Enhancements for the Bees Algorithm

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

Doctor of Philosophy

By

Azar Imanguliyev

Manufacturing Engineering Centre

School of Engineering

Cardiff University

United Kingdom

September 2013

ABSTRACT

This work introduces new enhancements to the Bees Algorithm in order to improve its overall performance. These enhancements are early neighbourhood search process, efficiency based recruitment for neighbourhood search process, hybrid strategy involving tabu search, new escape mechanism to escape locals with similar fitness values and autonomy to minimise interaction between search process and the user.

The proposed enhancements were applied alone or in pair to develop improved versions of the Bees Algorithm. Three Enhanced Bees Algorithms were introduced: the Early Neighbourhood Search and Efficiency Based recruitment Bees Algorithm (ENSEBRBA), the Hybrid Tabu Bees Algorithm (TBA) and the Autonomous Bees Algorithm (ABA).

The ENSEBRBA with an empowered initialisation stage and extra recruitment for neighbourhood search is introduced to improve performance of the Bees Algorithms on high dimensional problems.

The TBA is proposed as a new version of the Bees Algorithm which utilises the memory lists to memorise less productive patches. Moreover, the local escape strategy was also implemented to this algorithm. Proposed modifications increased the productivity of the Bees Algorithm by decreasing number of evaluations needed to converge to the global optimum.

The ABA is developed to provide independency to the Bees Algorithm, thus it is able to self tune its control parameters in a sub-optimal manner.

All enhanced Algorithms were tested on continuous type benchmark functions and additionally, statistical analysis was carried out. Observed experimental results proved that proposed enhancements improved the Bees Algorithm’s performance.

ACKNOWLEDGEMENTS

I would like to thank my first supervisor Professor Duc Truong Pham for his excellent supervision, help and continuous encouragement. My thanks go to him for accepting me to be one of his students in the Manufacturing Engineering Centre (MEC) at Cardiff University.

In addition, without the assistance of my main supervisors Dr. Michael Packianather and Professor Rossi Setchi this thesis might not have been possible to produce.

I also thank to the Ministry of Education of The Azerbaijan Republic for sponsoring me during my study.

Also, special thanks to Dr. Fuad Omar, Dr. Baris Yuce, Dr. Mario Javier Gonzalez Romo, Dr Janyarat Phrueksanant and Dr. Siti Azfanizam Ahmad.

Last but not least, I thank my lovely family for always supporting me; mentally, emotionally and physically during this four years journey.

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 ……. … (Azar Imanguliyev) Date…….

STATEMENT 1

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

Signed ……. … (Azar Imanguliyev) 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 ……. … (Azar Imanguliyev) 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 ……. … (Azar Imanguliyev) Date…….

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABA	The Autonomous Bees Algorithm
ABC	The Artificial Bee Colony
ACO	The Ant Colony Optimisation
BA	The Bees Algorithm
BBA	The Basic Bees Algorithm
D	Dimension
ENSEBRBA	The Early Neighbourhood Search and Efficiency-based Recruitment Bees Algorithm
EP	The Evolutionary Programming
ER	Efficiency Rate
GA	The Genetic Algorithm
GP	The Genetic Programming
IR	Improvement Ratio
MEC	Manufacturing Engineering Centre
PSO	The Particle Swarm Optimisation
RO	The Random Optimisation
SA	The Simulated Annealing
SHC	The Stochastic Hill Climbing SHC
TA	The Tabu Search
TBA	The Tabu Bees Algorithm
TSP	Travelling Salesman Problem

LIST OF SYMBOLS

Chapter 2	
$f(X)$	Objective function
$\mathbf{A}$	Candidate solution
$\Re^n$	Euclidian space
X	Local optima
$p_{ij}^{\ k}(t)$	The transition probability from node *i* to node *j*
τ_{ij}	The posterior effectiveness of the move from node *i* to node j
η_{ij}	Already available heuristic information
α	Influence of pheromone trail
β	Influence of heuristic information.
M	Number of ants
$\Delta\tau^k{}_{ij}(t)$	The amount of pheromone deposited by ant m from node *i* to node *j* at time step *t* .
Vn	The particle velocity in iteration *n*
Pn	The particle position in iteration *n*
Pbestn	“personal” best position in iteration *n*
Gbestn	“global” best position in iteration *n*
C1,C2	Weight factors

w	Inertia weight
n	Number of scout bees
m	Number of sites selected out of n visited sites
e	Number of best sites out of m selected sites
nep	Number of bees recruited for best e sites
nsp	Number of bees recruited for the other (m-e) selected sites
ngh	Patch size around a selected best location
Chapter 3	
nab	The number of added bees according to the efficiency calculation
β	Number of iterations to do Efficiency calculation
$\Delta F_{\min}$	Difference of less productive fitness values in β iteration
$\Delta F_{\max}$	Difference of most productive fitness values in β iteration
ΔF_i^j	Difference of the fitness values for the searched patch in β iteration
α	Confidence level for the t-test
Chapter 4	
w	Worst patches out of n-m
Tnbh	Test neighbourhood area

Chapter 1

Introduction

1.1 Introduction

Population growth and resource depletion creates tough competition in many areas of life. In order to address these issues, industries try to maximise their productions. Therefore, optimisation techniques become an important tool for efficient operation.

Optimisation is a process of seeking the values of variables to find an optimal solution for the optimisation problem that needs to be maximised or minimised. There are various types of optimisation techniques available in the literature. These techniques can be classified in many different ways. One such method is to classify based on their variables. Classification based on variables divides optimisation techniques into two groups, deterministic and stochastic. To solve problems in polynomial time, deterministic optimisation techniques are used. On the other hand there are optimisation problems which cannot be solved in polynomial time. Stochastic optimisation techniques are utilised to solve these types of problems. Many stochastic optimisation techniques such as Genetic Algorithm Evolutionary Programming, Particle Swarm Optimisation, The Ant Colony technique or the Bees Algorithm were inspired by nature.

The motivation for this research is described in the following section.

1.2 Motivation

The Bees Algorithm is a stochastic optimisation technique inspired by the foraging behaviour of honey bees. The Bees Algorithm has both global exploration and local exploitation strategies which increase the success rate of the algorithm in finding the global optimum. In order to demonstrate its performance, the Bees Algorithm was implemented on several single and multi-objective functions. The Basic Bees Algorithm has undergone many improvements since it was introduced in 2005 by Professor D.T Pham and colleagues. Most of the improvements were focused on the neighbourhood search site such as an abandonment strategy, population and neighbourhood size change strategies. The other improvements were focused on parameter tuning and hybridisation of the basic Bees Algorithm with other well-known optimisation techniques, such as Ant Colony and Particle Swarm Optimisation techniques.

Although several modifications were introduced to the Bees Algorithm, there is still opportunity for further improvements. For example, the Bees Algorithm has certain weaknesses which were not studied properly, such as a poor initialisation stage, the absence of the memory and number of parameters. Moreover, new neighbourhood search strategies can also be developed to make the Bees Algorithm more competitive.

1.3 Aims and objectives

The overall aim of this study is to explore the possibilities of further improvements to the Bees Algorithm for single objective optimisation problems.

The following objectives were set to achieve this aim:

- Develop a strategy to improve the initialisation stage of the Bees Algorithm.
- Develop an adaptive neighbourhood search strategy to improve the Bees Algorithm's performance on high dimensional optimisation problems.
- Provide memory to the Bees Algorithm to avoid site repetitions.
- Develop a strategy for the Bees Algorithm to prevent producing similar fitness values around local optimum.
- Develop a version of the Bees Algorithm which does not need to be tuned manually for each problem.

