area in these assessments. This small area increased the chance that the bees might miss out the potential solutions that were on the other side of the current best solution. From this observation, it was necessary to allocate an amount of neighbourhood on both sides, as exhibited in assessment-ii and assessment-iv.

Between assessment-ii and assessment-iv, the difference was not statistically significant (see last two columns of Table 4.4). This similar result suggested that there was no advantage to perform neighbourhood search on one particular side prior to the other and vice versa. Theoretically, if the bees were focusing on a particular side due to the improvement made, the bees might stand a chance to miss out other better solutions that were on the other side. Moreover, there was no clue of which direction could help the bees to get to the global optimum quickly. The global optimum did not necessarily lie in the same direction as the path made by the current best solution. Furthermore, getting to a higher (lower) position in the maximisation (minimisation) problem in the fitness landscape might lead to local optima, which were the points that the bees should avoid.

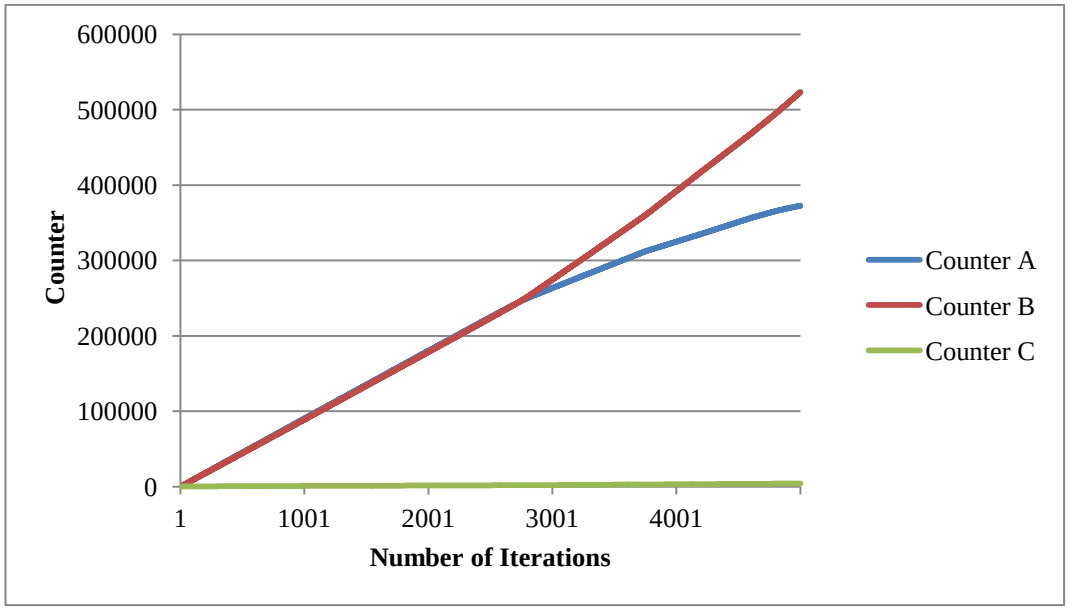
4.4.3. Discussions

On the Goldstein & Price, Schwefel, Rosenbrock and Martin & Gaddy, more than one third of the solutions were obtained at the first attempt, as revealed by the value of Counter A. This means that the algorithm made improvements most of the time and so encouraged the bees to perform an asymmetrical neighbourhood

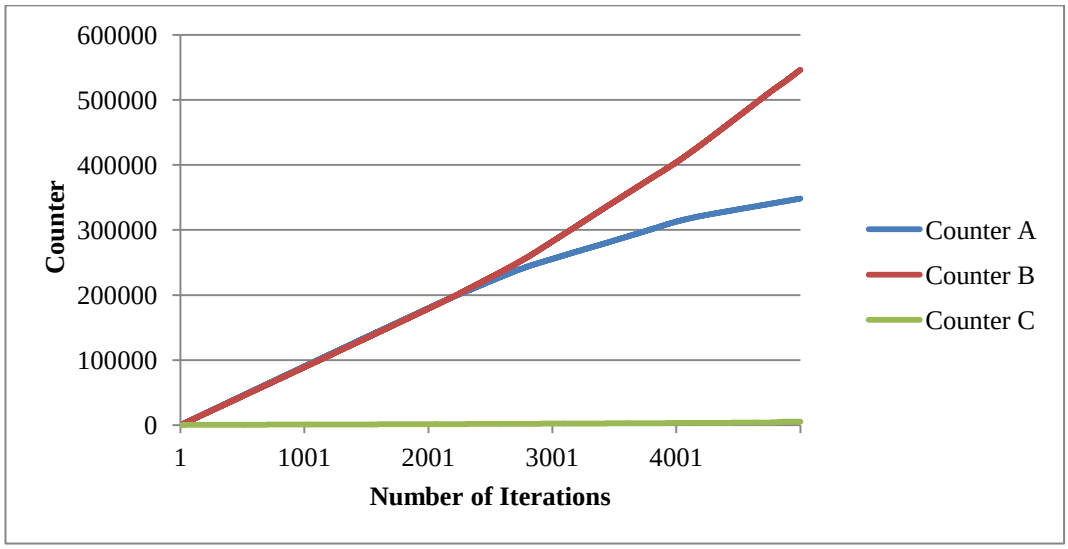
search. The bees had no difficulty in locating better solutions even at the first attempt. Any better solution would drive the bees to repeat the neighbourhood search in the asymmetrical neighbourhood. This scenario accumulated the value of Counter A. The value of Counter B was also higher compared to that of Counter C when solving these functions. This indicated that after failing to find a better solution at the first attempt, the bees found one straight away at the second attempt. As a result, the bees rarely performed symmetrical neighbourhood search, as indicated by the low value of Counter C. For better visualisation, Figure 4.6 shows the value of counter accumulations with respect to the number of iterations on the Rosenbrock.

On the other hand, on the Rastrigin and Easom, most of the time the bees were not able to find a better solution at the first attempt, forcing them to perform the second attempt. On the Rastrigin, even though most of the improvements were made at the second attempt (see Figure 4.7), the algorithm failed to reach the global optimum. On the Easom, the flat surface challenged the algorithm. If the bees landed on the flat surface, any modification of the neighbourhood size would not benefit the algorithm. The algorithm kept adopting the symmetrical neighbourhood size until at least one bee landed in the hole. This was the reason why the value of Counter C was about as high as that of Counter B.

Because of failing to find a better solution at the first and second attempt, the Schaffer, Ackley and Griewank used a symmetrical neighbourhood for the neighbourhood search. The high value of Counter C for these functions implied that the asymmetrical neighbourhood was not able to deal with the numerous local

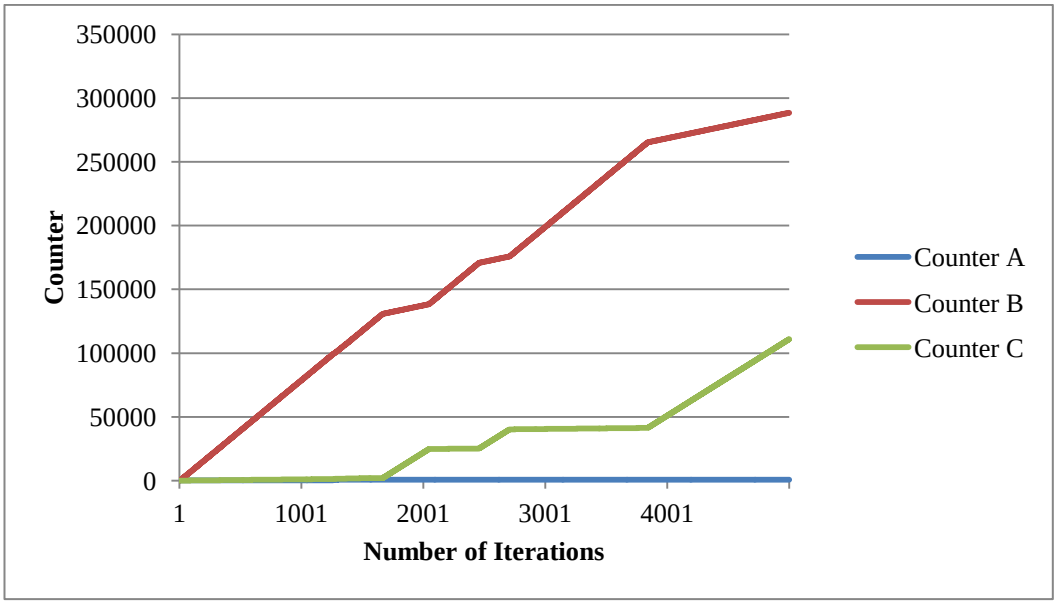


(a)

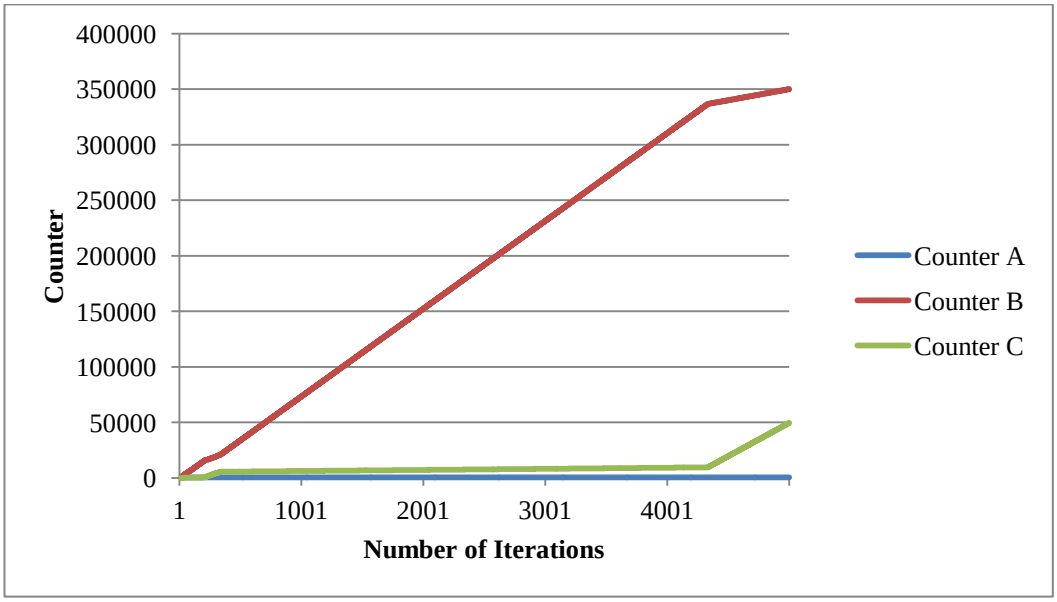


(b)

Figure 4.6 Counter gained by (a) assessment-ii and (b) assessment-iv on the Rosenbrock



(a)



(b)

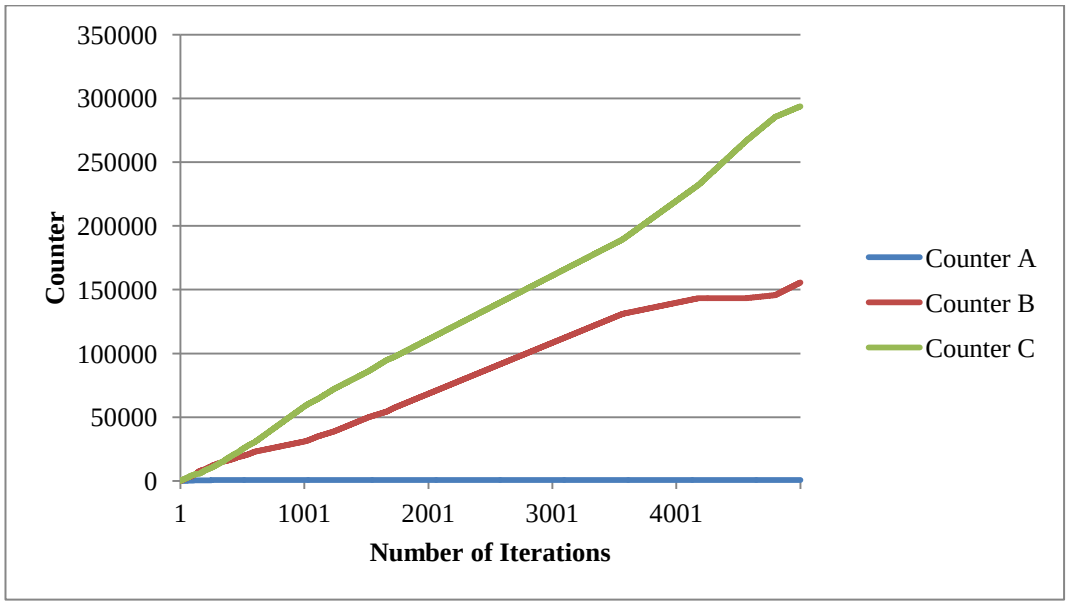
Figure 4.7 Counter gained by (a) assessment-ii and (b) assessment-iv on the Rastrigin

optima. For example, Figure 4.8 shows the value of counter accumulations with respect to the number of iterations on the Ackley.

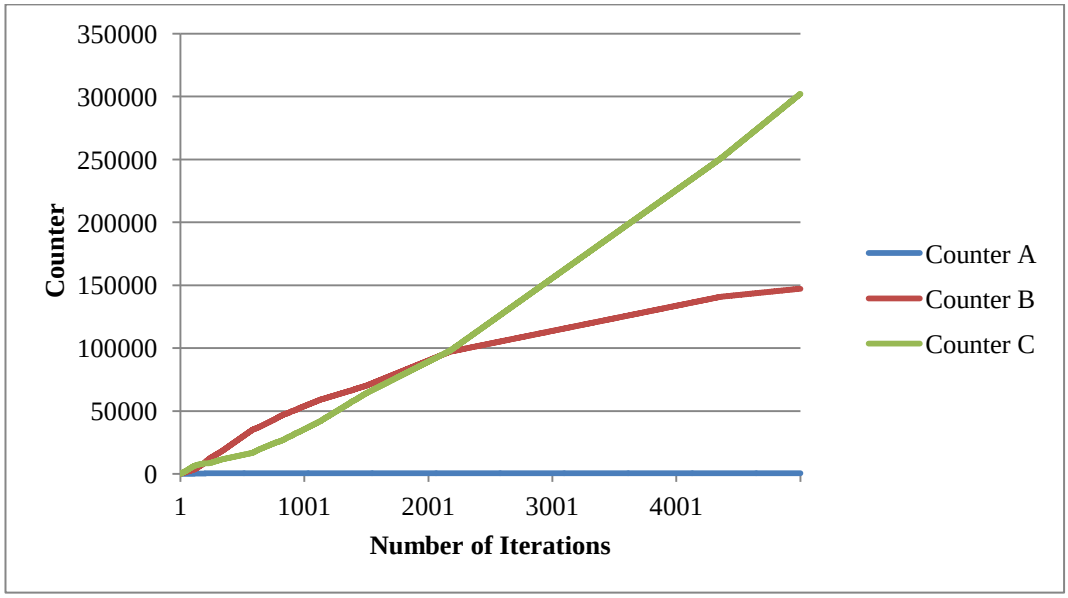
Even though there were no local optima on the Sphere, the BA-AN tended to use the symmetrical neighbourhood for the neighbourhood search. This might be caused by the high dimensionality here. High dimensionality might lead the bees to a higher position (since the Sphere was a minimisation problem), which would result in no improvement to the solution.

4.4.4. Comparison against the BA

The experimental results that were obtained by the BA-AN assessment-ii and assessment-iv were compared to the one obtained by the BA. The significance of the difference by the Mann Whitney test shows that the BA-AN produced a similar performance in all cases except on the Rosenbrock and Sphere (Table 4.5). The solution accuracy obtained on the Rosenbrock was significantly improved by the BA-AN. This promising result might be caused either by the asymmetrical search neighbourhood or by the adaptive enlargement, which will be discussed in the next section. Meanwhile, on the Sphere, the BA performed better than the BA-AN. In the BA-AN, the asymmetrical search neighbourhood might be rarely used due to less solution improvements being made. As a result, the first and second attempt might cause more time consumption by the algorithm to optimise the problem, even though it did not yield a better solution. On the other hand, the algorithm could locate the global optimum faster by only adopting the symmetrical search neighbourhood, as shown by the high value of Counter C,



(a)



(b)

Figure 4.8 Counter gained by (a) assessment-ii and (b) assessment -iv on the Ackley

Table 4.5 Significance of the difference between the BA-AN and BA

No.	Function	BA-AN (assessment-ii)		BA-AN (assessment-iv)	
		Accuracy	Number of evaluations	Accuracy	Number of evaluations
1	Goldstein & Price (2D)	NS	NS	NS	NS
2	Schwefel (2D)	NS	NS	NS	NS
3	Schaffer (2D)	NS	NS	NS	NS
4	Rosenbrock (10D)	S	-	S	-
5	Sphere (10D)	S	S	S	S
6	Ackley (10D)	NS	-	NS	-
7	Rastrigin (10D)	NS	-	NS	-
8	Martin & Gaddy (2D)	NS	NS	NS	NS
9	Easom (2D)	NS	NS	NS	NS
10	Griewank (10D)	NS	-	NS	-

which was discussed in the previous section.

4.4.5. Comparison against the BA-NE

The comparison of the BA-AN assessment-ii and assessment-iv against the BA-NE is presented in Table 4.6. The BA-AN was outperformed by the BA-NE on the Goldstein & Price, Rosenbrock, Ackley, Martin & Gaddy and Griewank. However, their performance was similar in other cases.

The promising solution on the Rosenbrock obtained by the BA-AN was better than the one obtained by the BA. This might be caused by the adaptive enlargement, rather than by the use of an asymmetrical search neighbourhood. This was proved by the solution accuracy on the Rosenbrock that was better obtained by the BA-NE. In addition, in Section 3.2, the BA-NE failed to reach the global optimum on the Rosenbrock, Ackley, Rastrigin and Griewank. The asymmetrical search neighbourhood also failed to improve this situation. In fact, the solution accuracy on the Rosenbrock and Griewank obtained by the BA-AN deteriorated. Therefore, it could be said that an asymmetrical search neighbourhood did not have a positive influence on the BA-AN.

4.4.6. Discussions of TRIZ

Despite the fact that it had never been studied, the asymmetrical search neighbourhood was derived from a TRIZ inventive solution. The failure to improve the performance of the algorithm after TRIZ analysis might be caused by two factors. First, a different measurement of asymmetry might contribute to this

Table 4.6 Significance of the difference between the BA-AN and BA-NE

No.	Function	BA-AN (assessment-ii)		BA-AN (assessment-iv)	
		Accuracy	Number of evaluations	Accuracy	Number of evaluations
1	Goldstein & Price (2D)	NS	S	NS	S
2	Schwefel (2D)	NS	NS	NS	NS
3	Schaffer (2D)	NS	NS	NS	NS
4	Rosenbrock (10D)	S	-	S	-
5	Sphere (10D)	NS	NS	NS	NS
6	Ackley (10D)	S	-	S	-
7	Rastrigin (10D)	NS	-	NS	-
8	Martin & Gaddy (2D)	NS	S	NS	S
9	Easom (2D)	NS	NS	NS	NS
10	Griewank (10D)	S	-	S	-

failure. In this work, the asymmetrical search neighbourhood was formed by setting the sum of the old neighbourhood and enlargement on one side of the current best solution and only the old neighbourhood on another side. Other individuals might set the asymmetrical search neighbourhood using different measurements, such as increasing/ decreasing the degree of the asymmetry based on their preference. Different measurements of asymmetrical search neighbourhood might affect the algorithm differently.

The other reason was that the TRIZ solution might hide in six other inventive solutions (1. Segmentation, 3. Local Quality, 14. Curvature, 24. Intermediary, 31. Porous Materials and 39. Inert Atmosphere). This list, however, does not guarantee a solution. On the other hand, they provided a number of potential solutions, in order to reduce the number of trial and error attempts (Altshuller 2001).

4.5. Application

The effectiveness of the BA-AN was tested on the gear train problem (see Appendix A). The parameter setting and stopping criterion were set the same as described at Section 3.3.1. The result was then compared against the BA, BA-NE, UPSOm (Parsopoulos and Vrahatis 2005) and ABC (Akay and Karaboga 2010). The experimental result revealed that the BA-AN assessment-ii and assessment-iv failed to beat the average solution and % error produced by the UPSOm and ABC (see Table 4.7). Also, the overall solutions produced by the BA-AN was poor.

Table 4.7 Comparison against other algorithms on the gear train problem

	UPSOm	ABC	BA	BA-NE	BA-AN (assessment-ii)	BA-AN (assessment-iv)
Avg. Solution	3.80562E-08	3.641339E-10	6.84E-05	2.12E-06	4.53E-06	6.79E-06
Std. Dev.	1.09631E-07	5.525811E-10	0.000366946	4.43477E-06	1.16133E-05	2.26966E-05
Best Solution	2.70085E-12	2.700857E-12	9.92158E-10	1.54505E-10	9.92E-10	1.31252E-08
x_1	NA	49	47	43	47	46
x_2	NA	16	12	13	26	12
x_3	NA	19	26	21	12	21
x_4	NA	43	46	44	46	38
Gear ratio	NA	0.144281	0.144311	0.144292	0.14431	0.144165
% error	NA	0.001%	0.022%	0.009%	0.022%	-0.079%

This fact was supported by the large standard deviation obtained by the BA-AN assessment-ii and assessment-iv. The large standard deviation also suggested that these two assessments were not good at producing a set of robust solutions. This was proved by the Mann Whitney test where there was no statistical difference between the BA, BA-AN assessment-ii and BA-AN assessment-iv (see Table 4.8). On the other hand, the performance of the BA-NE was better than these three algorithms. This experimental result again implied that an asymmetrical search neighbourhood with the stated measurement was not useful to the BA-AN.

4.6. Summary

A symmetrical search neighbourhood was normally applied in the BA. This kind of search neighbourhood was also adopted in the BA-NE. An asymmetrical search neighbourhood, which was derived from a TRIZ inventive solution, was used to replace the symmetrical search neighbourhood in the BA-NE. The BA-NE with such a neighbourhood was developed and named BA-AN. In the BA-AN, four different types of asymmetrical search neighbourhood were analysed. The results suggested that a certain neighbourhood area should be allocated on both sides of the current best solution to speed up the optimisation. In addition, evaluating the solution on one side of the current best solution before the other side and vice versa simply gave a similar performance.

This work has also proved that an asymmetrical search neighbourhood formed by the sum of the old neighbourhood and enlargement on one side of the current best

Table 4.8 Significance of the difference between the BA, BA-NE and BA-AN on the gear train problem

Methods	BA	BA-NE	BA-AN (assessment-ii)	BA-AN (assessment-iv)
BA	-	S	NS	NS
BA-NE	S	-	S	S
BA-AN (assessment-ii)	NS	S	-	NS
BA-AN (assessment-iv)	NS	S	NS	-

solution and only the old neighbourhood on another side did not bring benefits to the BA-AN. The experimental results from the gear train problem reinforced this finding. Thus, it could be concluded that there was no advantage of using an asymmetrical search neighbourhood as opposed to a symmetrical one.

Chapter 5

COMBINATION OF ADAPTIVE ENLARGEMENT AND REDUCTION IN THE SEARCH NEIGHBOURHOOD IN THE BEES ALGORITHM

5.1. Preliminaries

Based on work discussed in Chapter 3, adaptive enlargement of the search neighbourhood helped the BA-NE to reach better solutions faster when the search space was smooth. However, this new approach gave no benefit when the search surface was highly multi-pocketed, which was created by a cosinusoidal noise component. The failure to reach the global optimum in such problems implied that the policy of neighbourhood enlargement was not always helpful in every case. In other words, the method benefited when the surface was smooth but became useless with a noisy landscape. Furthermore, the study in Chapter 4 revealed that asymmetrical search neighbourhood also failed to reach the global optimum of problems that featured a noisy surface. Because of this, the symmetrical search neighbourhood was readopted in this work.

This chapter is organised as follows: Section 5.2 reviews the shrinking method in the BA. Section 5.3 describes the current problem that motivates this work. Section 5.4 explains the proposed idea, experimental setup and results. Three types of engineering design problems were used to test the capability of the modification and they are presented in Section 5.5. Section 5.6 summarises the achievements of this work.

5.2. ‘Neighbourhood shrinking’ method

The ‘neighbourhood shrinking’ method was initially introduced by (Ghanbarzadeh 2007). The method stated that a large patch size should be set at the beginning of the optimisation. The patch size was then decreased by a half if the neighbourhood search did not bring any improvement in the solution. Conversely, the patch size was kept unchanged if the forager bees managed to find better solutions.

In contrast, the idea proposed in this work was to combine the adaptive enlargement and reduction in the search neighbourhood. The initial neighbourhood size was not necessarily large but kept equal to the values as in Table 3.2. It was anticipated that the reduction of neighbourhood size was needed to stimulate the optimisation progress.

5.3. Current problem

The BA-NE failed to find the global optimum with an accuracy of 0.001 after 5000 iterations on the Rosenbrock, Ackley, Rastrigin and Griewank. This might have been caused by the current neighbourhood size becoming large. This neighbourhood size could be larger than the distance of current best solution and global optimum. As a result, the bees associated with the best solution might overshoot the global optimum in the next iteration. To solve the overshooting problem, the distance of the best solution and the global optimum needs to be investigated.

5.3.1. Euclidean Distance

The distance of two points in an *n*-space can be measured using the Euclidean Distance (ED) formula (Fiedler 2011). Mathematically, ED is defined as the length of the line segment that connects points p and q. If p is located at (p_1, p_2, p_3,…,p_n) and q is at (q_1, q_2, q_3,…,q_n), the ED between points p and q is summarised as Equations 5.1:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad (5.1)$$

Having Equation 5.2 to hand, the distance of the best solution obtained by the BA-NE from the global optimum was calculated and denoted as r (see Table 5.1.) Note that only problems for which the BA-NE failed to find the global optimum, i.e., the Rosenbrock, Ackley, Rastrigin and Griewank, were considered for this calculation.

5.3.2. Calibration

The failure to reach an acceptable solution with the neighbourhood size as in Table 3.2 implied that there was a rough surface between the best solution obtained by the BA-NE and the global optimum. The neighbourhood size which was used before failed to adapt to this rough surface. Consequently, it was not possible to find the global optimum, unless the parameter setting was retuned. In order to reach the global optimum on these problems without retuning parameters, the size of search space of each problem was calibrated. This calibration meant that the standard search space was shrunk to a reasonably smaller one so that the bees could easily locate the global optimum. This procedure provided an insight

Table 5.1 Distance between the final solution and the global optimum

No.*	Function	Solution	r
4	Rosenbrock (10D)	0.0289	0.3172
7	Ackley (10D)	0.5573	0.2344
8	Rastrigin (10D)	3.9915	1.9912
10	Griewank (10D)	0.0475	9.2214

*The numbering is based on Table 3.1

into whether the neighbourhood size should be smaller towards the global optimum.

The worst acceptable solution that was captured after the calibration of the search space was recorded. Its distance from the global optimum was also calculated and denoted by r_{calib}. This kind of solution was selected because it indicated that a solution must not be farther than r_{calib} to be an acceptable one. The calibrated search space, worst acceptable solution and the r_{calib} are shown in Table 5.2.

Compared to r_{calib}, the sum of the old neighbourhood and enlargement was too large to capture the global optimum, as shown in Table 5.3. A large neighbourhood might lead the bees to overshoot the global optimum. Therefore, it was necessary to reduce the neighbourhood size especially when the solution did not improve after a certain time.

5.4. BA-NER

The procedure of neighbourhood reduction was developed on the BA-NE and the proposed algorithm named BA-NER, which stands for the Bees Algorithm-neighbourhood enlargement and reduction. The BA-NER also adopted the same initialisation procedure as the BA-NE. The n solutions that were obtained after random initialisation were ranked in descending order. The top m sites were selected for the neighbourhood search. A number of elite sites (e) were selected

Table 5.2 Calibration

No.*	Function	Calibrated search space	Worst acceptable solution	r_{calib}
4	Rosenbrock (10D)	[0.9,1.1]	0.0002	0.0002
7	Ackley (10D)	[-0.0005,0.0005]	0.00083	0.0007
8	Rastrigin (10D)	[-0.0005,0.0005]	0.00026	0.0005
10	Griewank (10D)	[99.99,100.01]	-9.94E-06	0.0101

*The numbering is based on Table 3.1

Table 5.3 Comparison on the sum of old neighbourhood and enlargement and r_{calib}

No.*	Function	old ngh + enlargement	r_{calib}
4	Rosenbrock (10D)	0.0024	0.0002
7	Ackley (10D)	0.8166	0.0007
8	Rastrigin (10D)	0.0117	0.0005
10	Griewank (10D)	1.6209	0.0101

*The numbering is based on Table 3.1

from the m sites. Each elite and non-elite sites (m-e) received nep and nsp forager bee(s) respectively.

Figure 5.1 is the flow chart of the neighbourhood search performed in the BA-NER, where i is the index of forager bees (nsp and nep) and i=1,2,3,..., nsp_{max} (nep_{max}). In this algorithm, two new parameters were to be introduced. They were the allowed number of consecutive iterations (ci) and the amount of neighbourhood reduction (nr) and are included in the flow chart. If there were no forager bees making an improvement after ci had elapsed, the neighbourhood size was reduced by the factor nr. The stopping criteria in the flowchart were either when the solution obtained met the preset threshold or the index i > nsp_{max} (i > nep_{max}).

After all forager bees of every m sites completed the neighbourhood search, the unselected bees, (n-m) performed the global search. A set of solutions obtained upon completion of the global search would then be ranked in the next iteration. This process was repeated until any stopping criterion was met. Figure 5.2 shows the pseudocode of the BA-NER.

5.4.1. Experimental setup

BA-NER was tested on the ten mathematical benchmarks (see Table 3.1). As illustrated in the flowchart, two new parameters were included, which were the allowed number of consecutive iteration (ci) and the amount of neighbourhood reduction (nr). In this experiment, the neighbourhood size was reduced to 1/10 of the current size if there was no improvement made by the forager bees after 10

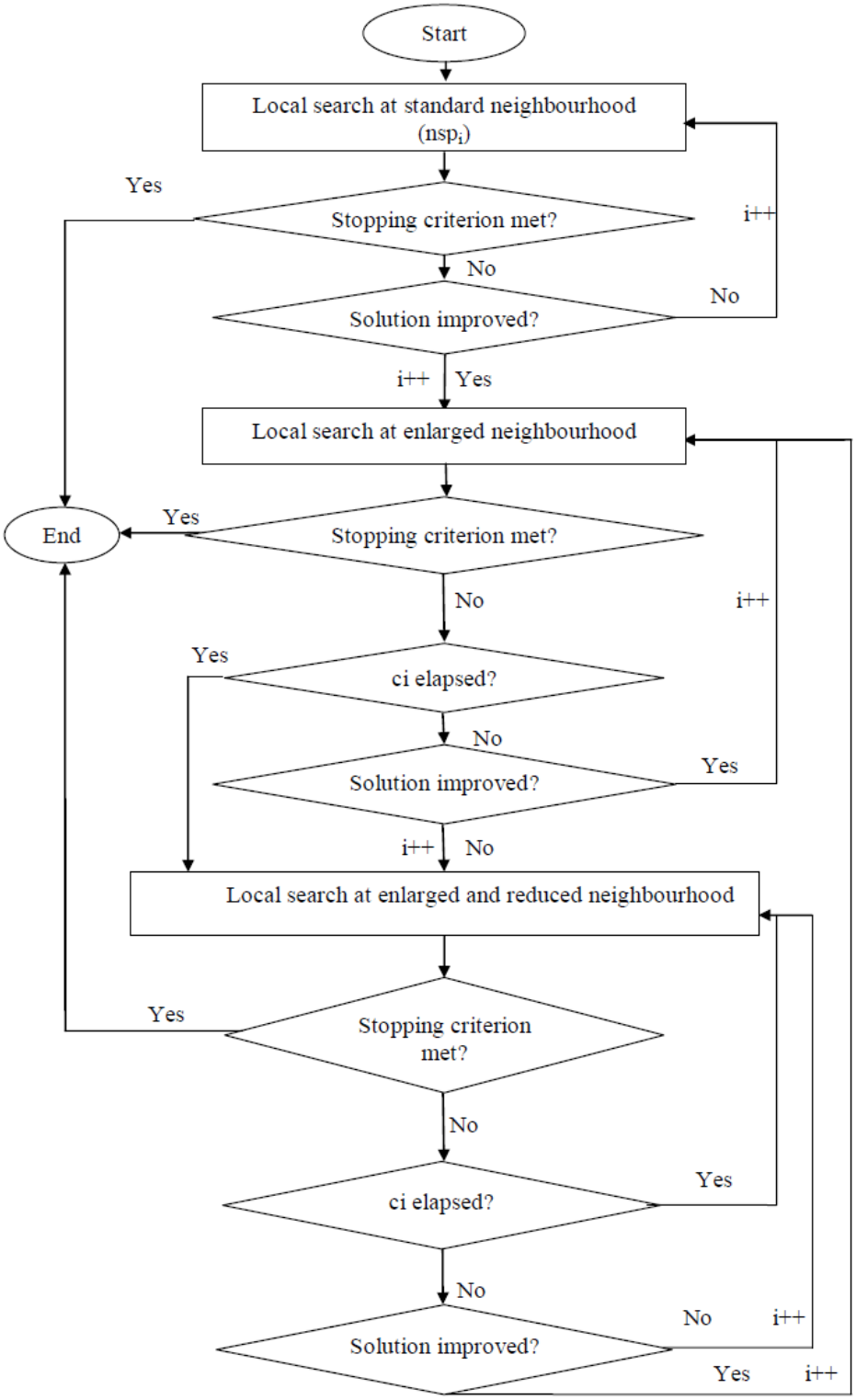


Figure 5.1 Flowchart of the neighbourhood search procedure in the BA-NER

1. Initialise population with random solutions.

2. Evaluate and rank of the fitness of the population.

3. While (stopping criterion not met).

    // Forming new population.