1.3 Research methods

To carry out this research, the following methodologies were used:

- Surveying previous work related to optimisation algorithms focusing on swarm-based optimisation techniques.
- Studying all available versions of the Bees Algorithm.
- Developing three new versions of the Bees Algorithm.
- Implementing the proposed algorithms in MATLAB
- Utilising the proposed algorithms to solve continuous-type benchmark functions.
- Comparing results with some other optimisation techniques for the verification of the algorithm.
- Testing the statistical significances of the algorithms using the T-test.

1.4 Outline of the thesis

The remainder of this thesis is organised as follows:

Chapter 2 reviews both stochastic and deterministic optimisation techniques. The chapter is mainly focussed on stochastic optimisation techniques. Also, the Basic Bees Algorithm is described in detail.

Chapter 3 presents the Bees Algorithm with Early Neighbourhood Search and Efficiency-based Recruitment. The proposed algorithm has been tested on continuous-type benchmark functions. Also, compared results with other well known optimisation algorithms are presented in this chapter. Moreover, statistical analysis has been carried out using a T-test.

Chapter 4 introduces a Hybrid Tabu Bees Algorithm. The proposed algorithm was tested on Continuous-type benchmark functions. In addition, results were compared to the Basic Bees Algorithm and The Bees Algorithm with Early Neighbourhood Search and Efficiency-based Recruitment. Moreover, statistical analysis has been carried out using a T-test.

Chapter 5 presents the Autonomous Bees Algorithm. The proposed algorithm was tested on Continuous-type benchmark functions. In addition, results were compared to the Basic Bees Algorithm. T-test results are also included.

Chapter 6 summarises the conclusions and contributions of the research, and gives suggestions for further investigations.

Chapter 2

Optimisation Techniques

2.1 Preliminaries

This chapter presents an overview of current optimisation techniques. There are various methods to classify and one of them is classification based on the type of variables. The main focus of this chapter is on stochastic optimisation techniques but brief information about deterministic methods is also provided.

2.2 Optimisation Techniques

The mathematical technique concerned with finding the “best” solution for a problem, where the “best” refers to the fittest solution in the solution space, is called optimisation. In many fields like physics, chemistry, medicine, manufacturing or economic analysis, various optimisation techniques have been used. However, there is no optimisation technique which is suitable for every problem (Wolpert and Macready, 1997). A block diagram of the optimisation process is given in Figure 2.1 (Chinneck, 2000).

Moving from the real world problem to the algorithm, model or solution technique is called analysis. Here, the main task is eliminating non-crucial details and focusing on important elements. Moving from the algorithm, model, solution technique to the computer implementation is called numerical methods, and from computer implementation back to the algorithm, model, solution technique is called verification.

Finally, moving from the algorithm, model, solution technique to real world problems is called validation and sensitivity analysis. In this step, obtained results are compared with the real world and in case of failure; the process goes to the next cycle.

The goal of an optimisation is to maximise or minimise the objective function concerning constraints and search space. An example is given below:

Given:

Function:

$f(X)$ defined as $f : A \rightarrow \Re^n$

$X \in A$ and A is subset of n dimensional Euclidian space $\Re^n$

Constraints:

Inequality constraints: $a_i(X) \leq 0, \quad i = 1,2.....m$

Equality constraints: $b_i(X) = 0, \quad i = 1,2.....p$

Sought:

Maximisation: $\max \in A$ such that $f(\max) \geq f(X)$ for all $X \in A$

Minimisation: $\min \in A$ such that $f(\min) \leq f(X)$ for all $X \in A$

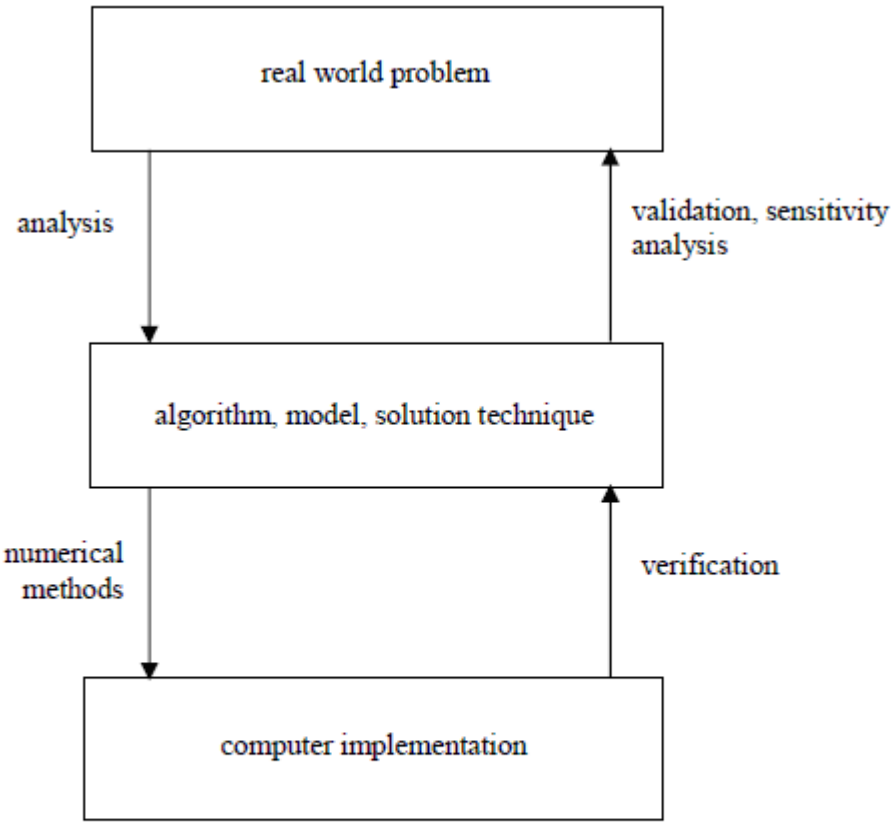

Figure 2.1: Block diagram of the optimisation process (Chinneck, 2000).

Where *f(X)* is called the “objective function”, *f* is called the “search space” or “parameter space”, each element of A is called the “candidate solution” (Blondin, 2009).Candidate solutions are tested in the objective function to find an “optimal solution”. An optimal solution is the maximised or minimised solution of an objective function.

2.2.1 Classification of the optimisation techniques

Many different strategies can be used to classify optimisation techniques. One of these strategies is classification of optimisation techniques based on the nature of the variables. In this classification, optimisation techniques are distributed in to two different groups (deterministic and stochastic optimisation techniques) depending on whether their variables are deterministic or stochastic. Figure 2.2 illustrates the variable-based classification of optimisation techniques (Weise, 2009).