4. Select the m solutions.

5. Select the e solutions.

6. Recruit bees and evaluate their fitness.

7. Select the fittest bee from each patch.

8. Update new neighbourhood size.

9. Update number of iteration.

10. Update amount of neighbourhood reduction.

11. Assign bees to search randomly and evaluate their fitness.

12. End While.

Figure 5.2 Pseudocode of the BA-NER

consecutive iterations. The amount of neighbourhood reduction and number of iterations specified above were part of the parameter setting and were of course not obliged to have these values. Since the aim of this work was to study the effect of neighbourhood reduction rather than finding the global optimum, no attempt was made to tune these new parameters.

In this experiment, BA-NER was designed to adapt to two groups of problems; Group A and Group B. Group A were the problems for which BA-NE could not find the global optimum (Rosenbrock, Ackley, Rastrigin and Griewank), whereas Group B were the problems that had been successfully solved by the BA-NE (Goldstein & Price, Schwefel, Schaffer, Sphere, Martin & Gaddy, Easom). In both Group A and Group B, the adaptive enlargement of the search neighbourhood was employed as long as the solutions improved. The only difference was:

5.4.1.1. Group A

If there was no improvement in the solution after 10 consecutive iterations, the neighbourhood size was reduced to 1/10 of the current size. This procedure was repeated until the neighbourhood size became smaller than r_{calib}. After reduction, the neighbourhood size that was smaller than r_{calib} was considered as the limit (see Table 5.4). Once the neighbourhood size became equal to or smaller than the limit, it was increased back to its initial size and the optimisation was noted as completing one cycle. If there were more than one cycle and no improvement had been made, it was assumed that the bees were entrapped in the local optima. Also, this situation implied that the adopted neighbourhood size failed to deal with the current landscape.

Table 5.4 Range of neighbourhood

No.	Function	Range of neighbourhood size
1	Goldstein & Price (2D)	$0.005 \geq ngh$
2	Schwefel (2D)	$0.5 \geq ngh$
3	Schaffer (2D)	$3.0 \geq ngh$
4	Rosenbrock (4D)	$0.0015 \geq ngh \geq 0.00015$
5	Sphere (10D)	$0.05 \geq ngh$
6	Ackley (10D)	$0.7 \geq ngh \geq 0.00007$
7	Rastrigin (10D)	$0.01 \geq ngh \geq 0.0001$
8	Martin & Gaddy (2D)	$0.1 \geq ngh$
9	Easom (2D)	$0.5 \geq ngh$
10	Griewank (10D)	$1.5 \geq ngh \geq 0.0015$

5.4.1.2. Group B

For this group, the limit was not set. This meant that the neighbourhood size could be of any value, with a 1/10 reduction factor. The purpose of this procedure was to investigate which size of neighbourhood contributed most in reaching the global optimum. In addition, the bees were not risking entrapment in local optima in these problems. This was proved by the success of the BA-NE in dealing with these problems in Section 3.2.2.

Apart from ci, nr, and limit, other parameters were set as shown in Table 3.2. Stopping criteria were also the same as described in Section 3.2. For each benchmark, BA-NER was run 100 times.

5.4.2. Experimental results

The experimental results were compared to the ones obtained by other algorithms (see Table 5.5). The performance of the BA-NER was comparable to the PSO, EA and ABC. Moreover, without considering the significance, the average final solution obtained by the BA-NER was the best on the Rosenbrock, Ackley and Griewank, compared to the BA and BA-NE. Also, the average number of evaluations was improved by the BA-NER on the Goldstein & Price, Schwefel, Schaffer, Sphere, Ackley and Griewank. For a clearer comparison, the significance of the difference of the accuracy and number of evaluations between BA-NER and BA-NE is shown in Table 5.6. The comparison on the number of evaluations on the Rosenbrock and Rastrigin were omitted since they were the same in all runs.

Table 5.5 Comparison on (a) accuracy and (b) average number of evaluations against other algorithms

(a)

No.	Functions	PSO		EA		ABC		BA		BA-NE		BA-NER	
		Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.
1	Goldstein & Price (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0003	0.0000	0.0003	0.0000	0.0003
2	Schwefel (2D)	4.7376	23.4448	4.7379	23.4448	0.0000	0.0000	0.0000	0.0005	0.0000	0.0005	0.0000	0.0005
3	Schaffer (2D)	0.0000	0.0000	0.0009	0.0025	0.0000	0.0000	0.0000	0.0003	0.0000	0.0003	0.0000	0.0003
4	Rosenbrock (10D)	0.5998	1.0436	61.5213	132.6307	0.0965	0.0880	44.3210	112.29	0.0508	0.0337	0.0046	0.0059
5	Sphere (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0002	0.0000	0.0002
6	Ackley (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.2345	0.3135	1.2297	0.2383	0.0240	0.2316
7	Rastrigin (10D)	0.1990	0.4924	2.9616	1.4881	0.0000	0.0000	24.8499	8.3306	23.3201	9.1703	25.4908	8.2092
8	Martin & Gaddy (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0003	0.0000	0.0003
9	Easom (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	2.0096	0.0000	0.0003	0.0000	0.0003	0.0000	0.0003
10	Griewank (10D)	0.0008	0.0026	0.0210	0.0130	0.0052	0.0078	0.3158	0.1786	0.1912	0.1024	0.0495	0.0323

(b)

No.	Functions	PSO		EA		ABC		BA		BA-NE		BA-NER	
		Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.	Avg. evaluations	Std. Dev.
1	Goldstein & Price (2D)	3,262	822	2,002	390	2,082	435	504	211	384	168	360	166
2	Schwefel (2D)	84,572	90,373	298,058	149,638	4,750	1,197	1,140	680	1,140	701	950	570
3	Schaffer (2D)	28,072	21,717	219,376	183,373	21,156	13,714	121,088	174,779	132,176	157,520	38,368	113,425
4	Rosenbrock (10D)	492,912	29,381	500,000	0	497,728	16,065	935,000	0	935,000	0	935,000	0
5	Sphere (10D)	171,754	7,732	36,376	2,736	13,114	480	285,039	277,778	325,125	252,987	14,841	4,495
6	Ackley (10D)	236,562	9,119	50,344	3,949	18,664	627	910,000	0	910,000	0	88,816	118,495
7	Rastrigin (10D)	412,440	67,814	500,000	0	207,486	57,568	885,000	0	885,000	0	885,000	0
8	Martin & Gaddy (2D)	1,778	612	1,512	385	1,498	329	600	259	450	187	450	182
9	Easom (2D)	16,124	15,942	36,440	28,121	1,542	201	5,280	6,303	4,576	3,344	4,752	3,559
10	Griewank (10D)	290,466	74,501	490,792	65,110	357,438	149,129	4,300,000	0	4,300,000	0	4,171,860	654,013

Table 5.6 Significance of the difference between the BA-NER and BA-NE

No.	Function	Accuracy	Number of evaluations
1	Goldstein & Price (2D)	NS	NS
2	Schwefel (2D)	NS	NS
3	Schaffer (2D)	NS	S
4	Rosenbrock (10D)	S	-
5	Sphere (10D)	S	S
6	Ackley (10D)	S	S
7	Rastrigin (10D)	NS	-
8	Martin & Gaddy (2D)	NS	NS
9	Easom (2D)	NS	NS
10	Griewank (10D)	S	S

5.4.3. Discussions

Compared to the BA-NE, the performance of the BA-NER was significantly better in finding the global optimum of the Schaffer, Rosenbrock, Sphere, Ackley and Griewank. These experimental results suggested that the neighbourhood reduction was beneficial to the BA-NER. Despite this success, the performance of both algorithms was similar on the Goldstein & Price and Martin & Gaddy. This was because the rapid progress in the optimisation did not allow the neighbourhood reduction procedure to take place. In other words, the algorithm never reached 10 consecutive iterations without an improvement in the neighbourhood search. In order to investigate whether the neighbourhood reduction had an effect on the Goldstein & Price and Martin & Gaddy, the ci was altered. The ci was gradually decreased until at least a neighbourhood reduction occurred once. Table 5.7 shows that the 99 of 100 runs reached the global optimum when the neighbourhood size was 0.005 and ci was 2 on the Goldstein & Price. On the Martin & Gaddy, there was no single iteration that did not improve the neighbourhood search. This analysis proved that neighbourhood reduction was not crucial to the BA-NER in order to find an acceptable solution on these two functions. On the other hand, BA-NER was able to find the global optimum without reducing the size of the search neighbourhood.

On the Schwefel, the neighbourhood reduction also did not improve the results that were obtained by the BA-NE. Of 100 runs, 87 reached the global optimum with the original neighbourhood size, which was 0.5 (see Figure 5.3). In fact, the

Table 5.7 Neighbourhood reduction on the Goldstein & Price and Martin & Gaddy

No.*	Function	ci	Original ngh	Number of runs reached acceptable solution	0.1x original ngh	Number of runs reached acceptable solution
1	Goldstein & Price (2D)	2	0.005	99	0.0005	1
8	Martin & Gaddy (2D)	1	0.1	100	-	-

*The numbering is based on Table 3.1

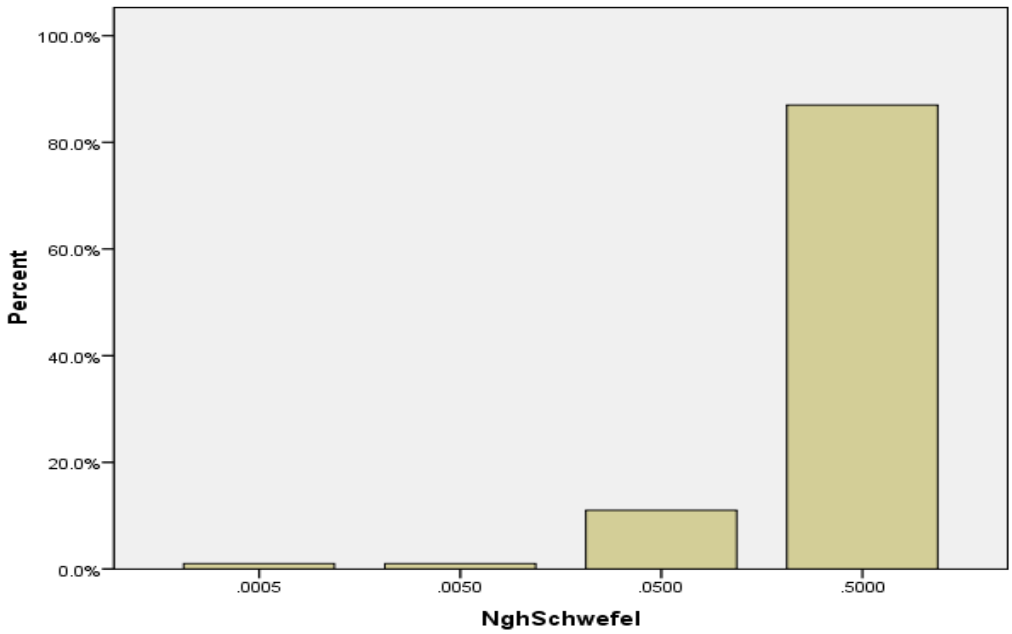


Figure 5.3 Neighbourhood size preference on the Schwefel

number of iterations was getting larger as the neighbourhood size decreased. It was 111, 771 and 4921, when the neighbourhood size was 0.05, 0.005 and 0.0005, respectively. This size was just too small and thus slowed down the progress. Hence, the neighbourhood reduction was not necessary on the Schwefel.

On the other hand, neighbourhood reduction stimulated a rapid progress on the Schaffer. 74 runs reached the global optimum when the neighbourhood size was 0.3 and this was one tenth of the original size (see Figure 5.4). This analysis supported the fact that the global optimum was surrounded by a high number of local minima (Zhao et al. 2009). For better visualisation, the contour of this function within [-100, 100] of search space is shown in Figure 5.5 (Zhao et al. 2009). In the figure, local optima which were near the edge were not obvious. In this area, the larger neighbourhood size, which was 3.0, was able to speed up the convergence towards the global optimum. However, the surface was getting noisier as optimisation was approaching the global optimum, which was located at [0, 0]. When the best solution was getting closer to the global optimum, the neighbourhood size (3.0) was no longer able to accommodate the bees to reach the global optimum. With this size, the bees tended to overshoot the global optimum. Fortunately, a neighbourhood size of 0.3 minimises this risk. This is reflected in the fact that a neighbourhood size of 0.3 contributed to the success of 74 runs.

Meanwhile, on the Rosenbrock, the algorithm reached the global optimum in 9 runs when the neighbourhood size was 0.0015. The success of the remaining 91

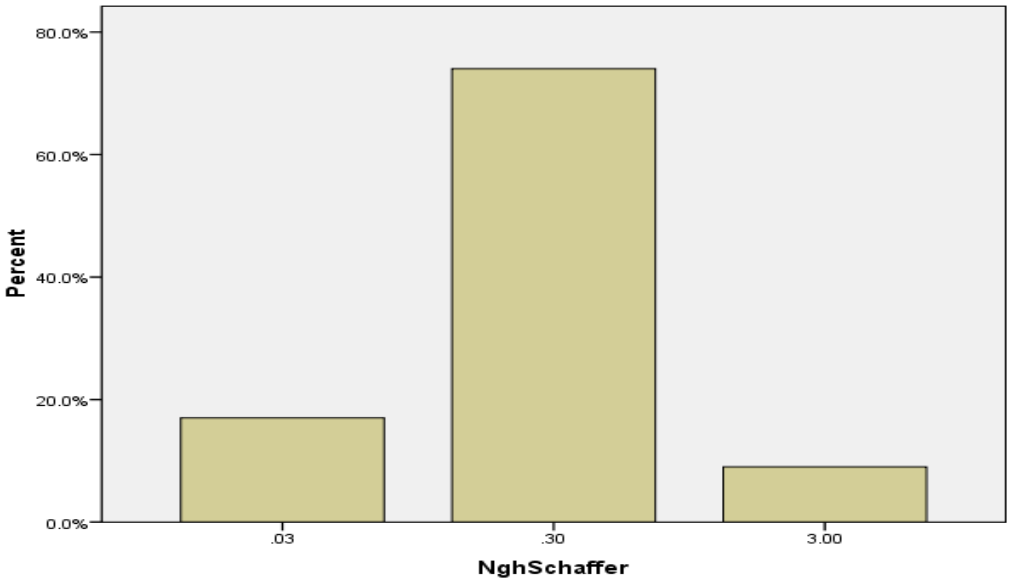


Figure 5.4 Neighbourhood size preference on the Schaffer

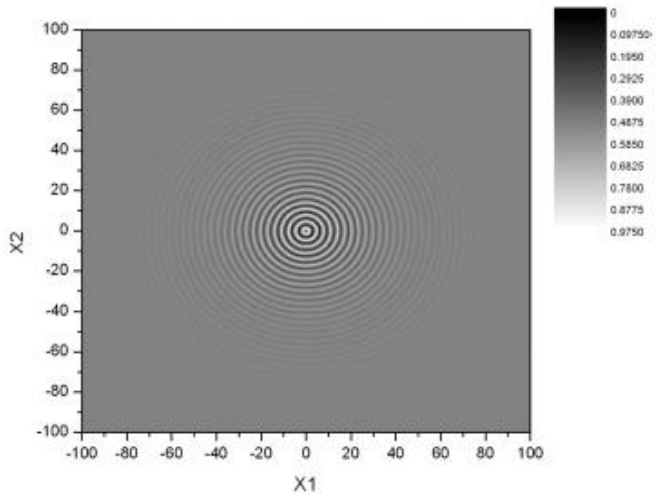


Figure 5.5 Contour of the Schaffer within [-100, 100] (Zhao et al., 2009)

runs was generated when the neighbourhood size was 0.00015, as in Figure 5.6. Moreover, Figure 5.7 shows the optimisation progress on the Rosenbrock. The improvement in the solution was remarkable when the neighbourhood size dropped from 0.0015 to 0.00015, which happened at 3360^{th} iteration. The solution was progressing until the 5000^{th} iteration.

The Sphere also benefited from neighbourhood reduction. Once the neighbourhood size dropped from 0.05 to 0.005, the bees found the global optimum easily. In fact, 96 runs reached the global optimum when the neighbourhood size was 0.005, which was one tenth of the original size (see Figure 5.8). It was well known that the Sphere is an easy function. Its smooth and parabolic shape encouraged the bees to find the global optimum quickly. However, when approaching the global optimum, the neighbourhood size should be small in order to avoid the overshooting problem. In this case, 0.005 was an optimal one. In fact, the bees found the global optimum straight away after the neighbourhood size dropped to 0.005 (see Figure 5.9).

The neighbourhood reduction was also useful when optimising the Ackley. 75 runs reached the global optimum when the neighbourhood size was 0.0007 (1/1000 of original neighbourhood size), while the other 25 trials benefited from 0.00007 of neighbourhood size (1/10000 of the original neighbourhood size). The distribution of neighbourhood size in Figure 5.10 indicated that the neighbourhood size (0.0007) was just right when approaching the global optimum on the Ackley.

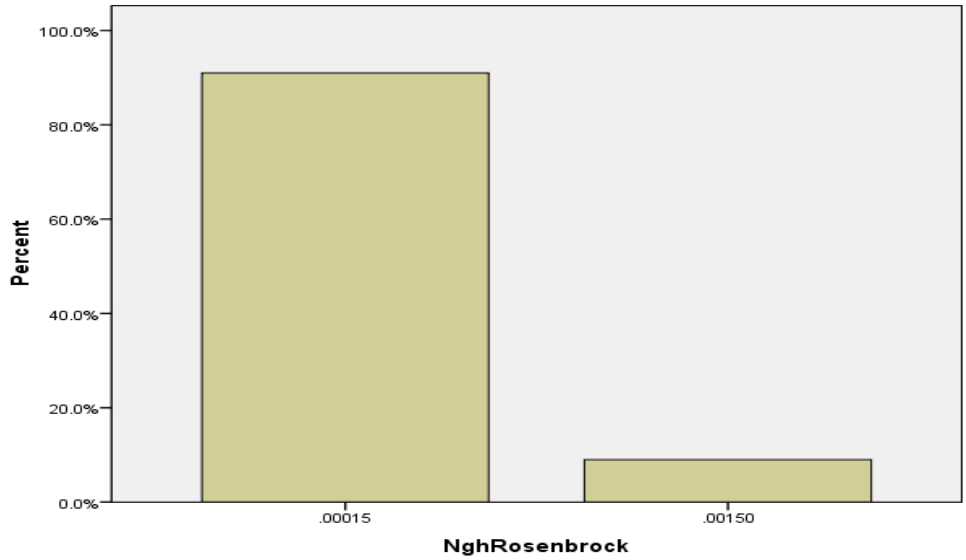


Figure 5.6 Neighbourhood size preference on the Rosenbrock

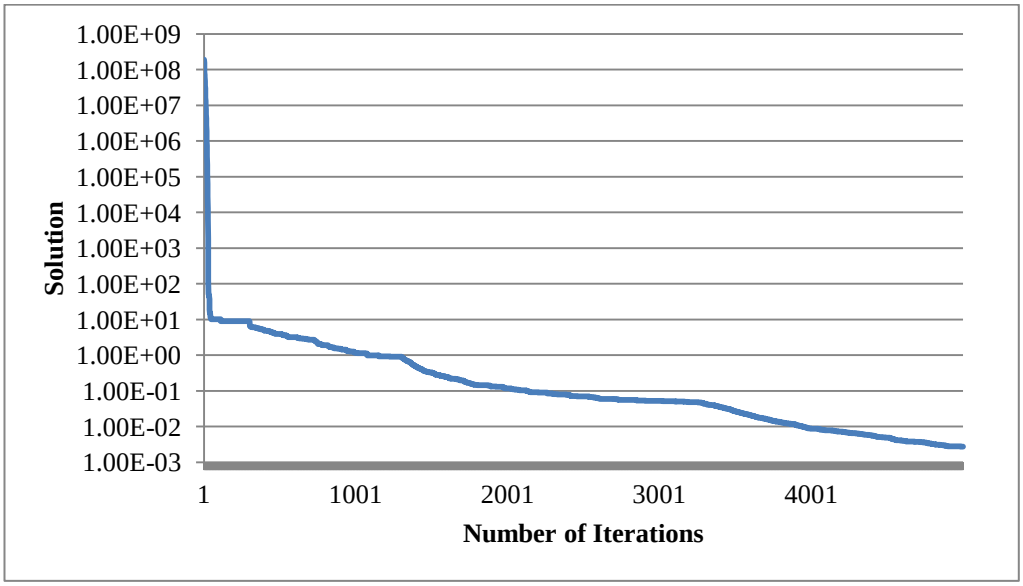


Figure 5.7 Optimisation progress on the Rosenbrock

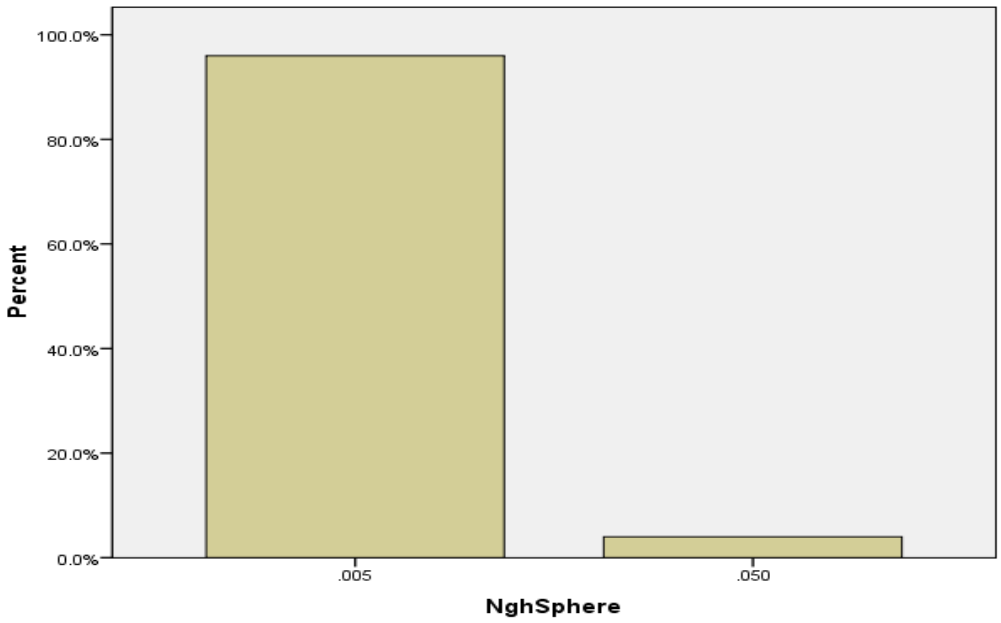


Figure 5.8 Neighbourhood size preference on the Sphere

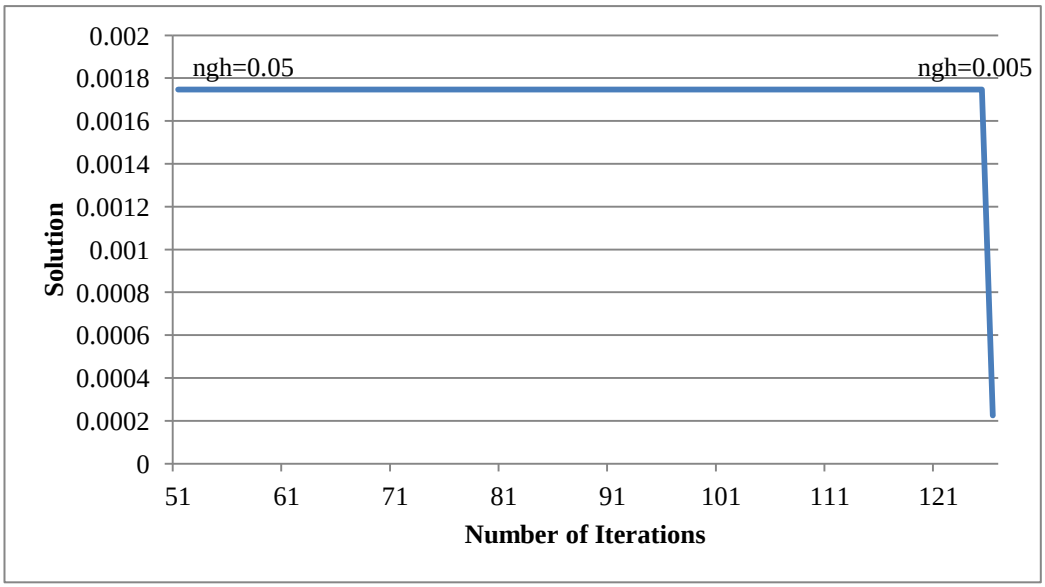


Figure 5.9 Optimisation progress on the Sphere

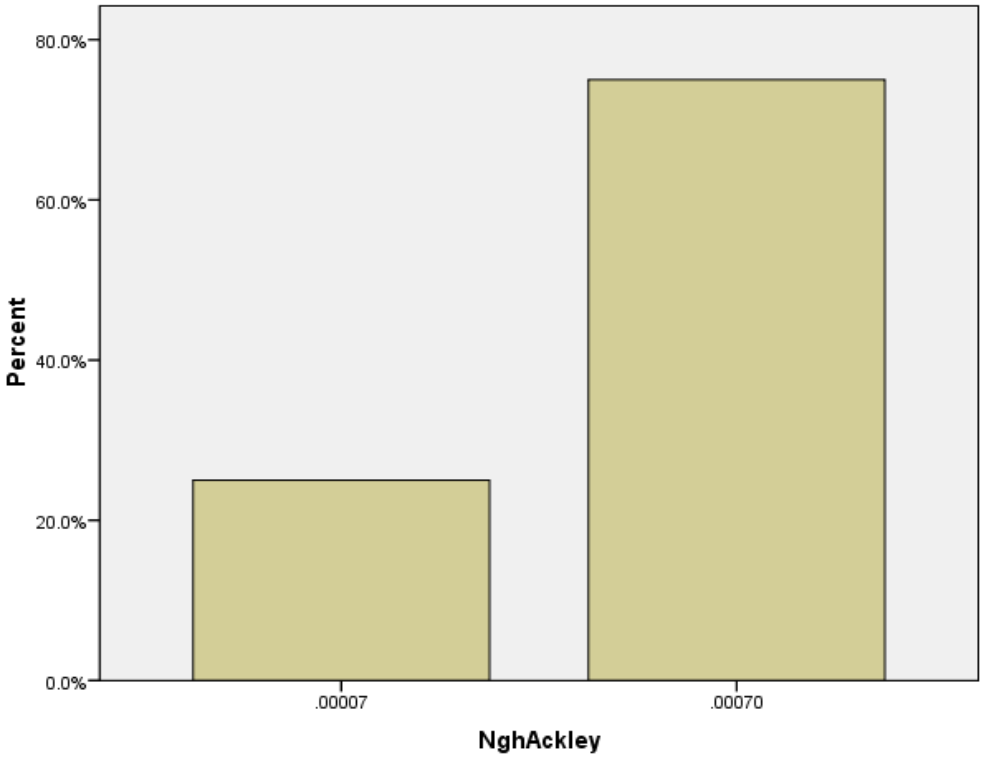


Figure 5.10 Neighbourhood size preference on the Ackley

There was however no significant difference between BA-NER and BA-NE when solving the Rastrigin. Based on a calculation, BA-NER only managed to give the best solution, which was 7.9597 when the bees were approximately 2.81 units from the global optimum. When getting closer than this distance, the bees most often failed to find the global optimum, regardless of the size of neighbourhood adopted; 0.01, 0.001 and 0.0001 (see Figure 5.11). In fact, the algorithm still failed even after completing one cycle. Hence, it could be said that the neighbourhood reduction failed to adapt with the problem landscape. Also, the small neighbourhood size was not able to guide the bees to escape from the local optima.

There was no difference in the performance of the algorithms on the Easom. In fact, 84 runs reached the global optimum at the original neighbourhood size, 0.5 (see Figure 5.12). This was because its flat surface would not bring the bees to any better solution. This unimproved solution forced the bees to use the old size and fixed search neighbourhood. Therefore, neither neighbourhood enlargement nor the reduction procedure would assist whilst the bees were searching for the solution on the flat surface. The fixed search neighbourhood was sufficient for the algorithm to optimise on this surface. Once the bees entered the hole, the procedure of neighbourhood enlargement and reduction might happen. Since the hole did not have local optima, the bees had no possibilities to get trapped at any point. Therefore, any size of neighbourhood could lead to an acceptable solution.

On the Griewank, the algorithm only managed to find the global optimum in four runs, with 0.015 of neighbourhood size as shown in Table 5.8. The progress on

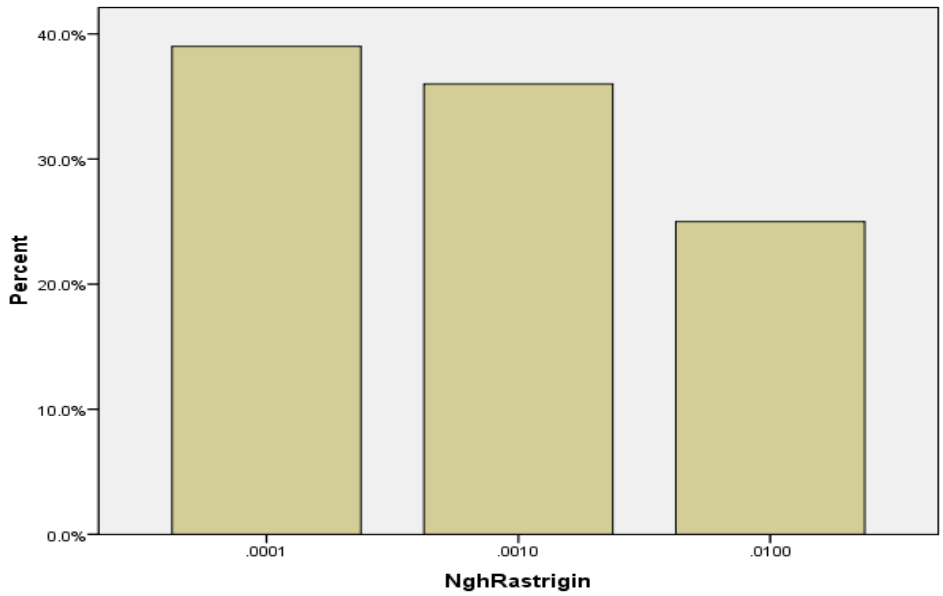


Figure 5.11 Neighbourhood size preference on the Rastrigin

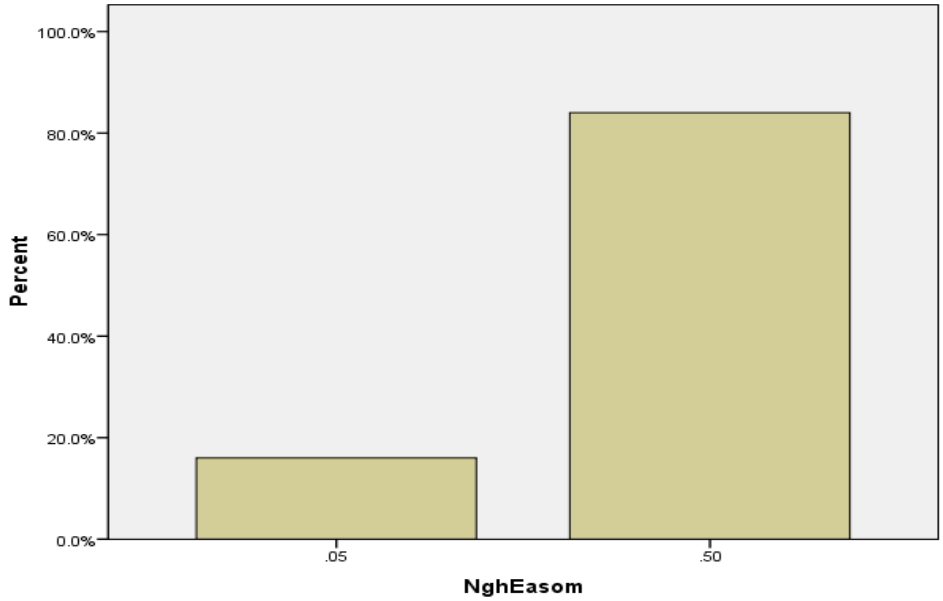


Figure 5.12 Neighbourhood size preference on the Easom

Table 5.8 Acceptable solutions found on the Griewank

i_{th} run	Number of iterations	Fitness	ngh
17	3014	0.00039	0.015
39	845	0.00074	0.015
55	515	0.00074	0.015
72	740	0.00033	0.015

the Griewank was too dramatic at an early stage but stagnated at the 1191st iteration (see Figure 5.13). At the 1191st iteration, the fitness was 0.06152 and this remained until the maximum iteration. In fact, the fitness did not change even after completing 127 cycles. This gave rise to the idea that the smaller neighbourhood size was not the only factor in improving algorithm performance when solving problems like the Griewank. To obtain a more acceptable solution within the allocated iteration, the amendment to ci should be made. Increasing the ci could give more time to the bees to search the neighbourhood and increase the chance of finding a better solution.

5.5. Applications

5.5.1. Single objective problem without constraints

BA-NER was tested on the gear train problem. The nr, ci and limit were obtained empirically and they were as in Table 5.9. The rest of the parameters and the stopping criterion were set as shown in Table 3.5.