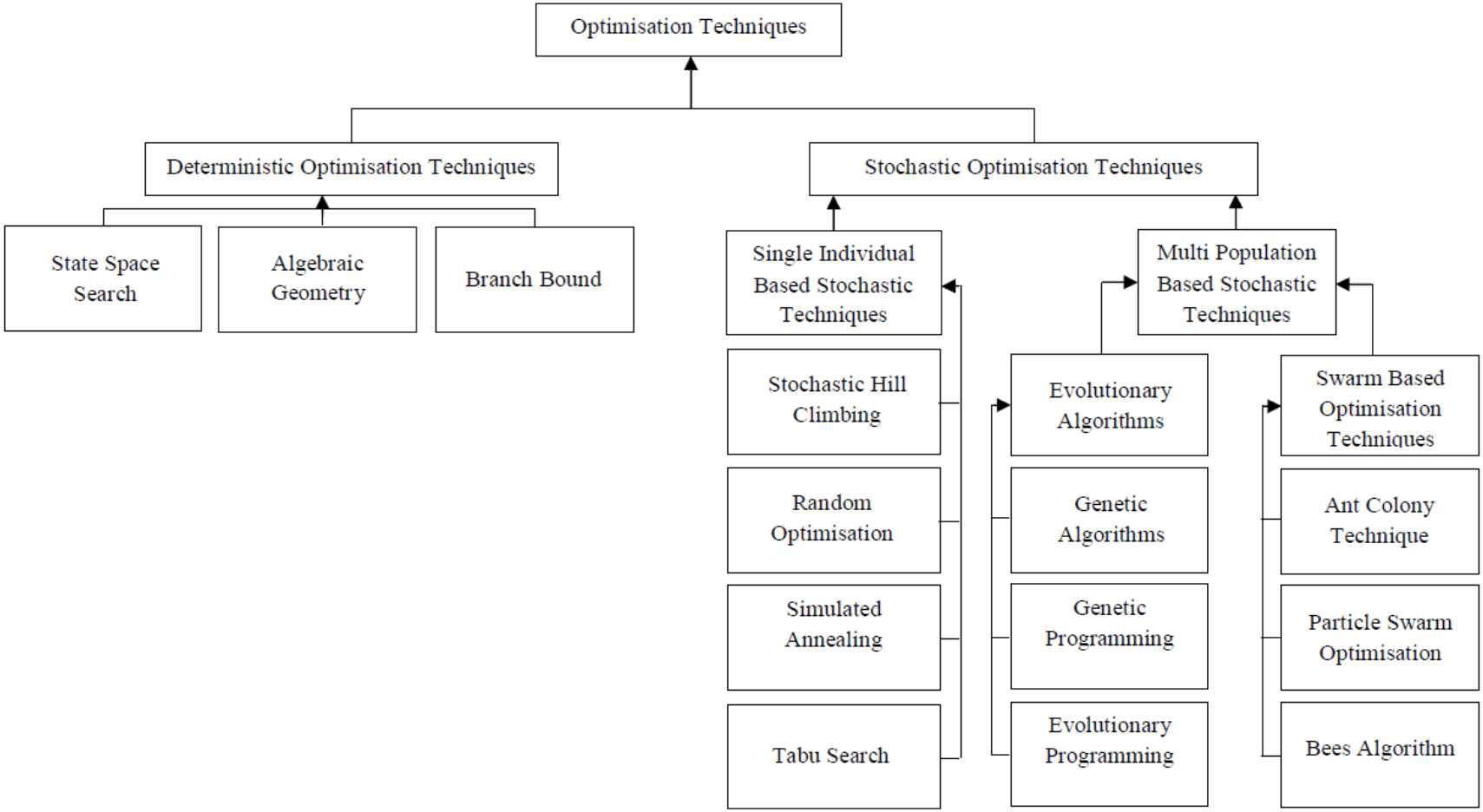

Figure 2.2 Classification of optimisation techniques based on types of the parameters (Baris, 2012).

2.2.2 Deterministic optimisation techniques

Deterministic optimisation techniques are those where a direct relation exists between the characteristics of the possible solutions and their utility and they can be solved in polynomial time.

Examples of deterministic optimisation techniques are given below.

2.2.2.1 State space search.

State space search is a deterministic search method. Information is needed to guess the effects of an action and to decide if it is a goal state recorded in **state (**David Poole and Alan Mackworth, 2010**)**. State space searching considers that the agent has complete knowledge about state space and can tell what state it is in:

- the agent has a set of actions that have known deterministic effects;
- there are more than one goal states, the agent can identify them and agent wants to reach that state.
- sequence of actions to get the agent from its current state to a goal state is a solution

2.2.2.2 Algebraic geometry.

Algebraic geometry is a branch of mathematics, classically studying zeros of polynomial equations. Thus, the technique focuses on the resolution of the stationary conditions in the polynomial optimisation as a system of polynomial equations (Kavasseri and Nag, 2007).

2.2.2.3 Branch bound.

Branch bound are techniques to solve discrete and combinatorial optimisation problems (A. H. Land and A. G. Doig, 1960). The idea of a branch bound search is to maintain the lowest-cost path to a goal found so far, and its cost **(**David Poole and Alan Mackworth, 2010**)**. A branch bound algorithm starts with setting the cost as a bound. If the search finds a path *p* where $cost(p) \geq bound$, path *p* can be eliminated. Only a better path to the goal will be accepted. Any further new better solution is memorised and *bound* is set to the cost of this new solution. The process continues until all paths have been checked.

2.2.3 Stochastic optimisation techniques

If the relationship between the candidate solution and the problem's fitness is not clear or the problem has no solution in polynomial time, then stochastic optimisation

techniques bring a different solution which searches for optimum value, generating random variables.

2.2.3.1 Stochastic hill climbing

The Stochastic Hill Climbing (SHC) technique is a local search technique which is based on a direct search strategy (Schmidhuber and Zhao, 1999). SHC climbing attempts to maximize (or minimize) a target function *f(X)*. At each iteration, hill climbing will change one element in $\mathbf{X}$ to find if the change improves the value of *f(X)*. Any change that improves *f(X)* is accepted and this process continues until no improvements can be found. Final X is called the "local optima" of the problem.

2.2.3.2 Random optimisation

The Random Optimisation (RO) technique is one of the most straightforward numerical techniques used to search for the global optimum which does not require the gradient of the problem (Li and Rhinehart, 1998). RO is used as starting point for most stochastic-based optimisation techniques (Kristoffersen, 2007).

The point at which to start the RO is chosen randomly. There is a "*reproduce*" operator in RO which is responsible for reaching all of the points in the search space from every other point (Weise, 2009).

2.2.3.3 Simulated annealing

Annealing is a metallurgical technique involving heating and the controlled cooling of materials in order to change the size of their crystals. This affects some of their physical properties including strength, hardness and ductility (Koppen and et al., 2011). Slow temperature change gives a material the right hardness and ductility but if the temperature change is too rapid, the metal may become too weak. Simulated Annealing (SA) is a single-point random search technique imitating the annealing process (Goffe et al., 1994). It is one of main methods to locate an approximation of the global minimum / maximum for problems with a large search space (Koziel and Yang, 2011). The Slow controlled cooling process of the material is implemented as a slow decrease in the probability of accepting worse solutions while exploring the solution space. Accepting worse solutions allows more extensive search for the optimal solution.

2.2.3.4 Tabu search

Tabu Search (TS) is a Single Individual Based Stochastic search technique with a local optima avoidance mechanism (Pham and Karaboga, 2000).

As for every local search algorithm, TS takes a potential solution to a problem and checks its neighbourhood to find an improved solution. The main problem with most local search methods is getting stuck in areas where many solutions are equally fit but

in Tabu Search this problem is solved by implementing a special memory unit called 'tabu list'(Tsubakitani and Evans, 1998) .

Previously visited or not satisfactory solutions are recorded in the 'tabu list'. All data in this list is marked as tabu and this helps algorithm to shrink the search space.

Three different structures can be used while creating 'tabu list' (F. Glover, 1990).

- Short-term: The list of recently considered solutions. The size of the list is limited and with every new element entering the list, the oldest one is erased. When a potential solution appears on this list, the algorithm does not revisit it until a solution drops out from list.

- Intermediate-term: A list of rules to lead the search in the direction of the promising areas of the search space.

- Long-term: A list of rules that brings variety in the search process. As an example, the algorithm can reset when it becomes stuck around equally fit solutions.

The pseudo code of the Tabu Search with short term memory for minimising the cost function is given as an example in figure 2.3 (Jason Brownlee, 2011).

As for every algorithm, Tabu Search has some weaknesses. One of biggest weaknesses of Tabu Search is being effective on discrete spaces because it is very rare for the algorithm to visit the same point in real value spaces (Sean Luke, 2009).

```
sBest ← initial solution
tabuList ← null
while (not stoppingCondition())
   candidateList ← null
   for(sCandidate in sNeighborhood)
        if(not containsTabuElements(sCandidate, tabuList))
             candidateList ← candidateList + sCandidate
        end
   end
   sCandidate ← LocateBestCandidate(candidateList)
   if(fitness(sCandidate) > fitness(sBest))
        tabuList ← featureDifferences(sCandidate, sBest)
        sBest ← sCandidate
        while(size(tabuList) > maxTabuListSize)
            ExpireFeatures(tabuList)
        end
   end
end
return(sBest)
```

Figure 2.3 Pseudo code of Tabu Search with short term memory.

2.2.3.5 Genetic algorithms

The Genetic Algorithm (GA) is population-based algorithm which was proposed by Holland in 1975. In 1983 GA's engineering applications were studied by Goldberg. In nature only strong species pass their genes to future generations when weak ones are facing extinction. This phenomenon was the inspiration for the creation of the Genetic Algorithm. During many years, various modifications to the original structure of GA were proposed. To distinguish it from numerous versions of the algorithm, the original GA proposed by Holland is often referred to as the 'canonical' GA. Crossover and Mutation are fundamental operators of the canonical GA (Rutkowski, 2008).

Crossover creates offspring by randomly mixing sections of the parental genome. *One-point crossover*, *two-point crossover* and *uniform crossover are* the most common crossover procedures. (Davis, 1991). Couples not selected for recombination will generate two offspring identical to the parents.

A small group of the offspring are randomly chosen to be mutated. Mutation is the changing of the bit value, in the case of a binary coding, from 0 to 1 and vice versa (Ho *et al.*, 1999). The mutation operator is not extremely important. However, it provides diversity to the genetics of the created population.

For GA's better performance, mutation and crossover rates are two important parameters requiring careful tuning (Eiben and Smit, 2011).

2.2.3.6 Genetic programming

Genetic programming (GP) is a set of instructions and a fitness function to measure a computer's performance on a given task. GP is a specific type of genetic algorithm (GA) where each individual is a computer program. Therefore, GP's operators are basically GA's operators (Banzhaf, W 1998).

2.2.3.7 Evolutionary programming

Evolutionary programming (EP) is evolutionary algorithm developed by Lawrence J. Fogel in 1960. EP uses simulated evolution for the learning process to generate artificial intelligence (Back *et al.*, 1997). Traditional EP uses the Gaussian mutation operator. Traditional EP has no crossover operator. However, in the modern version of EP there is a crossover operator and the population for crossover will be selected by a mutation operator. In modern EP the mutation operator is adaptive.

The steps for modern EP are given below:

- Firstly generate an initial population,
- Secondly EP duplicates the initial solutions. After duplication each solution is mutated using any chosen distribution function,
- The last step is the evaluation of the crossover solution of population.

2.2.3.8 Ant colony optimisation

Ant colony optimisation (ACO) in swarm-based optimisation techniques was introduced by M.Dorigo and his colleagues, inspired by the behaviour of real ants. ACO was developed to solve combinatorial optimisation problems (Dorigo *et al.*, 1996).

In nature, ants scout for food randomly wandering around their nest. Every scout ant explores a wide area to find sources of food. When any of them find food they bring it back to the nest. On the way back, the ant marks its passage by laying down a pheromone trail (Shtovba, 2005). If another ant finds such a path, it stops random scouting and checks for the food source at the end of the trail. In case of success it goes back to the nest and brings reinforcements to collect the food more effectively (Dorigo and Stutzle, 2004). Reinforcement ants will lay down pheromones on that trail as well. Pheromones evaporate in time. The main foraging behaviour of ants is based on finding the shortest path between the source and their nest (Panigrahi *et al.*, 2011). The pheromone level on a shorter path will be reinforced but it will evaporate as time passes (Sumathi and Surekha, 2010). A short path will be visited by more ants and thus the pheromone level will be higher compared to other paths. That is why pheromone density on short passes will remain higher than that on long passes. After observing this behaviour of ants, ACO was created.

Steps for the simple version of ACO is given in Figure 2.4.

```
Start
  While (stopping criterion not met)
      Generate solutions
      Pheromone update using equation (2.2)
      Move according probability calculated with equation (2.1)
   End While.
End
```

Figure 2.4: Steps for simple version of ACO

The first problem where ACO was used was the Travelling salesman problem (TSP) (M. Dorigo, 2003). In ACO, each ant is initially placed in a random location (city) and has a memory which stores the partial solution it has constructed so far in that city. Each ant starts to move from city to city. Ant *k* decides to move from city *i* (initial location) to city *j* with provided probability:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in N^{k}_{i}} [\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta}} \qquad \text{If} \qquad j \in N^{k}_{i} \qquad (2.1)$$

- $\eta_{ij} = 1/d_{ij}$ is a already available heuristic information,
- α and β are parameters to determine the influence of pheromone trail and heuristic information.
- N is cities around ant *k* which were not visited yet. After every ant has completed a tour solution construction ends. Next step is updating pheromone trails. The update process shown in given equation.

$$\tau_{ij}(t+1) = (1-p) * \tau_{ij}(t) + \sum_{k=1}^{M} \Delta\tau^{k}{}_{ij}(t) \qquad \forall(i, j) \qquad (2.2)$$

- $0 < \rho \leq 1$ is the evaporation rate of the pheromone trail.
- *M* is the number of ants.

- $\Delta\tau^{k}{}_{ij}(t)$ is the amount of pheromone deposited by ant m from node *i* to node *j* at time step *t* .

2.2.3.9 Particle swarm optimisation

The Particle Swarm Optimisation (PSO) is a swarm-based optimisation algorithm which was proposed by Eberhart and Kennedy (Eberhart and Kennedy, 1995). Inspiration for the creation of PSO was the socially organised behaviour of different animal populations such as animal herds or bird flocks (Blum and Merkle, 2008). The concept of PSO gained in popularity due to its simplicity. Individuals in PSO are called particles and a population is called a swarm (Li and Liu, 2011). Each particle has a position and velocity. Particles are freely flying in the search space by at a given velocity. In each iteration, the velocities of particles are stochastically changed based on the previous best position for the particle itself and the neighbourhood best position. Basically, particles of PSO are travelling in the search space and change their positions from time to time. This change happens based on their previous experience and the experiences of their neighbours. This behaviour allows particles to move toward better locations while being able to explore a wider area.

The PSO Algorithm has been successfully applied to a number of optimisation problems such as; determination of optimum location and its type (Onwunalu and Durlofsky, 2010), determination of the optimum constriction factors, inertia weights, and tracking dynamic systems (Eberhart and Yuhui, 2001). Due to its simplicity and relatively low number of parameters than other algorithms, PSO has become very popular.

The pseudo code for a simple version of PSO is given below (Figure 2.5).

```
For each particle
        Initialise position P0 and  velocity V0
End

 While maximum iterations are not exceeded or
     minimum error is not attained

      Do For each particle
          Calculate fitness value
           If fitness better than Pbest
          Update Pbest
    End

   Determine Gbest among all particles

      For each particle
        Update position
        Update velocity
      End
End
```

Figure 2.5: Pseudo code for simple PSO.

Every iteration velocity and position of the particles change based on 2 cryteria:

- Pbest : this is the best position visited by particle itself (local optimum).

- Gbest : this is best position visited by any particles of the swarm (global optimum).