The result obtained by BA-NER and comparison against UPSOm (Parsopoulos and Vrahatis 2005), ABC (Akay and Karaboga 2010), BA-NE and BA-AN assessment-ii and assessment-iv is presented in Table 5.10. Even though the result produced by BA-NER was not as good as the UPSOm and ABC, it was better than the ones obtained by the BA-NE and BA-AN assessment-ii and assessment-iv. The Mann Whitney test confirmed that BA-NER collectively outperformed the BA-NE and BA-ANs (see Table 5.11). This was caused by the smaller average solution and standard deviation generated by BA-NER. This analysis also

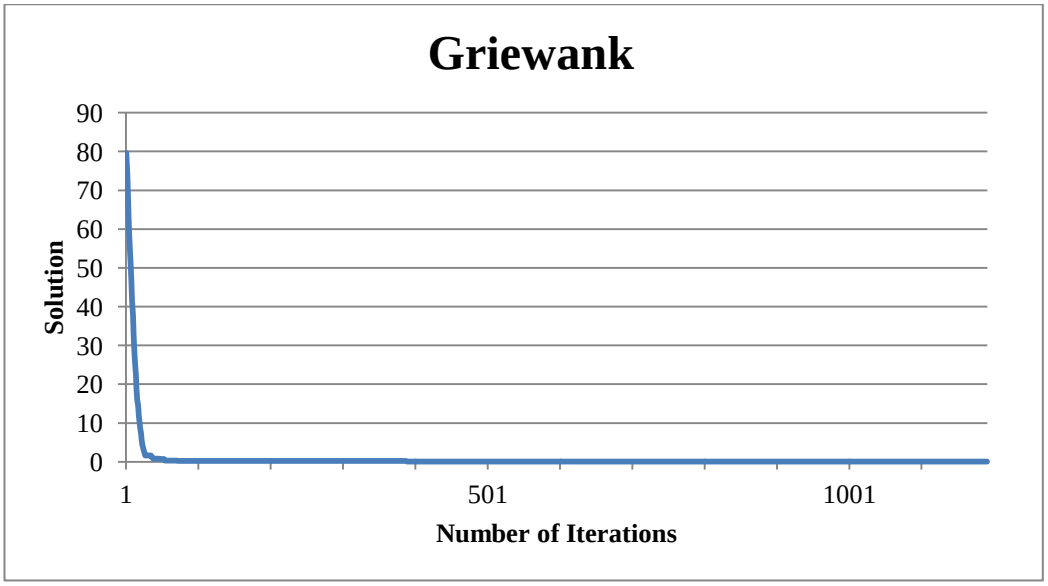


Figure 5.13 Optimisation progress on the Griewank

Table 5.9 The BA-NER parameters for gear train problem

Parameter	Value
Number of scout bees, n	5
Number of selected sites, m	2
Number of forager bees for each selected sites, nsp	1
Number of elite sites, e	1
Number of forager bees for each elite sites, nep	2
Neighbourhood size, ngh	2.0
nr	1/10
limit	0.2
ci	5

Table 5.10 Comparison against other algorithms on the gear train problem

	UPSOm	ABC	BA	BA-NE	BA-AN (assessment-ii)	BA-AN (assessment-iv)	BA-NER
Avg. Solution	3.80562E-08	3.641339E-10	6.84E-05	2.12E-06	4.53E-06	6.79E-06	3.61E-07
Std. Dev.	1.09631E-07	5.525811E-10	0.000366946	4.43477E-06	1.16133E-05	2.26966E-05	8.71333E-07
Best Solution	2.70E-12	2.70E-12	9.92E-10	1.55E-10	9.92E-10	1.31E-08	2.31E-11
x_1	NA	49	47	43	47	46	53
x_2	NA	16	12	13	26	12	13
x_3	NA	19	26	21	12	21	20
x_4	NA	43	46	44	46	38	34
Gear ratio	NA	0.144281	0.144311	0.144292	0.14431	0.14417	0.144284
% error	NA	0.001%	0.022%	0.009%	0.022%	-0.079%	0.003%

Table 5.11 Significance of the difference against the BA-NE and BA-ANs on the gear train problem

Methods	BA	BA-NE	BA-AN (assessment-ii)	BA-AN (assessment-iv)	BA-NER
BA	-	S	NS	NS	S
BA-NE	S	-	S	S	S
BA-AN (assessment-ii)	NS	S	-	NS	S
BA-AN (assessment-iv)	NS	S	NS	-	S
BA-NER	S	S	S	S	-

suggested that BA-NER was able to produce robust solutions.

5.5.2. Single objective problem with constraints

BA-NER was also tested on the four engineering design problems. Each problem had a set of constraints that must not be violated. It should be noted that BA-NER was also not designed to adapt to constrained problems. The same parameter setting as in Table 3.7 was employed on BA-NER, while nr, ci and limit were empirically tuned (see Table 5.12). As described in Section 3.3.2, for each problem, the algorithm was run 30 times, with 30,000 evaluations as the stopping criterion.

The experimental results produced by the BA-NER and other algorithms are shown in Table 5.13. In general, the average of the solutions produced by BA-NER was comparable to other algorithms. Table 5.14 suggests that BA-NER was not as good as BA-NE on the welded beam problem. However, BA-NER and BA-NE equally performed on the pressure vessel problem. Furthermore, these two algorithms outperformed the BA on this problem. The BA-NER also produced the best solution on the speed reducer problem. Meanwhile, the BA, BA-NE and BA-NER performed equally on the tension/ compression spring problem. The BA neither beat the BA-NE nor the BA-NER in all cases.

The neighbourhood size that was used to capture the final solution of each problem was recorded. Table 5.15 shows the significance of neighbourhood size

Table 5.12 Parameter setting for the engineering design problems

No.	Function	n	m	nsp	e	nep	ngh	nr	limit	ci
1	Welded Beam (4D)	10	5	2	2	4	0.08	1/10	0.008	1
2	Pressure Vessel (4D)	10	5	2	3	6	0.2	1/10	0.02	14
3	Tension/ Comp. Spring (3D)	6	5	5	1	8	0.001	1/10	0.0001	1
4	Speed Reducer (7D)	35	15	5	5	15	0.01	1/10	0.001	5

Table 5.13 Comparison against other algorithms on the engineering design problems

Problem	Stats.	SCA	PSO	(μ + λ)-ES	UPSOm	ABC	BA	BA-NE	BA-NER
Welded Beam	Best	NA	NA	1.724852	1.92199	1.724852	1.734783	1.731916	1.725877
	Mean	NA	NA	1.777692	2.83721	1.741913	1.768855	1.749546	1.796066
	Std. Dev.	NA	NA	0.088	0.680	0.031	0.040	0.013	0.045
	Evaluations	NA	NA	30000	100000	30000	30000	30000	30000
Pressure Vessel	Best	6171.00	6059.7143	6059.701610	6544.27	6059.714736	6289.745562	6283.130775	6234.788218
	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.308144	6853.349849	6749.722776	6711.8400
	Std. Dev.	NA	310	210	996	205	609	542	499
	Evaluations	20000	30000	30000	100000	30000	30000	30000	30000
Tension/ Com. Spring	Best	0.012669	0.012665	0.012689	0.0131200	0.012665	0.00988	0.00988	0.00988
	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	0.01036	0.01027	0.01036
	Std. Dev.	0.00059	0.000041	0.00039	0.0072	0.013	0.00048	0.00048	0.0004
	Evaluations	25167	15000	30000	100000	30000	30000	30000	30000
Speed Reducer	Best	2994.744241	NA	2996.348094	NA	2997.058412	2997.843904	2998.348453	2996.87953
	Mean	3001.758264	NA	2996.348094	NA	2997.058412	3005.295876	3003.358497	3002.371449
	Std. Dev.	4.0	NA	0	NA	0	3.2	3.1	2.9
	Evaluations	54456	NA	30000	NA	30000	30000	30000	30000

Table 5.14 Significance of the difference against other algorithms on engineering design problems

Problem	Algorithms	BA	BA-NE	BA-NER
Welded Beam Problem	BA	-	S	S
	BA-NE	S	-	S
	BA-NER	S	S	-
Pressure Vessel	BA	-	S	S
	BA-NE	S	-	NS
	BA-NER	S	NS	-
Tension/ Com. Spring	BA	-	NS	NS
	BA-NE	NS	-	NS
	BA-NER	NS	NS	-
Speed Reducer	BA	-	NS	S
	BA-NE	NS	-	S
	BA-NER	S	S	-

Table 5.15 Distribution of neighbourhood size corresponds to the number of runs

Problem	ngh	Number of runs	Avg. solution	Std. Dev.	Significance
Welded Beam	0.08	16	1.79447	0.049181	NS (0.803)
	0.008	14	1.79795	0.041441	
Pressure Vessel	0.2	10	6796.98333	635.645625	NS (0.93)
	0.02	20	6626.69667	464.9944074	
Tension/ Compression Spring	0.001	9	0.010212194	0.000307181	NS (0.541)
	0.0001	21	0.010309777	0.000386236	
Speed Reducer	0.01	7	3004.337143	1.913092734	NS (0.066)
	0.001	23	3002.154348	3.038526596	

corresponding to the number of runs. It could be said that the distribution of neighbourhood size was not significant on all problems since the significance value given by the Mann Whitney test was more than 0.05. This analysis suggested that the procedure of neighbourhood reduction on the BA-NER when solving these four design problems was not necessary.

However, on the speed reducer problem, even though the difference was not significant, the value given by the Mann Whitney test was close to 0.05, which was 0.066. Furthermore, 23 (77%) runs ended up with neighbourhood size (0.001) (see Figure 5.14). In fact, for the best solution (2996.87953), the solution stagnated at 3000.75 for as long as 13775 evaluations. It was assumed that at this stage, the bees were overshooting the global optimum. Once the neighbourhood size dropped to 0.001, the improvement was remarkable, which was turning to 2997.84. This suggested that neighbourhood size (0.001) was suitable to avoid the overshooting problem. Figure 5.15 shows the optimisation progress of the BA, BA-NE and BA-NER on the speed reducer problem.

This experimental result supported the idea that the combination of neighbourhood enlargement and reduction was beneficial to a problem that suffers from stagnation. Also, it should be emphasised that the BA did not outperform the BA-NE and BA-NER in any case. Table 5.16 (a)-(d) provides the values of design variables (x_i) and constraints (g_i) of the welded beam, pressure vessel, tension/ compression spring and speed reducer obtained by the BA-NER.

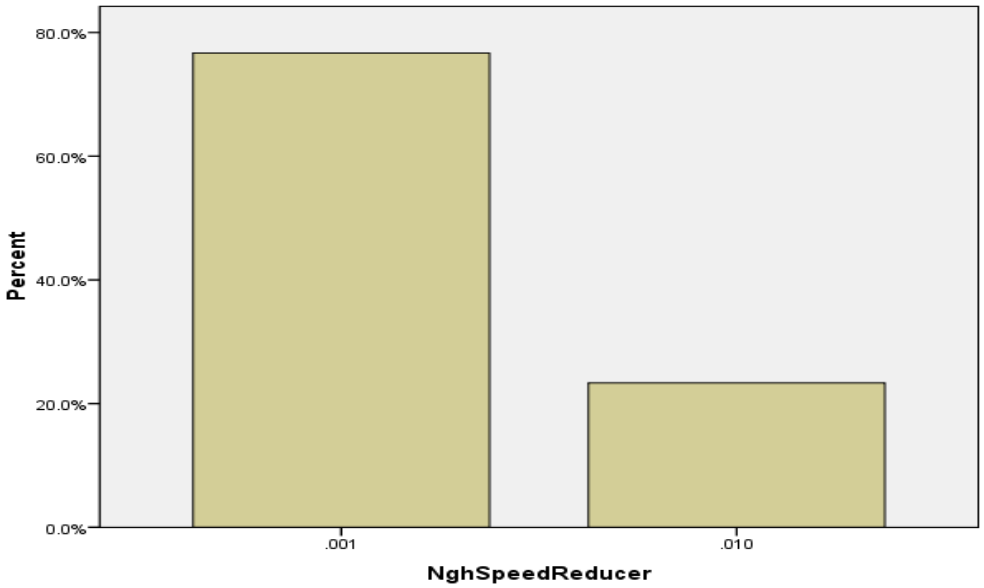


Figure 5.14 Neighbourhood size preferences on the speed reducer problem

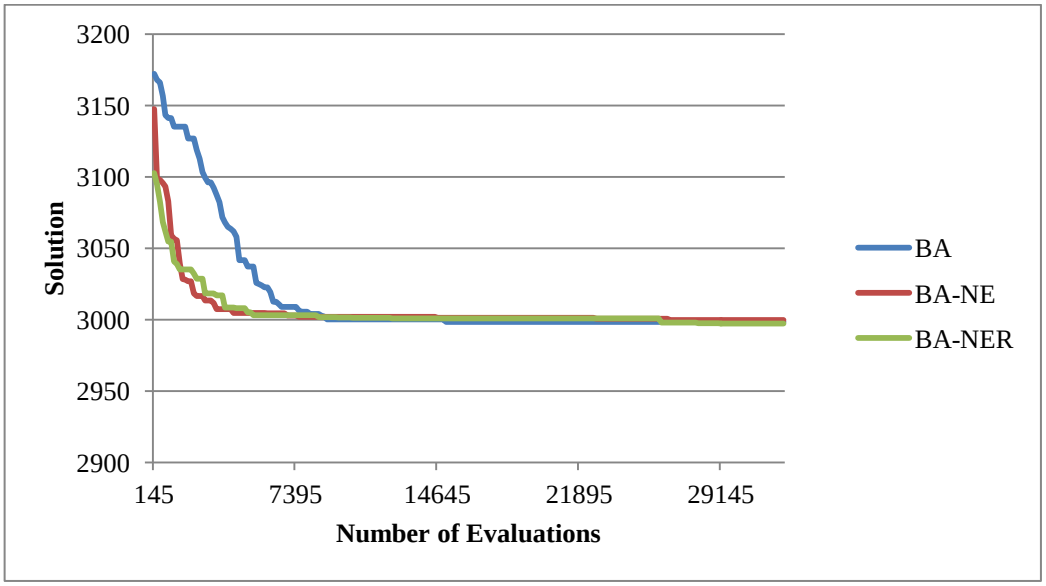


Figure 5.15 Optimisation progress of the algorithms on the speed reducer problem

Table 5.16 The x_i and g_i values of the best solution obtained by the BA-NER on the (a) welded beam, (b) pressure vessel, (c) tension/ compression spring and (d) speed reducer problem

(a)

Variables and Constraints	Values
x_1	0.205281
x_2	3.48091
x_3	9.03491
x_4	0.20581
g_1	-0.23993
g_2	-0.33519
g_3	-0.00053
g_4	-3.43176
g_5	-0.08028
g_6	-0.23554
g_7	-6.28462

(b)

Variables and Constraints	Values
x_1	0.798587
x_2	0.395051
x_3	41.375
x_4	199.643
g_1	-5E-05
g_2	-0.00033
g_3	-2.14E+08
g_4	-40.357

(c)

Variables and Constraints	Values
x_1	0.050061
x_2	0.375824
x_3	8.49478
g_1	-0.00017
g_2	-6.5E-05
g_3	-4.86003
g_4	-0.71608

(d)

Variables and Constraints	Values
x_1	3.50001
x_2	0.700002
x_3	17
x_4	7.34552
x_5	7.80322
x_6	3.35033
x_7	5.28671
g_1	-0.07392322
g_2	-0.1980054
g_3	-0.48981651
g_4	-0.901351904
g_5	-2.7E-05
g_6	-1.5E-05
g_7	-0.7024992
g_8	0.000000
g_9	-0.583333333
g_{10}	-0.057181112
g_{11}	-0.011256763

5.5.3. Multi objective problem with constraints

The BA-NER was also tested on a multi-objective problem with a number of constraints. In a multi-objective problem, every objective has the same priority. It means that there was no objective that is superior to any other and optimisation on all objectives should be dealt with simultaneously. Since all objectives held the same priority and solved at the same time, it was not possible to sort and rank the solutions as practiced in a single objective problem. Because of this, the elite bees (e) were no longer employed.

A solution in a multi-objective problem is known as a Pareto optimal if and only if there is no other solution that dominates it (Lee 2010). Specifically, a solution (x_1) was said to dominate the other solution (x_2) if these both conditions were met (Li et al., 2008):

1. The solution x_1 is no worse than x_2 in all objectives
2. The solution x_1 is strictly better than x_2 in at least one objective

For a multi-objective problem with constraints, a multiple-disk clutch brake problem was chosen (Deb and Srinivasan 2006; Osyczka 2002). The problem objectives were to minimise the brake mass (kg) and the stopping time (s), where all design variables were discrete. The details of the problem including the diagram, formulation and constraints are provided in Appendix C.

5.5.3.1. Experimental setup

The following experiment was to compare the performance of the BA-NE and BA-NER. Before performing simultaneous optimisation on the problem, the

algorithms solved the two objectives consecutively. First, the algorithms were to minimise the brake mass. The design variables that were associated with the best solution of the brake mass were then substituted in the function of stopping time. Afterwards, the algorithms minimised the stopping time. The design variables that resulted in a minimum stopping time were substituted in the brake mass formulation. The points of minimum brake mass and stopping time were then plotted on a graph and a straight line that connected these points was sketched. This procedure was to provide a boundary line to a Pareto optimal. The solutions that lay above the line were considered as poor, while those below the line were considered promising ones.

Since there is no sorting and ranking procedure of the solutions obtained in a multi-objective problem, there would be no global search performed. Optimising without global search also meant that the improvement was solely relying on the neighbourhood search. Neighbourhood search was performed by all the bees (solutions) that were generated at the initialisation stage. Each bee, followed by the same number of forager bees, returned to discovered flower patches for neighbourhood search.