Equations for velocity and position updates of the particles are given below:

$$Vn+1 = wVn + c1 * rand1*(Pbestn – Pn) + c2 * rand2 * (Gbestn – Pn) \quad (2.3)$$

$$Pn+1 = Pn + k * Vn+1 \quad (2.4)$$

where:

Vn , is the particle velocity in iteration *n*

Pn , is the particle position in iteration *n*

Pbestn is “personal” best position in iteration *n*

Gbestn is “global” best position in iteration *n*

rand1 and *rand2* are random numbers between 0 and 1

c1, c2 are weighting factors. These factors determine the size of movement a particle can do in a single step (number in the range 0 to 4)

w is the ‘inertia’ weight. If w has large value it performs a global search. If it is small then it performs a local search.

2.2.3.10 Artificial Bees Colony

The Artificial Bee Colony (ABC) algorithm is a swarm-based meta-heuristic optimisation technique inspired by the intelligent foraging behaviour of honey bees which was proposed by Karaboga in 2005 (Karaboga, 2005). Base for the ABC algorithm was the model proposed by Tereshko and Loengarov (Tereshko *et al.*, 2005) for the foraging behaviour of honey bee colonies.

The model proposed by Tereshko and Loengarov has three main components: Food sources, employed and unemployed bees. Employed bees are foragers employed at a promising food source. Unemployed bees are divided into two groups:

- Scouts: Bees looking for a new food source.

- Onlookers: Bees waiting at the hive for information about the food source (They get information related to the food sources from employed bees)

The model defines two type of behaviour: the recruitment to a nectar source and the abandonment of a source.

- Recruitment: Scouts become employed bees when they find a promising food source. Onlookers convert to employed bees as well when they get necessary information on a food source.

- Abandonment: Employed bees abandon an extinct source. Some bees go for further scouting; some fly back to the hive and become onlookers.

After observing the proposed model, the ABC algorithm was developed. The ABC algorithm follows the rules of the proposed model. The main steps of the algorithm are given below (Figure 2.6):

Send the scouts to the initial food sources

REPEAT

Send the employed bees to the food sources and determine their nectar amounts

Calculate the probability value of the sources which are preferred by the onlooker bees

Send the onlooker bees to the food sources and determine their nectar amounts

Stop the exploitation process of the sources exhausted by the bees

Send the scouts into the search area for the discovery of new food sources, randomly

Memorize the best food source found so far

UNTIL (requirements are met)

Figure 2.6: Steps of the ABC

2.2.4 The Basic Bees Algorithm

The Bees Algorithm is also one of the swarm intelligence-inspired algorithms which was developed by researchers at the Manufacturing Engineering Centre (MEC) in Cardiff University, under the supervision of Prof. D.T. Pham (Pham *et al.*, 2005) after observing bees foraging for nectar.

2.2.4.1 Foraging behaviour of honey bees

A colony of honey bees explores a wide area around their hive to find a food source (nectar). Bees assigned for initial exploration are called scout bees. Scout bees can fly up to 11 km to find better flower patches (Seeley, 1995 and Gould and Gould, 1988). When a scout bee finds a food source its job is to share information regarding the discovered patch with bees waiting in the hive. After delivering nectar to the hive, scouts go to a special area (dance floor) in front of the hive and perform eight shape movements, also known as the 'waggle dance' (Seeley, 1995). The waggle dance contains information about the direction, distance and quality of the flower patch found by the bee (Talbi, 2009). A waggle dance is shown in Figure 2.7**.** The relation between the duration of the dance and distance from hive is given in Figure 2.8 (Seeley *et al.* 2006). Information related to the direction of the source is given in Figure 2.9.

Figure 2.7: Waggle dance of the scout bee.