The parameter setting for the BA-NE was obtained by a trial and error procedure. The parameter setting that produced the best solution of the BA-NE was then used by the BA-NER. With this parameter setting, a number of experiments were run on the BA-NER to obtain the best nr, ci and limit. The best parameter configuration on the BA-NER was captured and shown in Table 5.17. With these

Table 5.17 Parameter settings for the BA-NE and BA-NER

Parameter	Single objective	Multi-objective
n	10	10
m	8	10
nsp	3	2
e	5	0
nep	5	0
ngh	1.0	1.0
nr	1/10	1/10
limit	0.001	0.001
ci	3	5

parameter settings, the BA-NE and BA-NER were run 30 times and each run was terminated when the number of iterations reached 1000.

5.5.3.2. Experimental results

The best solution of each objective that was obtained by the BA-NE and BA-NER was as in Table 5.18 (a) and (b). With the minimum brake mass as the objective, both algorithms produced the same brake mass and stopping time, which were 0.470485kg and 14.83092s, respectively. When minimising the stopping time, the BA-NE produced shorter time, which was 3.805228s. However, after substituting the design variables into the brake mass formulation, BA-NER generated a lighter brake mass, which was 1.845918kg.

The points of minimum brake mass and stopping time that were obtained by the BA-NE and BA-NER were plotted on the graph (■) and a line that connected these two points was sketched (see Figure 5.16 (a) and (b)). With simultaneous optimisation on the objectives, the Pareto optimal solution that had been obtained by the BA-NE and BA-NER were tabulated on the same graph (■). In terms of proximity, BA-NER produced more solutions that were closer to this line, whereas solutions obtained by BA-NE were more scattered away from the line. The solutions that were farther from the line were marked with (/) sign.

To make the comparison clearer, the experiment was repeated 100 times. Figure 5.17 (a) and (b) shows the distribution of the Pareto optimals. Even though it was not easy to compare proximity to the boundary line, the solutions obtained by BA-NER were more clustered along the line compared to the BA-NE. This analysis implied that the BA-NER was able to find more Pareto optimals solutions that

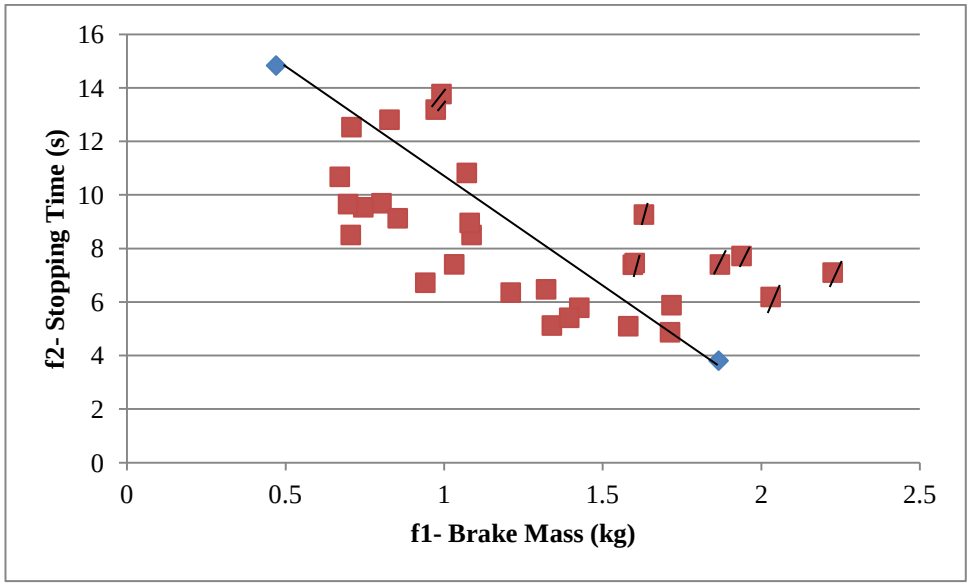
Table 5.18 The best solution of (a) brake mass and (b) stopping time

(a)

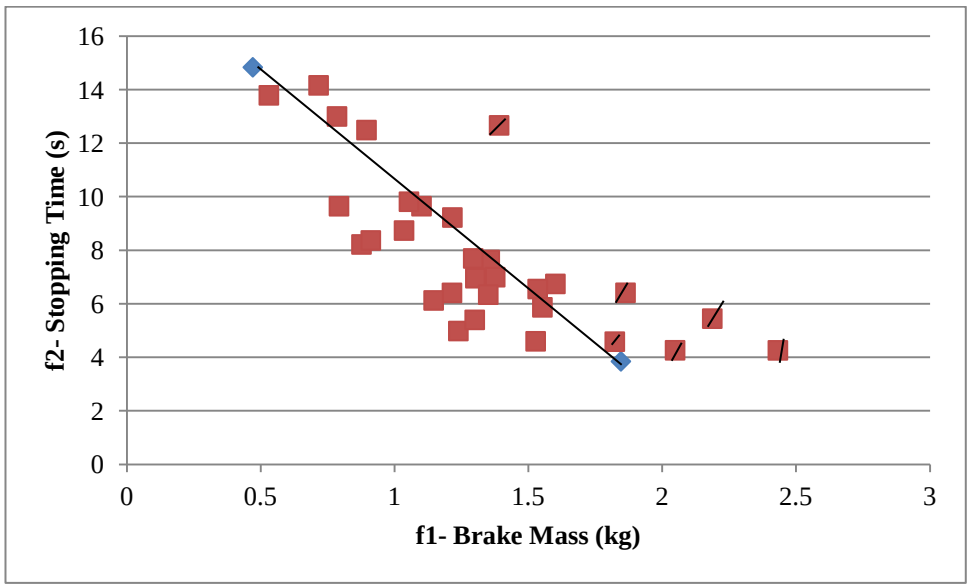
	BA-NE	BA-NER
Brake Mass, f1	0.470485	0.470485
x_1	70	70
x_2	90	90
x_3	1.5	1.5
x_4	780	780
x_5	3	3
Stopping Time, f2 (by substitution)	14.83092	14.83092

(b)

	BA-NE	BA-NER
Stopping Time, f2	3.805228	3.845122
x_1	79	78
x_2	109	108
x_3	1.5	1.5
x_4	990	990
x_5	8	8
Brake Mass, f1 (by substitution)	1.865767	1.845918

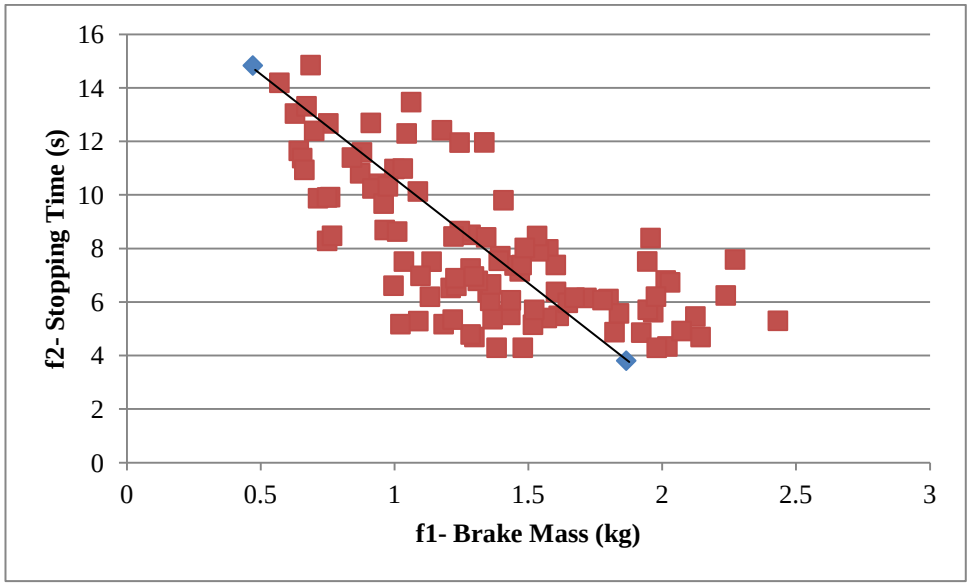


(a)

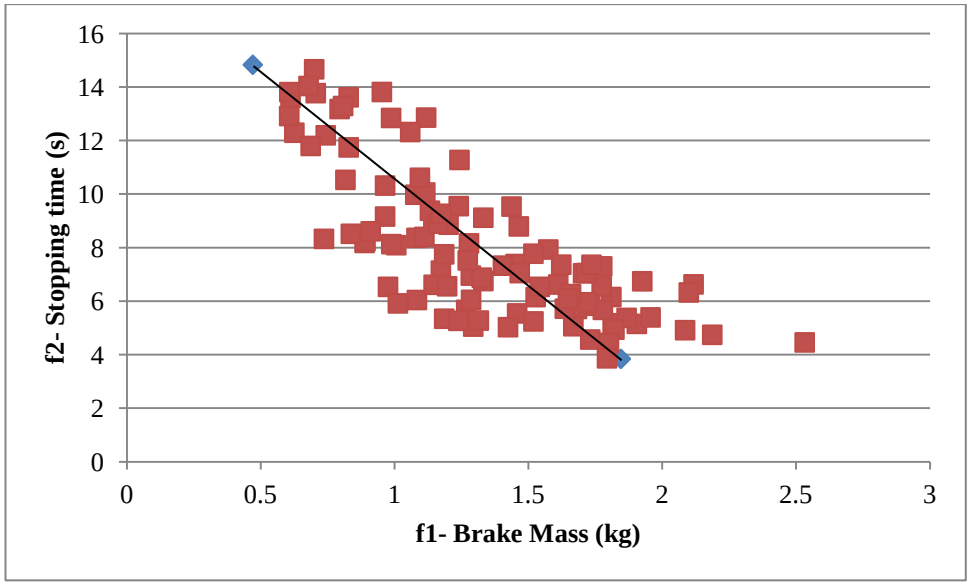


(b)

Figure 5.16 Proximity of 30 Pareto optimals to the boundary line obtained by the (a) BA-NE and (b) BA-NER



(a)



(b)

Figure 5.17 Proximity of 100 Pareto optimals to the boundary line obtained by the (a) BA-NE and (b) BA-NER

were closer to the boundary line compared to the BA-NE.

5.6. Summary

Neighbourhood size as used in the early works contributed to overshooting the global optimum in four mathematical benchmarks. A calibration procedure was performed to identify the most appropriate neighbourhood size when approaching the global optimum. The analysis from this calibration procedure supported the view that the neighbourhood size should be small when approaching the global optimum to avoid the overshooting problem. Alongside neighbourhood reduction, the adaptive enlargement of the search neighbourhood was also employed to help the bees to escape from local optima.

The BA-NER significantly produced better solutions when solving multi-pocketed problems. However, the role of adaptive enlargement and reduction of the search neighbourhood was not necessary when solving easy and smooth test functions. Experimental results also showed that the BA-NER had advantages in finding better solutions on the gear train problem. Furthermore, the BA-NER generated the best solution on one of the four engineering design problems. This new approach was also able to reach more promising solutions in a multi-objective problem.

Chapter 6

CONCLUSION

This chapter summarises the contributions and conclusions of this study. It also provides suggestions for further research.

6.1. Contributions

First, this work investigated the types of neighbourhood in the Bees Algorithm, rather than proving the superiority of one algorithm over the others. In addition, this research found that the modified algorithms were robust, with respect to various criteria, as follows:

1. The performance of the modified algorithms was never worse than that of the original BA, even after parameter tuning was carried out on the latter algorithm. For that reason, it could be said that the good performance of the modified algorithms was not caused by their own parameter selection. Implicitly, this procedure proved that the proposed methods were not parameter-dependent.
2. The statistical difference obtained by the Student's t-test and Mann Whitney test showed that the solutions obtained by the modified algorithms were better than the BA, regardless of the random initialisation.

Second, this research proved that an adjustment to the neighbourhood size influenced the performance of the proposed algorithms. However, for some problems, the amendment to the neighbourhood size only was not sufficient to improve the solutions.

Third, the proposed algorithms were tested on well-known mathematical benchmarks which each exhibited different characteristics. Also, the algorithms were tested on selected engineering design problems. The problems were single objective with and without constraints, and multi-objective with constraints.

Fourth, this work supported the No Free Lunch Theorem by showing that the modified algorithms did not always outperform the original algorithm in solving all classes of problems, without some efforts being spent on other parameter(s). For instance, a good performance of the BA-NER came only after adopting neighbourhood reduction.

Finally, this work attempted to solve the problem faced by the algorithm by using the TRIZ approach.

6.2. Conclusions

In conclusion, all objectives stated in Chapter 1 have been met.

Adaptive enlargement of the search neighbourhood in the BA was developed (**Objective 1**). This method gave better performance when compared to that of the standard BA. The problem landscape highly influenced the performance of the

proposed algorithm. A smooth surface encouraged bees to converge quickly to the global optimum, while a multi-pocketed surface might cause difficulties to the bees. The adaptive enlargement of the search neighbourhood was unable to help the bees to improve the solution in the latter type of problem and thus caused stagnation.

An asymmetrical search neighbourhood, which was derived from a TRIZ inventive solution, was proposed. This kind of search neighbourhood was introduced as opposed to a symmetrical search neighbourhood, which was normally practiced in the BA. Four different types of asymmetrical search neighbourhood were analysed. Experimental results suggested that an amount of neighbourhood size should be located at the both sides of the current best solution in order to improve the solutions quickly. In addition, the evaluation of the solution on one side of the current best solution prior to the other side and vice versa simply produced a similar outcome. Also, the analysis confirmed that under a certain measurement, the asymmetrical search neighbourhood did not give a positive influence to the proposed algorithm (**Objective 2**).

The calibration procedure revealed that the neighbourhood size used in early works contributed to overshoot the global optimum. The algorithm with adaptive enlargement and reduction of the search neighbourhood significantly produced better solutions when solving multimodal problems that had a noisy landscape (**Objective 3**). The small size of the neighbourhood, which was caused by the neighbourhood reduction procedure, helped the bees to avoid the overshooting problem. In addition, the large neighbourhood size that resulted from no

improvement made after a number of iterations assisted the bees to get out from local optima. Despite this remarkable improvement, adaptive enlargement and reduction of the search neighbourhood was not necessary for the algorithm when solving easy optimisation problems.

6.3. Further research

First, this work investigated the effect of asymmetrical search neighbourhood on the algorithm. This kind of search neighbourhood was derived from a TRIZ inventive solution and aimed to reduce the noisy surface. Under a certain measurement, the asymmetrical search neighbourhood was not capable of accomplishing optimisation on multi-pocketed problems. Hence, it is worth studying asymmetrical search neighbourhoods with different measurements. Furthermore, there were seven inventive solutions listed in solving the problem. However, in this work, only one solution was considered. There is plenty of room to explore the other six solutions in respect to the current condition of the BA. Also, researchers might be interested to formulate the problem in different ways. As a result, different mapping on the TRIZ matrix could be done and other solutions could be derived. Besides, researchers might want to improve the BA in terms of ease of use, automation, etc. Improving these features might worsen other features in the BA. This contradiction might be solved by applying other TRIZ inventive solutions.

Second, the modified algorithms were tested on selected engineering design problems, with and without constraints. For constrained optimisation problems, it is worth studying how to handle the constraints. Good constraints handling would reduce the time consumed and thus high computation costs could be avoided.

Third, this study found that the combination of adaptive enlargement and reduction in the search neighbourhood positively affected the algorithm, especially when solving problems with multi-pocketed surface. In the proposed algorithm, three new parameters were introduced, in addition to the six parameters of the BA. Since this work was to study the effects of various neighbourhoods, no effort has been made to reduce the number of parameters. Hence, it is suggested to develop a similar mechanism of neighbourhood enlargement and reduction, but with fewer parameters.

Finally, this work focused entirely on the search neighbourhood of the algorithms. The algorithms which were developed based on the current situation of the BA exposed other elements that are worth studying. The elements are initialisation, randomisation, recruitment and global search.

Appendix A

The schematic compound gear train that is to be designed is as Figure A.1.

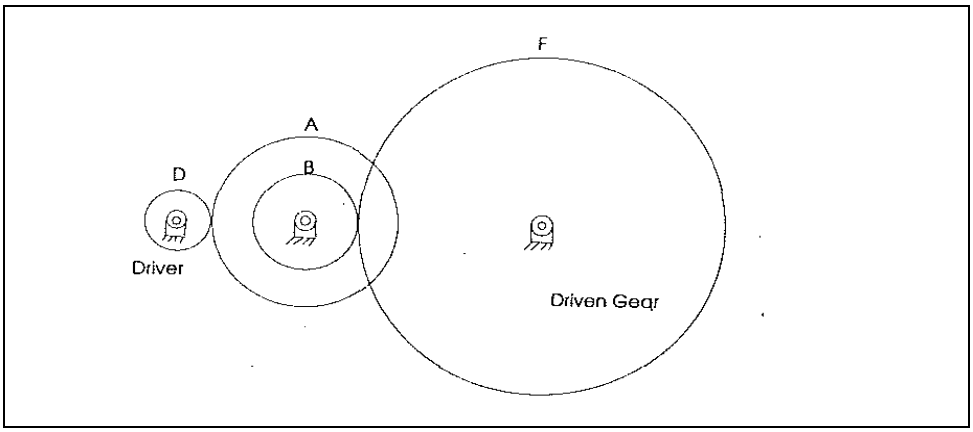


Figure A.1 Schematic of a gear train

The gear ratio is defined as:

Gear ratio = $\frac{T_d T_b}{T_a T_f}$

and it should be as close as possible to 1/ 6.931. Subsequently, the objective problem is written as below:

Min f (x) =$\left(\frac{1}{6.931} - \frac{T_d T_b}{T_a T_f}\right)^2 = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2$,

Subject to $12 \leq x_i \leq 60$, i=1,…4 and x_i are all integers.

Appendix B

Problem1: Minimisation of the cost of a welded beam

The problem is to design a welded beam for a minimum fabrication cost, subject to constraints on shear stress(τ), bending stress in the beam (σ), buckling load on the bar (Pc) and end deflection of the beam (δ). There are four design variables x_1, x_2, x_3 and x_4 corresponding to h, l, t and b, respectively. Figure B.1 shows the schematic of a welded beam.

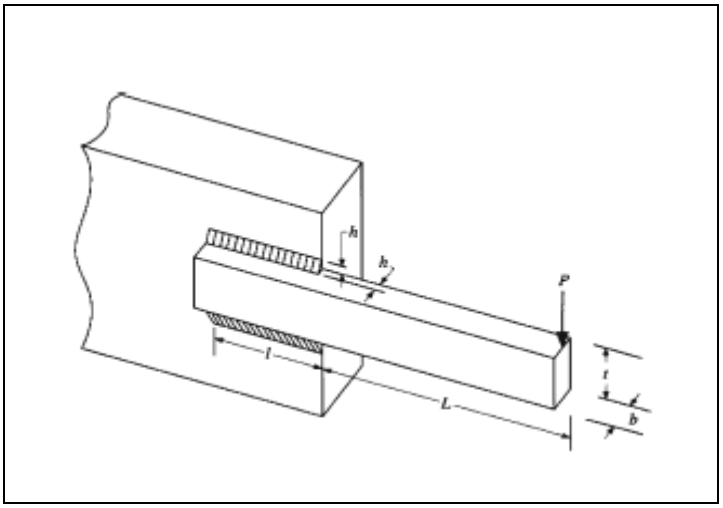


Figure B.1 Welded Beam

The problem can be expressed as follows:

Min f(x) =$1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

Subject to:

$g_1(x) = \tau(\overrightarrow{x}) - 13{,}600 \leq 0$

$g_2(x) = \sigma\overrightarrow{(x)} - 30{,}000 \leq 0$

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

$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5.0 \leq 0$

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

$g_6(x) = \delta\overrightarrow{(x)} - 0.25 \leq 0$

$g_7(x) = 6000 - Pc\overrightarrow{(x)} \leq 0$

$$\tau\overrightarrow{(x)} = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{6000}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = 6000\left(14 + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma\overrightarrow{(x)} = \frac{504000}{x_4x_3^2}$$

$$\delta\overrightarrow{(x)} = \frac{65{,}856{,}000}{30x10^6x_4x_3^3}$$

$$Pc(\overrightarrow{x}) = \frac{4.013(30 \text{ x } 10^6)\sqrt{\frac{x_3^2 x_4^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30 \text{ x } 10^6}{48 \text{ x } 10^6}}}{28}\right)$$

$0.1 \leq x_1 \leq 2.0$

$0.1 \leq x_2 \leq 10$

$0.1 \leq x_3 \leq 10$

$0.1 \leq x_4 \leq 2.0$

Problem2: Minimisation of the total cost of designing a pressure vessel

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure B.2. The objective is to minimise the total cost, including the cost of the material, forming and welding. The design variables are: thickness x_1, thickness of the head x_2, the inner radius x_3, and the length of the cylindrical section of the vessel x_4. The variables x_1 and x_2 are integer multiples of 0.0625 inch.

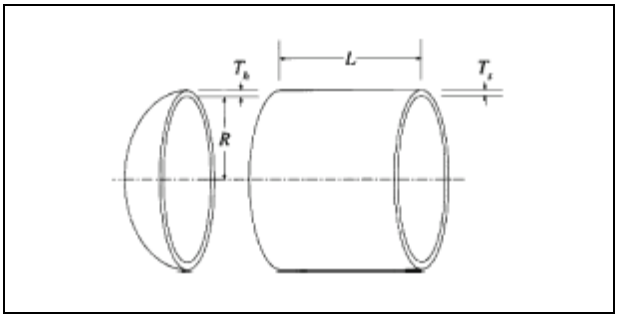


Figure B.2 Pressure Vessel

The problem can be stated as follows:

Min f(x) = $0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

Subject to:

$g_1(x) = -x_1 + 0.0193x_3 \leq 0$

$g_2(x) = -x_{2+}0.00954x_3 \leq 0$

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

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

$1 \leq x_1 \leq 99$

$1 \leq x_2 \leq 99$

$10 \leq x_3 \leq 200$

$10 \leq x_4 \leq 200$

Problem3: Minimisation of the weight of a tension/ compression spring

This problem is to minimise the weight of the tension/ compression spring, subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. The schematic of a tension/ compression spring is shown in Figure B.3.

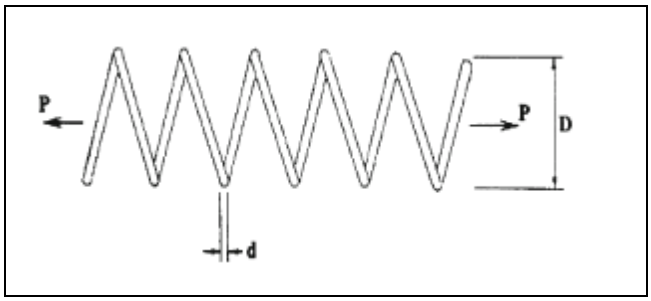


Figure B.3 Compression/ Tension Spring

The objective function can be stated as:

Min f(x) = $(x_3 + 2)x_2x_1^2$

Subject to:

$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$

$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0$

$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$

$g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$

$0.05 \leq x_1 \leq 2.0$

$0.25 \leq x_2 \leq 1.3$

$2.0 \leq x_3 \leq 15$

Problem4: Minimisation of the weight of a speed reducer

The weight of the speed reducer is to be minimised subject to constraints on bending stress of the gear teeth, surfaces stress, transverse deflections of the shafts and stresses in the shafts. The schematic of a speed reducer is shown in Figure B.4. The variables $x_1,x_2,\ldots x_7$ are the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings and the diameter of the first and second shafts. All variables are continuous, except x_3 that is integer.

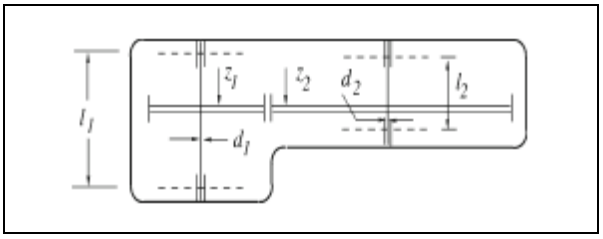


Figure B.4 Speed Reducer

The problem can be expressed as follows:

Min f(x) $=0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 +$ x72+7.4777x63+x73+0.7854(x4x62+x5x7 2)

Subject to:

$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$

$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$

$g_3(x) = \frac{1.93x_4^3}{x_1x_3x_6^4} - 1 \leq 0$

$$g_4(x) = \frac{1.93x_5^3}{x_1x_3x_7^4} - 1 \leq 0$$

$$g_5(x) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16900000} - 1 \leq 0$$

$$g_6(x) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157500000} - 1 \leq 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0$$

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

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

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

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

$$2.6 \leq x_1 \leq 3.6$$

$$0.7 \leq x_2 \leq 0.8$$

$$17 \leq x_3 \leq 28$$

$$7.3 \leq x_4 \leq 8.3$$

$$7.8 \leq x_5 \leq 8.3$$

$$2.9 \leq x_6 \leq 3.9$$

$$5.0 \leq x_7 \leq 5.5$$

Appendix C

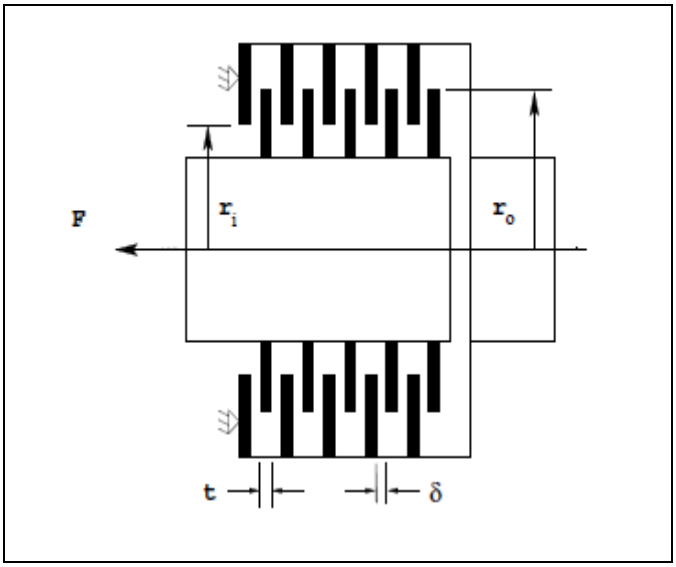


Figure C.1 Multiple-disk clutch brake

Figure C.1 is the schematic of a multiple-disk clutch brake design. Two conflicting objectives are considered:

(i) Minimisation of mass (f_1) of the brake system in kilogram, formulated as follows:

$$\text{Min } f_1(\vec{x}) = \pi({x_2}^2 - {x_1}^2)x_3\,(x_5 + 1)\,\rho$$

(ii) Minimisation of stopping time (f_2) in seconds, formulated as follows:

$$\text{Min } f_2(\vec{x}) = T = \frac{I_z\,\omega}{M_h + M_f}$$

There are five design variables to be considered: $\vec{x} = (r_i, r_o, t, F, Z)$, where $r_i = (60,61,62, \ldots, 78, 79, 80)\text{mm}$

$r_o = (90,91,92, \dots, 108, 1099, 110)\text{mm}$

$t = (1, 1.5, 2, 2.5, 3)\text{mm}$

$F = (600, 610, 620, \dots, 980, 990, 1000)\text{mm}$

$Z = (2, 3, 4, 5, 6, 7, 8, 9, 10)\text{mm}$

Subject to:

$g_1(\vec{x}) = x_2 - x_1 - \Delta R \geq 0$

$g_2(\vec{x}) = L_{max} - (x_5 + 1)(x_3 + \delta) \geq 0$

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

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

$g_5(\vec{x}) = V_{sr,max} - V_{sr} \geq 0$

$g_6(\vec{x}) = M_h - sM_s \geq 0$

$g_7(\vec{x}) = T \geq 0$

$g_8(\vec{x}) = T_{max} - T \geq 0$

$r_{i,min} \leq x_1 \leq r_{i,max}$

$r_{o,min} \leq x_2 \leq r_{o,max}$

$t_{min} \leq x_3 \leq t_{max}$

$0 \leq x_4 \leq F_{max}$

$2 \leq x_5 \leq Z_{max}$

Following are the parameters:

$$M_h = \frac{2}{3}\mu x_4 x_5 \left(\frac{{x_2}^3 - {x_1}^3}{{x_2}^2 - {x_1}^2}\right) \text{N.mm}$$

$$V_{sr} = \frac{\pi R_{sr} n}{30} \text{ mm/s}$$

$\mu = 0.5$

$s = 1.5$

$M_f = 3\ \text{N.m}$

$r_{i,max} = 80\ \text{mm}$

$t_{max} = 3\ \text{mm}$

$\omega = \pi n/30\ \text{rad/s}$

$R_{sr} = \frac{2}{3}\left(\frac{x_2{}^3 - x_1{}^3}{x_2{}^2 - x_1{}^2}\right)\ \text{mm}$

$p_{max} = 1\text{MPa}$

$T_{max} = 15\ \text{s}$

$I_z = 55\ \text{kg.m}^2$

$r_{o,min} = 90\ \text{mm}$

$F_{max} = 1000\ \text{N}$

$A = \pi(x_2{}^2 - x_1{}^2)\text{mm}^2$

$\Delta R = 20\ \text{mm}$

$\rho = 0.0000078\ \text{kg/mm}^3$

$n = 250\ \text{rpm}$

$\delta = 0.5\ \text{mm}$

$r_{o,min} = 110\ \text{mm}$

$Z_{max} = 9$

$p_{rz} = \frac{x_4}{A}\ \text{N/mm}^2$

$L_{max} = 30\ \text{mm}$

$V_{sr,max} = 10\ \text{m/s}$

$M_s = 40\ \text{Nm}$

$r_{i,min} = 60\ \text{mm}$

$t_{min} = 1.5\ \text{mm}$

Appendix D

Table D.1 Summary of the experimental results of the three types of search neighbourhood in comparison against the Bees Algorithm

Types of search neighbourhood \ Problems		Goldstein & Price (2D)	Schwefel (2D)	Schaffer (2D)	Rosenbrock (10D)	Sphere (10D)	Ackley (10D)	Rastrigin (10D)	Martin & Gaddy (2D)	Easom (2D)	Griewank (10D)	Gear Train (4D)	Welded Beam (4D)	Pressure Vessel (4D)	Ten/ Compression Spring (3D)	Speed Reducer (7D)
Adaptive enlargement		B			B		B		B		B	B	B	B		
Asymmetrical	Assessment-i	not analysed											not tested			
	Assessment-ii				B	W							not tested			
	Assessment-iii	not analysed											not tested			
	Assessment-iv				B	W							not tested			
Adaptive enlargement & reduction		B		B	B	B	B		B		B	B	B	B		B

B- Better, W- Worse, Blank Cell- Similar performance

REFERENCES

Abbass, H. A. 2001. *MBO: Marriage in honey bees optimization a haplometrosis polygynous swarming approach*. Congress on Evolutionary Computation 2001, Seoul, South Korea, 27-30 May 2001, pp. 207-214.

Adorio, E. P. 2005. *MVF- Multivariate Test Functions Library in C for Unconstrained Global Optimization* [Online]. Available at: http://geocities.com/eadorio/mvf.pdf [Accessed: 30 June 2010].

Akay, B. and Karaboga, D. 2010. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, pp. 1-14.

Ali, M. M. and Kaelo, P. 2008. Improved particle swarm algorithms for global optimization. *Applied Mathematics and Computation* 196(2), pp. 578-593.

Altshuller, G. 2001. *And Suddenly the Inventor Appeared. TRIZ, the Theory of Inventive Problem Solving*. Worcester, Massachusetts: Technical Innovation Center, Inc.

Anderson, C. and Ratnieks, F. L. W. 1999. Worker allocation in insect societies: Coordination of nectar foragers and nectar receivers in honey bee (Apis mellifera) colonies. *Behavioral Ecology and Sociobiology* 46(2), pp. 73-81.

Ang, M. C., Pham, D. T. and Ng, K. W. 2009. *Minimum-time motion planning for a robot arm using the bees algorithm*. 2009 7th IEEE International Conference on Industrial Informatics, INDIN 2009, Cardiff, 23-26 June 2009, pp. 487-492.

Ang, M. C., Pham, D. T., Soroka, A. J. and Ng, K. W. 2010. *PCB assembly optimisation using the bees algorithm enhanced with TRIZ operators*. 36th Annual Conference of the IEEE Industrial Electronics Society, IECON 2010, Glendale, AZ, 7-10 November 2010, pp. 2708-2713.

Angeline, P. J. 1998. *Using selection to improve particle swarm optimization*. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, ICEC'98; Anchorage, AK, USA, 4-9 May 1998, pp. 84-89.

Ball, L., Troness, D., Ariyur, K., Huang, J., Rossi, D., Krupansky, P., Hickman, S. 2011. *TRIZ POWER TOOLS Job # 5 Resolving Problems*- Electronic Book. Available at: http://www.opensourcetriz.com/attachments/Resolving_Problems.pdf [Accessed: 30 April 2011] Tempe, Arizona, USA: Third Millennium Publishing.

Beekman, M. and Ratnieks, F. L. W. 2000. Long-range foraging by the honey-bee, Apis mellifera L. *Functional Ecology* 14(4), pp. 490-496.

Beyer, H. G. and Sendhoff, B. 2006. Functions with noise-induced multimodality: A test for evolutionary robust optimization - Properties and performance analysis. *IEEE Transactions on Evolutionary Computation* 10(5), pp. 507-526.

Bilchev, G. and Parmee, I. C. 1995. *The Ant Colony Metaphor for Searching Continuous Design Spaces*. AISB Workshop on Evolutionary Computing, pp. 25-39.

Bitam, S. Batouche, M. and Talbi, E. G. 2010. *A survey on bee colony algorithms*. 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010, Atlanta, GA, 19-23 April 2010.

Bonabeau, E., Dorigo, M. and Theraulaz, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

Cagnina, L. C., Esquivel, S. C. and Coello Coello, C. A. 2008. Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica (Ljubljana)* 32(3), pp. 319-326.

Chai-ead, N., Aungkulanon, P. and Luangpaiboon, P. 2011. *Bees and firefly algorithms for noisy non-linear optimisation problems*. International MultiConference of Engineers and Computer Scientists 2011, IMECS 2011, Kowloon, 16-18 March 2011, pp. 1449-1454.

Chaiyaratana, N. and Zalzala, A. M. S. 1997. *Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications*. Genetic Algorithms in Engineering Systems: Innovation and Applications. 2-4 September 1997, pp. 270-277.