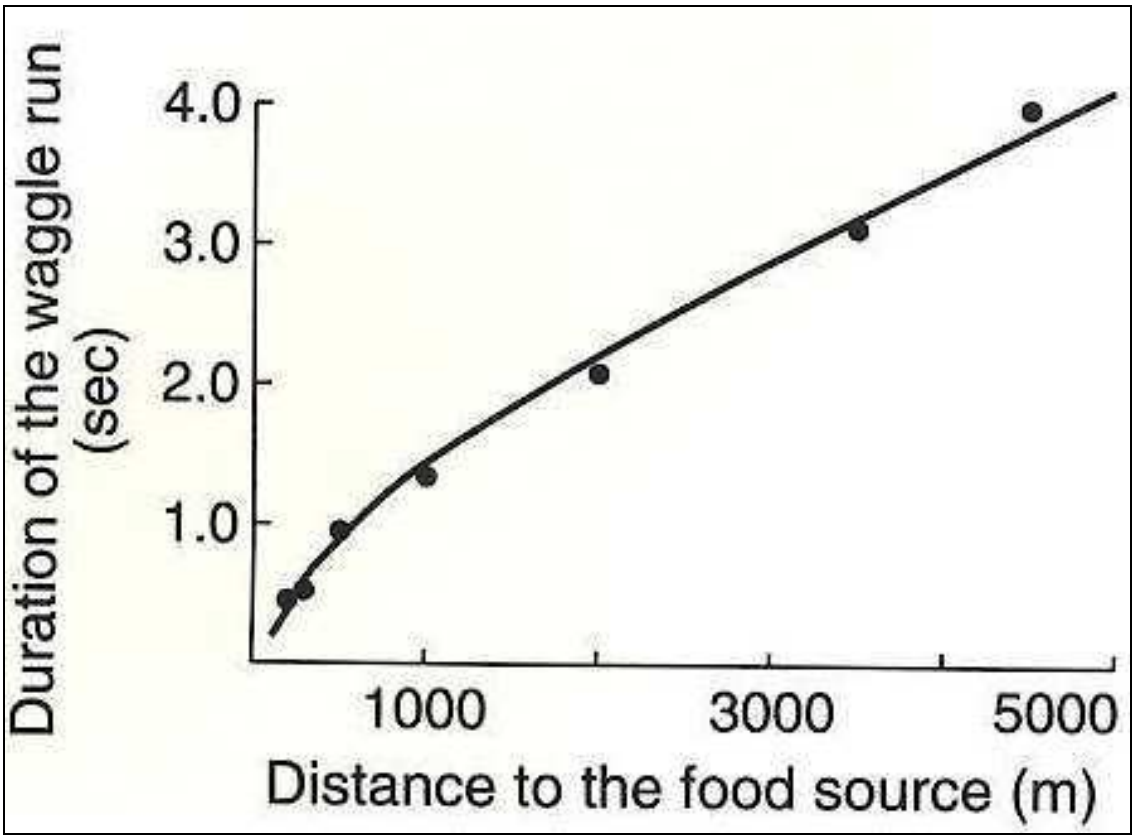

Figure 2.8: **Relation between duration of dance and distance to the food.**

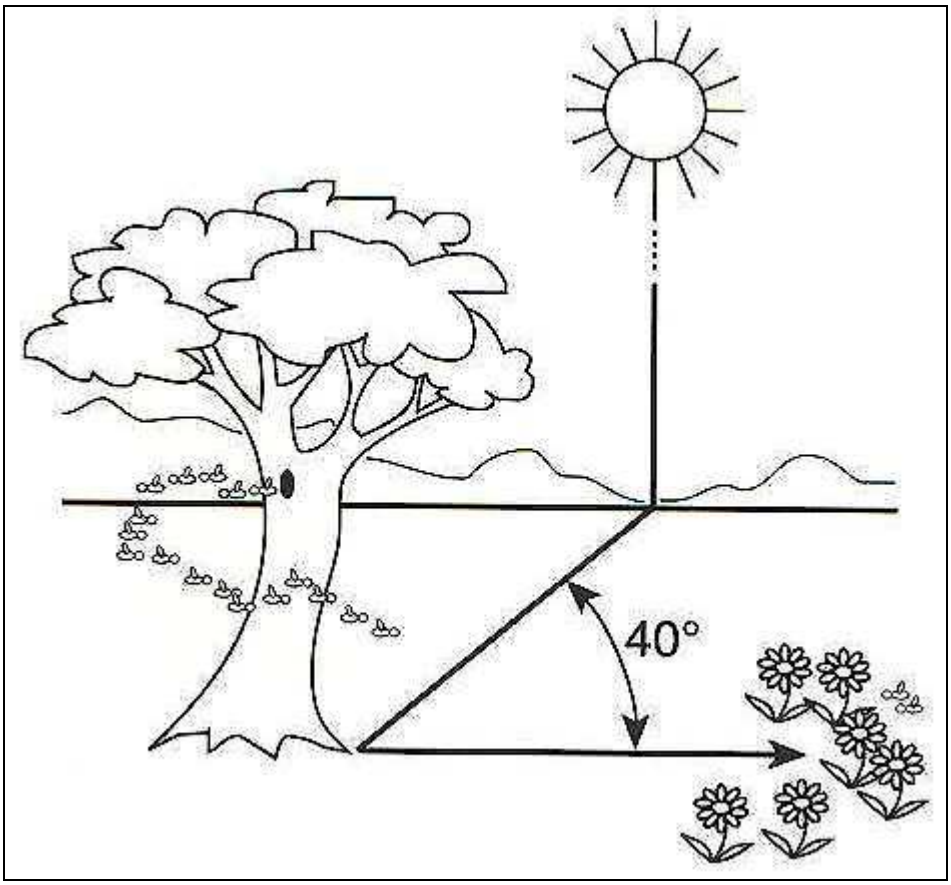

Figure 2.9: Relation between dance and the Direction of the food source

After the performance of the “dancers”, the colony decides the amount of bees that need to be assigned for the food source. More bees go to more promising patches for harvesting. Recruited bees monitor food levels on every patch and share this information with the colony when they go back to the hive. So, bees concentrate on better patches all the time, which makes the food gathering process much faster and more efficient. This behaviour of honey bees was computationally modelled as a search algorithm.

2.2.4.2 The Algorithm

The parameters and the pseudo code of the algorithm are given below.

- Number of scout bees (*n*),
- Number of sites selected out of n visited sites (*m*),
- Number of best sites out of m selected sites (*e*),
- Number of bees recruited for best e sites (*nep*),
- Number of bees recruited for the other (m-e) selected sites (*nsp*),
- Patch size around a selected best location (*ngh*).

Steps of the basic Bees Algorithm are given in Figure 2.10.

Start

Initialise population with random solutions.

Evaluate fitness value of the population.

While (stopping criterion met).

For each best patch

Select the best m patches for neighbourhood search.

Recruit bees for selected patches (more bees for best patches) and evaluate their fitness.

Select the fittest bee value from each patch.

End

Assign remaining bees to search randomly and evaluate their fitness.

End

Figure 2.10: Steps of the basic Bees Algorithm

According to Figure 2.10, the Bees Algorithm has six main steps. The first step is placing the “n” scout bees on a search space. In the following steps, the fitness values of the visited patches are evaluated.

The patches with the highest fitness values are chosen as “selected sites” for neighbourhood search as step 3. In step 4, the algorithm performs a neighbourhood search on selected areas by assigning more scout bees to the elite sites ‘e’, less scout bees to the non elite best sites ‘m-e’. In step 5, the scout bees around the best sites with the highest fitness values are selected as representative bees to form a new population. The remainder of the bees are assigned for random search to find potential solutions in step 6. This process continues until one of the stopping criteria has been met.

2.3 Applications of the Bees Algorithm

The Bees Algorithm was utilised to solve multiple optimisation problems. In this section, examples for applications of the Bees Algorithm are presented.

Continuous type benchmark functions were selected to test the performance of the Bees Algorithm. Optimisation of these functions was the first application of the Bees Algorithm (Pham *et al.*, 2006a). Later, the Bees Algorithm was tested on even more benchmark functions. Results were compared with different optimisation algorithms (Pham and Castellani, 2009a). Results obtained using the Bees Algorithm were better when compared to other algorithms.

The Bees Algorithm was used to optimise the cost of fabrication on a multi-objective welded beam problem by (Ghanbarzadeh, 2007). The goal of the study was to minimise the cost by finding an optimum weld thickness, weld length, beam thickness and beam width under the stress constraints. The Algorithm was also utilised to solve a multi-objective carbon energy system and an environmental dispatch problem (Lee, 2010). The goal was to minimise the total cost and CO emissions for designing a low carbon system.

The Bees Algorithm was also implemented to determine weights for the neural networks such as: Learning Vector Quantisation network (Pham *et al.,* 2006b), Multi Layered Perceptron neural network (Pham *et al.*, 2006c; Koc, 2010), Radial Basis neural network (Pham *et al.*, 2006d). Results showed that the Bees Algorithm is a good classifier and optimisation tool.

The Bees Algorithm was also applied to cellular manufacturing systems to optimise the cell information problem (Pham *et al.*, 2007a). The results obtained proved that the Bees Algorithm is good enough to be used for combinatorial applications.

The Bees Algorithm was tested on the job scheduling problem (Pham *et al.*, 2007b). The performance of the Bees Algorithm was better than that of TS, GA, and PSO on this problem.

Another application of the Bees Algorithm was on clustering problems. The Bees Algorithm was implemented on the K-means and C-means clustering problems (Pham

et al., 2007c; Al-Jabbouli, 2009). The results showed that the Bees Algorithm could be a powerful tool for clustering applications.

Promising results were obtained from a robotic application of the Bees Algorithm which has been proposed (Pham *et al.*, 2008). In this study the Bees Algorithm was used for learning the inverse kinematics of a robot manipulator. The second robotic application of the Bees Algorithm was proposed by (Pham *et al.*, 2009b). The Algorithm was utilised to tune the fuzzy logic controller parameters for stabilising and balancing an acrobatic robot. Experimental results were positive.

Several studies were done to increase the performance of the Bees Algorithm. One of these studies was a hybrid approach where the Bees Algorithm and PSO were combined (Sholedolu, 2009). This combination was done for the Bees Algorithm to benefit from PSO's advantages in an adaptive neighbourhood search. Hybrid PSO-Bees Algorithms results were promising and fast.

Another study done to improve performance of the Bees Algorithm was using algorithm to tune parameters (Otri, 2011). The performance of the Bees Algorithm was improved by this enhancement.

Moreover, the Bees Algorithm was applied on multiobjective Supply chain problem to minimise the total cost and the total lead-time (Ernesto *et al.*, 2013).

2.6 Summary

Different optimisation techniques have been described in this chapter. These techniques were classified based on their variables. The aim of this chapter was to provide background information for the following chapters. Brief information about deterministic optimisation techniques was given. Stochastic optimisation techniques were described in detail. The Bees Algorithm was described in detail, which will be used in Chapters 3, 4 and 5.

In following chapters three modified versions of the Bees Algorithm will be discussed. In chapter 2 the Bees Algorithm with early neighbourhood search and efficiency-based recruitment will be introduced. The following Chapter 4 will be about the Hybrid Tabu Bees Algorithm. Autonomous Bees Algorithm, which is third and last contribution, will be explained in Chapter 5.

CHAPTER 3

The Bees Algorithm with Early Neighbourhood Search and Efficiency-Based Recruitment Strategies

3.1 Preliminaries

In the literature there are several optimisation algorithms with different search abilities and each of them has their own strengths and weaknesses. Considering the Bees Algorithm which is the focus of this study, it has a random initialisation stage. Such initialisation has both advantages and disadvantages. The results produced by the algorithm are subject to this random initialisation process. This can be overcome by starting the search from a more promising location.

This study presents new modifications to the basic Bees Algorithm, which are early neighbourhood search and improved recruitment using an efficiency calculation. The aim of the Early Neighbourhood Search and Efficiency-based Recruitment Bees Algorithm (ENSEBRBA) is to enhance the performance of the initialisation stage and make the neighbourhood search more competitive, which will empower the overall performance of the algorithm on high dimensional problems. The steps and flow chart for proposed version of the Bees Algorithm are given in Figures 3.1 and 3.2.

Start

Initialise population with random solutions.

Do (early neighbourhood search for each random solutions)

Evaluate fitness values of each neighbourhood.

End

While (stopping criterion not met)

For each best patch

Select sites for neighbourhood search.

Recruit bees for selected sites (using normal strategy + efficiency based enhancement) and evaluate fitnesses.

Select the fittest bee from each patch.

End

Assign remaining bees to search randomly and evaluate their fitnesses.

End

Figure 3.1: Steps of the ENSEBRBA

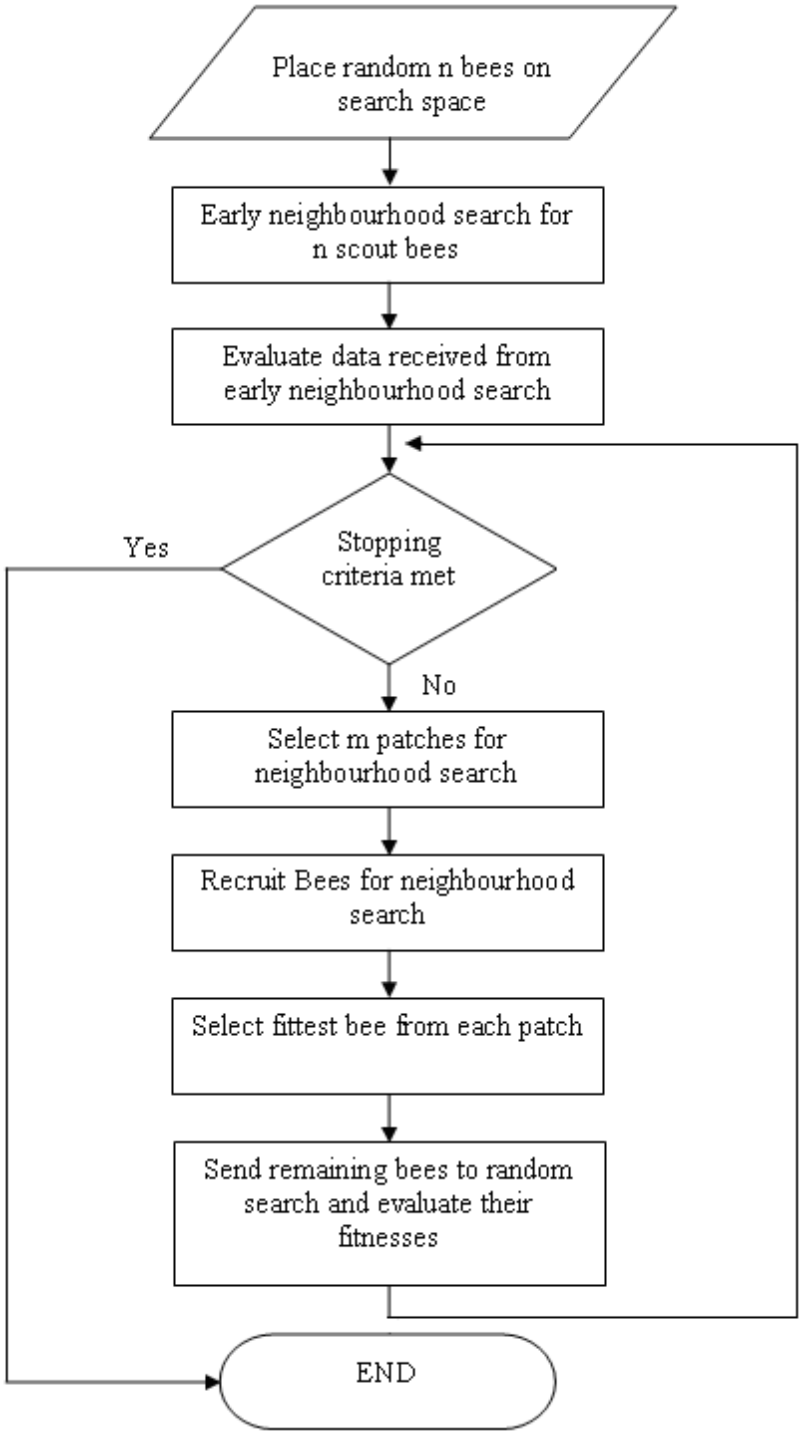

Figure 3.2: The flow chart of the ENSEBRBA

3.2 The Early Neighbourhood Search Strategy

The Early Neighbourhood Search Strategy-based Bees Algorithm starts with a random initialisation of n scout bees on search space, the same as in the Basic Bees Algorithm (BBA). The initialisation stage in BBA considers a list of all random points visited by each scout bee. This may not contain enough information about the space surrounding. To get a better view of the vicinity of the point and to start a neighbourhood search from more promising patches, an early neighbourhood search stage is introduced during the first scouting process. The neighbourhood search is carried out with a minimum number of scout bees in order not to affect the computational time of the algorithm too much by increasing the number of iterations. This leads to the discovery of better fitness valued sites from where the local search will be carried out because if the algorithm starts its search from an advantageous position, it is obvious that it will have better opportunity to converge to the global optimum.

In addition to this, another improvement is proposed in the following section, which is efficiency-based recruitment for each best patch to increase the performance of the algorithm.

3.3 Efficiency-based Recruitment Strategy

Efficiency characterises how well the time, cost and effort used for the implementation of a task or job compares to that achieved by alternative methods. This term has widely varying meanings and applications in different disciplines. In particular for engineering, it can be generalised such that Efficiency is a capability of producing a specific outcome effectively with a minimum amount or quantity of waste, cost, or unnecessary work.

Efficiency can be shown as a percentage of what ideally could be achieved. The efficiency of any work in its simplest form can be calculated with the formula below:

$$Efficiency = \frac{Output}{Input} \times 100 \tag{3.1}$$

Efficiency-based recruitment for neighbourhood search strategy is the second step of the proposed Bees Algorithm. In this stage, the fitness values of each patch are evaluated for choosing “m” best patches to start the neighbourhood search. The neighbourhood search process is performed as in the Basic Bees Algorithm with the addition of efficiency-based recruitment.

The number of recruited bees for the neighbourhood search changes dynamically according to the efficiency of the related sites where the number of bees around elite

(nep) and non-elite best sites (nsp) are computed based on Equations 3.2 and Equation 3.3 respectively.

$$nep_i = nep_{i-1} + nab \tag{3.2}$$

$$nsp_i = nsp_{i-1} + nab \tag{3.3}$$

where "*nab*" is the number of added bees according to the efficiency calculation and "i" is the number of the iteration.

The number of the added bees, nab, is computed based on the Efficiency Rate (ER) of the best sites after a predefined number of iterations β. The '*ER*' for each selected patch is calculated as given in Equation 3.4.

$$Er^j = \frac{\Delta F_i^{\,j} - \Delta F_{\min}}{\Delta F_{\max} - \Delta F_{\min}} \tag{3.4}$$

where "i" is the iteration number, "j" is the site number, $\Delta F_i^{\,j} = F_i^{\,j} - F_{i-\beta}^{\,j}$, $\Delta F_{\min} = \min\{F_i - F_{i-\beta}\}$, $\Delta F_{\max} = \max\{F_i - F_{i-\beta}\}$, Er^j is the efficiency rate of patch "j". Each patch is ranked according to their efficiency rate. So the number of recruited bees around each site changes according the ranks given in Table 3.1.