Chong, C. S., Sivakumar, A. I., Low, M. Y. H. and Gay, K. L. 2006. *A bee colony optimization algorithm to job shop scheduling*. 2006 Winter Simulation Conference, WSC, Monterey, CA; 3-6 December 2006, pp. 1954-1961.

Curkovic, P. and Jerbic, B. 2007. Honey-bees optimization algorithm applied to path planning problem. *International Journal of Simulation Modelling* 6(3), pp. 154-164.

Deb, K. and Srinivasan, A. 2006. *Innovization: Innovating design principles through optimization*. 8th Annual Genetic and Evolutionary Computation Conference 2006, Seattle, WA, 8-12 July 2006, pp. 1629-1636.

Dereli, T. and Das, G. S. 2010. A hybrid 'bee(s) algorithm' for solving container loading problems. *Applied Soft Computing Journal* 11(2), pp. 2854-2862.

Di Caro, G. and Dorigo, M. 1998. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research* 9, pp. 317-365.

Digalakis, J. G. and Margaritis, K. G. 2002. An experimental study of benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics* 79(4), pp. 403-416.

Dimopoulos, G. G. 2007. Mixed-variable engineering optimization based on evolutionary and social metaphors. *Computer Methods in Applied Mechanics and Engineering* 196(4-6), pp. 803-817.

Dorigo, M., Bonabeau, E. and Theraulaz, G. 2000. Ant algorithms and stigmergy. *Future Generation Computer Systems* 16(8), pp. 851-871.

Dorigo, M., Di Caro, G. and Gambardella, L. M. 1999. Ant algorithms for discrete optimization. *Artificial Life* 5(2), pp. 137-172.

Dorigo, M., Maniezzo, V. and Colorni, A. 1996. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 26(1), pp. 29-41.

Duran-Novoa, R., Leon-Rovira, N., Aguayo-Tellez, H., Said, D. 2011. Inventive problem solving based on dialectical negation, using evolutionary algorithms and TRIZ heuristics. *Computers in Industry* 62(4), pp. 437-445.

Eberhart, R. and Kennedy, J. 1995. *New optimizer using particle swarm theory*. Proceedings of the 1995 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4-6 October 1995, pp.39-43.

El-Gallad, A., El-Hawary, M., Sallam, A. and Kalas, A. 2002. *Enhancing the particle swarm optimizer via proper parameters selection.* 2002 IEEE Canadian Conference on Electrical and Computer Engineering, Winnipeg, Manitoba, 12-15 May 2002, pp. 792-797.

Fiedler, M. 2011. *Matrices and Graphs in Geometry - Encyclopedia of Mathematics*. Cambridge, UK: CambridgeUniversity Press.

Frisch, K. V. 1950. *Bees: Their Vision, Chemical Senses, and Language*. New York: Vail-Ballou Press, Inc.
Frisch, K. V. 1953. *The Dancing Bees*. Munich: Methuen & Co. Ltd., London.

Ghanbarzadeh, A. 2007. *The Bees Algorithm A Novel Optimisation Tool.* Cardiff University.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Massachusetts, USA: Addison-Wesley Publishing Company, Inc.

Gordon, D. M. 1996. The organization of work in social insect colonies. *Nature* 380(6570), pp. 121-124.

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

He, S., Prempain, E. and Wu, Q. H. 2004. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization* 36(5), pp. 585-605.

Hu, X., Shi, Y. and Eberhart, R. 2004. *Recent advances in particle swarm*. Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, Portland, OR, 19-23 June 2004, pp. 90-97.

Janson, S., Middendorf, M. and Beekman, M. 2007. Searching for a new home - Scouting behavior of honeybee swarms. *Behavioral Ecology* 18(2), pp. 384-392.

Kannan, B. K. and Kramer, S. N. 1994. Augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design, Transactions of the ASME* 116(2), pp. 405-411.

Karaboga, D. 2005. *An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report*. Turkey: Erciyes University.

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

Karaboga, D. and Akay, B. 2011. A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. *Applied Soft Computing Journal* 11(3), pp. 3021-3031.

Karaboga, D., Akay, B. and Ozturk, C. 2007. *Artificial Bee Colony (ABC) optimization algorithm for training feed-forward neural networks*. 4th International Conference on Modeling Decisions for Artificial Intelligence, MDAI 2007, Kitakyushu, 16-18 August 2007, pp. 318-329.

Karaboga, D. and Basturk, B. 2007a. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization* 39(3), pp. 459-471.

Karaboga, D. and Basturk, B. 2007b. *Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems*. 12th International Fuzzy Systems Association World Congress, IFSA 2007, Cancun, 18-21 June 2007, pp. 789-798.

Karaboga, D. and Basturk, B. 2008. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing Journal* 8(1), pp. 687-697.

Kennedy, J. 1997. *Particle swarm: Social adaptation of knowledge*. Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, ICEC'97, Indianapolis, IN, USA, 13-16 April 1997, pp. 303-308.

Kennedy, J. 2000. *Stereotyping: Improving particle swarm performance with cluster analysis*. Proceedings of the 2000 Congress on Evolutionary Computation, California, CA, USA, 16-19 July 2000, pp. 1507-1512.

Kennedy, J. and Eberhart, R. 1995. *Particle swarm optimization*. Proceedings of the 1995 IEEE International Conference on Neural Networks. Part 1 (of 6); Perth, Australia, 27 November-1 December 1995, pp. 1942-1948.

Kennedy, J. and Eberhart, R. 1997. *Discrete binary version of the particle swarm algorithm*. Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Part 1 (of 5), Orlando, FL, USA, 12-15 October 1997, pp. 4104-4108.

Kennedy, J. and Spears, W. M. 1998. *Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator*. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, ICEC'98, Anchorage, AK, USA, 4-9 May 1998, pp. 78-83.

Kim, J., Kim, J., Lee, Y., Lim, W. and Moon, I. 2009. Application of TRIZ creativity intensification approach to chemical process safety. *Journal of Loss Prevention in the Process Industries* 22(6), pp. 1039-1043.

Koc, E. 2010. *The Bees Algorithm Theory, Improvements and Applications.* Cardiff University.

Krishnanand, K. N. and Ghose, D. 2009. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence* 3(2), pp. 87-124.

Lee, J. Y. 2010. *Multi-Objective Optimisation using the Bees Algorithm.* Cardiff University.

Lee, J. Y. and Haj Darwish, A. 2008. *Multi-objective Environmental/Economic*

Dispatch Using the Bees Algorithm with Weighted Sum. Proceedings of the EU-Korea Conference on Science and Technology (EKC2008). Heidelberg, Germany, 28-31 August 2008, pp. 267-274.

Lemmens, N., De Jong, S., Tuyls, K. and Nowe, A. 2007. *A Bee Algorithm for Multi-Agent Systems: Recruitment and navigation combined*. AAMAS '07. Honolulu, Hawaii USA.

Lemmens, N., De Jong, S., Tuyls, K. and Nowe, A. 2008. *Bee behaviour in multi-agent systems (a bee foraging algorithm).* 7th European Symposium on Adaptive and Learning Agents and Multi-Agent Systems, ALAMAS 2007, Maastricht, 2-3 April 2007, pp. 145-156.

Li, L. D., Li, X. and Yu, X. 2008. *Power generation loading optimization using a multi-objective constraint-handling method via PSO algorithm*. 6th IEEE International Conference on Industrial Informatics (INDIN 2008), Daejeon, 13-16 July 2008, pp. 1632-1637**.**

Lučić, P. and Teodorović, D. 2002. *Transportation modeling: An artificial life approach*. 14th International Conference on Tools with Artificial Intelligence, Washington, DC, 4 June 2002- 6 November 2002, pp. 216-223.

MacNish, C. 2007. Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. *Connection Science* 19(4), pp. 361-385.

Mann, D. L. 2008. *Systematic (Software) Innovation*. Devon, UK: Lazarus Press.

Mann, D. L. 2009. *Matrix 2010 Re-updating the TRIZ Contradiction Matrix*. Devon, UK: Lazarus Press.

Mann, D., Dewulf, S., Zlotin, B. and Zusman, A. 2003. *Matrix 2003 Updating the TRIZ Contradiction Matrix*. Belgium.

Mehrabian, A. R. and Lucas, C. 2006. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics* 1(4), pp. 355-366.

Meissner, M., Schmuker, M. and Schneider, G. 2006. Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training. *BMC Bioinformatics 7.*

Mezura-Montes, E. and Coello Coello, C. A. 2005. *Useful infeasible solutions in engineering optimization with evolutionary algorithms*. 4th Mexican International Conference on Artificial Intelligence, MICAI 2005, Monterrey, 14-18 November 2005, pp. 652-662.

Nakrani, S. and Tovey, C. 2004. On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior* 12(3-4), pp. 223-240.

Osyczka, A. 2002. *Evolutionary Algorithms for Single and Multicriteria Design Optimization*. Physica-Verlag Heidelberg, New York.

Parrish, J. K. Viscido, S. V. and Grünbaum, D. 2002. Self-organized fish schools: An examination of emergent properties. *Biological Bulletin* 202(3), pp. 296-305.

Parsopoulos, K. E. and Vrahatis, M. N. 2005. *Unified Particle Swarm Optimization for solving constrained engineering optimization problems*. First International Conference on Natural Computation, ICNC 2005, Changsha, 27-29 August 2005, pp. 582-591.

Passino, K. M. and Seeley, T. D. 2006. Modeling and analysis of nest-site selection by honeybee swarms: The speed and accuracy trade-off. *Behavioral Ecology and Sociobiology* 59(3), pp. 427-442.

Passino, K. M., Seeley, T. D. and Visscher, P. K. 2008. Swarm cognition in honey bees. *Behavioral Ecology and Sociobiology* 62(3), pp. 401-414.

Pham, D. T. and 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), pp. 2919-2938.

Pham, D. T. and Ghanbarzadeh, A. 2007. *Multi-Objective Optimisation using the Bees Algorithm*. Proceedings 3rd International virtual conference on Innovative Production Machines and Systems (IPROMS). 2-13 July 2007. pp. 529-533.

Pham, D. T. and 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), pp. 885-892.

Pham, D. T., Afify, A. and Koc, E. 2007a. *Manufacturing Cell Formation using the Bees Algorithm*. Proceedings 3rd International Virtual Conference on Innovative Production Machines and Systems (IPROMS 2007), Scotland, pp. 523-528.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. 2005. *The Bees Algorithm - Technical Report*. Cardiff: Manufacturing Engineering Centre, Cardiff University.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. 2006a. *The Bees Algorithm - A Novel Tool for Complex Optimisation Problems*.

Proceedings 2nd International virtual conference on Intelligent Production Machines and Systems (IPROMS). 3-14 July 2006, pp. 454-459.

Pham, D. T., Soroka, A. J., Ghanbarzadeh, A., Koc, E., Otri, S. and Packianather, M. 2006b. *Optimising neural networks for identification of wood defects using the bees algorithm*. 2006 IEEE International Conference on Industrial Informatics, INDIN'06, Singapore, 16-18 August 2006, pp. 1346-1351.

Pham, D. T., Koç, E., Lee, J. Y. and Phrueksanant, J. 2007b. *Using the Bees Algorithm to schedule jobs for a machine*. The 8th International Conference on Laser Metrology, CMM and Machine tool performance (LAMDAMAP), Cardiff, UK, pp. 430-439.

Pham, D. T., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koç, E. and Otri, S. 2007c. *Using the Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification*. Proceedings 3rd International virtual conference on Innovative Production Machines and Systems (IPROMS), pp. 540-545.

Pham, D. T., Pham, Q. T., Ghanbarzadeh, A. and Castellani, M. 2008. *Dynamic Optimisation of Chemical Engineering Processes Using the Bees Algorithm.* The 17th World Congress The International Federation of Automatic Control. Seoul, Korea, 06-11 July 2008. pp. 6100-6105.

Rahmat-Samii, Y. 2007. *Modern antenna designs using nature inspired optimization techniques: Let Darwin and the bees help designing your multi band MIMO antennas*. 2007 IEEE Radio and Wireless Symposium, RWS, Long Beach, CA, 9 -11 January 2007, pp. 463-466.

Ray, T. and Liew, K. M. 2003. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation* 7(4), pp. 386-396.

Sato, T. and Hagiwara, M. 1997. *Bee System: Finding solution by a concentrated search*. Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Part 3 (of 5); Orlando, FL, USA; 12-15 October 1997, pp. 3954-3959.

Seeley, T. D. 2002. When is self-organization used in biological systems? *Biological Bulletin* 202(3), pp. 314-318.

Seeley, T. D. and Buhrman, S. C. 1999. Group decision making in swarms of honey bees. *Behavioral Ecology and Sociobiology* 45(1), pp. 19-31.

Seeley, T. D. and Buhrman, S. C. 2001. Nest-site selection in honey bees: How well do swarms implement the "best-of-N" decision rule? *Behavioral Ecology and Sociobiology* 49(5), pp. 416-427.

Seeley, T. D., Visscher, P. K. and Passino, K. M. 2006. Group decision making in honey bee swarms. *American Scientist* 94(3), pp. 220-229.

Shang, Y. W. and Qiu, Y. H. 2006. A note on the extended Rosenbrock function. *Evolutionary Computation* 14(1), pp. 119-126.

Shi, Y. and Eberhart, R. 1998. *Modified particle swarm optimizer*. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, ICEC'98, Anchorage, AK, USA, 4-9 May 1998, pp. 69-73.

Sholedolu, M. O. 2009. *Nature Inspired Optimisation: Improvements to the Particle Swarm Optimisation Algorithm and the Bees Algorithm.* Cardiff University.

Socha, K. and Dorigo, M. 2008. Ant colony optimization for continuous domains. *European Journal of Operational Research* 185(3), pp. 1155-1173.

Su, C. T. and Lin, C. S. 2008. A case study on the application of fuzzy QFD in TRIZ for service quality improvement. *Quality and Quantity* 42(5), pp. 563-578.

Sumpter, D. J. T. 2006. The principles of collective animal behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences* 361(1465), pp. 5-22.

Sundareswaran, K. and Sreedevi, V. T. 2008. *Development of novel optimization procedure based on honey bee foraging behavior*. IEEE International Conference on Systems, Man and Cybernetics, SMC 2008, Singapore, 12-15 October 2008, pp. 1220-1225.

Swarms 2011, BBC, [Online video]. Available at: http://www.youtube.com/watch?v=Ua2quxUDyRk [Accessed: 31 May 2011]

Teodorović, D. 2008. Swarm intelligence systems for transportation engineering: Principles and applications. *Transportation Research Part C: Emerging Technologies* 16(6), pp. 651-667.

Teodorović, D. and Dell'Orco, M. 2005. *Bee Colony Optimization- A Cooperative Learning Approach to Complex Transportation Problems*. Advanced OR and AI Methods in Transportation, pp. 51-60.

Wedde, H. F. and Farooq, M. 2005. *A performance evaluation framework for nature inspired routing algorithms*. EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, 30 March- 1 April 2005. pp 136-146.

Wedde, H. F., Farooq, M. and Zhang, Y. 2004. *BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior*. 4th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2004, Brussels, 5-8 September 2004, pp. 83-94.

Wedde, H. F., Farooq, M., Pannenbaecker, T., Vogel, B., Mueller, C., Meth, J. and Jeruschkat, R. 2005. *BeeAdHoc: An energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior*. GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, D.C., 25-29 June 2005, pp. 153-160.

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

Wong, L. P., Low, M. Y. H. and Chong, C. S. 2008. *Bee Colony Optimization with local search for Traveling Salesman Problem*. IEEE INDIN 2008: 6th IEEE International Conference on Industrial Informatics; Daejeon, Korea, 13-16 July 2008, pp. 1019-1025.

Yang, C., Chen, J. and Tu, X. 2007. *Algorithm of fast marriage in honey bees optimization and convergence analysis*. 2007 IEEE International Conference on Automation and Logistics, ICAL 2007, Jinan, 18-21 August 2007, pp. 1794-1799.

Yang, X. S. 2005. *Engineering optimizations via nature-inspired virtual bee algorithms*. First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005, Las Palmas, Canary Islands, 15-18 June 2005, pp. 317-323.

Yang, X. S. 2009. *Firefly algorithms for multimodal optimization*. 5th Symposium on Stochastic Algorithms, Foundations and Applications, SAGA 2009, Sapporo, 26-28 October 2009, pp. 169-178.

Yang, X. S and Deb, S. 2009. *Cuckoo Search via Levy flights*. Nature & Biologically Inspired Computing, NaBIC 2009. India, 9-11 December 2009, pp. 210- 214.

Yang, X. S and Deb, S. 2010. Engineering Optimisation by Cuckoo Search. *Int. J. Mathematical Modelling and Numerical Optimisation* 1(4), pp. 330-343.

Zhao, X., Yao, Y. and Yan, L. 2009. Learning algorithm for multimodal optimization. *Computers & Mathematics with Applications* 57(11-12), pp. 2016-2021.

Zhong, W. L., Zhang, J. and Chen, W. N. 2007. *A novel discrete particle swarm optimization to solve traveling salesman problem*. 2007 IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, pp. 3283-3287.