Range of *ER*	Group Type	Required Bees
0.0 <= *ER* <= 0.2	E	0 Bees
0.2 <= *ER* <= 0.4	D	+1 Bees
0.4 <= *ER* <= 0.6	C	+2 Bees
0.6 <= *ER* <= 0.8	B	+3 Bees
0.8 <= *ER* <= 1.0	A	+4 Bees

Table 3.1: The patch range and required numbers of bees.

Finally, the remaining scout bees are assigned randomly to carry out a global search. The process will run until stopping criteria are met. Stopping criteria for the proposed version of the algorithm are:

- Global optimum found with acceptable error rate (ER) (In this study error rate was chosen as, ER <0.0001).
- Maximum number of the Evaluations.(In this study this value is chosen as, 5000000)
- Number of repetitions of the global optimum.(In this study this value is chosen as, 100)

3.4 Experiments

To measure the performance of the algorithm, some well known continuous type benchmark problems were selected. Each of these functions has different characteristics, so obtained results illustrate strengths and weaknesses of the algorithm in different situations. The Algorithm was run a hundred times for each function. The results were compared with the basic Bees Algorithm (BA) and other well-known optimisation techniques such as Particle Swarm Optimisation (PSO), Evolutionary Algorithm (EA) and Artificial Bee Colony (ABC).

The Bees Algorithm requires a number of parameters to be set manually for each benchmark function. Further, the number of recruited bees for early neighbourhood search and 'β' for efficiency-based recruitment must be predefined in the proposed version of the Bees Algorithm. In this study, the number of recruit bees for early

neighbourhood search and 'β' were defined as 2 and 10, respectively. The other parameters to run the proposed algorithm to solve different benchmark problems are given in Table 3.2 (Ahmed, 2012).

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 3.2: The parameters to run the ENSEBRBA on different benchmark functions (Ahmad, 2012).

All used test functions are described below. The 3 D plots of all used test function can be found in appendix A. (Molga, 2005).

Goldstein-Price's function

The Goldstein-Price function is a two dimensional global optimisation test function which can be defined as following (Molga, 2005):

$$f(x1,x2) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2]\cdot \\ \cdot[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27_2^2)] \quad (3.5)$$

$$-2 \le x_1x_2 \le 2,; \quad f(x) = 3 \quad (x_1, x_2) = (0,-1)$$

Schwefel's function

The Schwefel function has complex geometrical topography, where the local minimuma are far from each other. Thus, search algorithms struggle to converge in the direction of the global minimum. A definition of the function is given below (Molga, 2005):

$$f(\vec{x}) = \sum_{i=1}^{n}[-x_i \sin(\sqrt{|x_i|})] \quad (3.6)$$

$$-500 \le x_i \le 500, \qquad i = 1\ldots\ldots n;$$

$f(\vec{x}) = -418.9829n;$

$x_i = 420.9687, \qquad i = 1\ldots\ldots n;$

Rosenbrock's valley

Rosenbrock's valley is also known as the *banana function* or the *second function of De Jong*. The global optimum for the function is located at the flat valley which has a long narrow parabolic shape. It is simple enough to find the valley. However convergence to the global optimum is difficult. This Function is defined as (Molga, 2005):

$$f(\vec{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \qquad (3.7)$$

$-2.048 \le x_i \le 2.048, \qquad i=1\ldots\ldots n;$

$f(\vec{x}) = 0$

$x_i = 0, \qquad i=1 \ldots\ldots n;$

Hyper sphere function

Hyper sphere is continues type unimodal, curved function, which is also known as the *weighted sphere model*. Function can be defined as (Molga, 2005):

$$f(\vec{x}) = \sum_{i=1}^{n} (i \cdot x_i^2) \qquad (3.8)$$

$-5.12 \le x_i \le 5.12$, $i=1 \ldots\ldots n;$

$f(\vec{x}) = 0$,

$x_i = 0$, $i = 1 \ldots\ldots n;$

Ackley's function

Ackley's is a widely used multimodal test function. This function can be defined as (Molga, 2005):

$$f(\vec{x}) = -a \cdot \exp(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)) + a + \exp(1) \qquad (3.9)$$

$a = 20, b = 0.2, c = 2\pi$

$-32{,}768 \le x_i \le 32{,}768$, $i=1 \ldots\ldots n;$

$f(\vec{x}) = 0$

$x_{ii} = 0$, $i = 1 \ldots\ldots n.$

Rastrigin function

Rastrigin's function is a modified version of the De Jong function. In order to produce numerous local minima with cosine modulation a Rastrigin function was utilised. This addition makes the test function highly multimodal. However, the locations of the minima are regularly distributed. The function has the following definition (Molga, 2005):

$$f(\vec{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \qquad (3.10)$$

$-5.12 \le x_i \le 5.12$ $\quad i=1\ldots\ldots n;$

$f(\vec{x}) = 0$,

$x_i=0,$ $\quad i=1\ldots\ldots n$

Martin & Gaddy

Martin & Gaddy is a widely used multimodal test function. The definition of the test functions is given below (Molga, 2005):

$$f(x_1, x_2) = (x_1 - x_2)^2 + [\frac{(x_1 + x_2 - 10)}{3}]^2 \qquad (3.11)$$

$-20 \le x_1 x_2 \le 20$, $\quad f(x)=0$, $\quad (x_1,x_2)=(5,5).$

Easom's function

The Easom function is a two dimensional, unimodal test function. This function's global optimum has a small area compared to the search space. The definition of the test functions is given below (Molga, 2005):

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) \qquad (3.12)$$

$-100 \le x_1 x_2 \le 100$, *f(x) =-1,* $(x_1, x_2) = (\pi, \pi)$.

Griewangk's function

Griewangk's function is similar to the function of Rastrigin, where local minima are widely spread using regular distribution. The definition of the test function is given below (Molga, 2005):

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1 \qquad (3.13)$$

$-600 \le x_i \le 600$, *i=1……..n;*
$f(\vec{x}) = 0$,
x_i*=0.* *i=1……..n.*

Benchmark functions can be used to represent key features of the real world problems. Some examples for the benchmark functions representing manufacturing problems are given below.

For example, Dynamic motion problems found in physics and manufacturing can be described as 3^{rd}, 4^{th} and 5^{th} degree polynomial functions (Klipp 2001). Goldstein Price is second degree polynomial problem (Goldstein, Price 1971). Therefore dynamic motion problems can be defined with modified Goldstein and Price benchmark function.

Another example is the representation of surfaces in atomic level by benchmark functions. Atomic force microscope (AFM) is used to analyse surfaces of the materials down to atomic level and can produce 3D topography of surface. It is possible to use Rastrigin, Schwefel, Schaffer and Ackley functions to represent the surface features of the materials. “Thus, they have the strength of an analytical expression with a known global minimum and they are extendable to arbitrary dimensionality allowing for scaling investigations on global structure optimization of atomic and molecular clusters” (Dieterich, Hartke 2012).

Furthermore, the Cost curve in engineering economy (Mishra 2009) can be represented by Hyper Sphere benchmark functions. Moreover for cost minimisation, Rosenbrock function was suggested to be utilised by Rosenbrock (Rosenbrock 1960).

Cases given above can also be extended for other benchmark functions.

3.5 Results and Discussion

The performance of the proposed algorithm was assessed according to the accuracy and the average evaluation numbers and results were compared to well known optimisation techniques. These are given in Tables 3.3 and 3.4. Experimental results for PSO, EA and ABC were extracted from Ahmad (2012).

The accuracy of algorithms was computed based on average absolute differences of the best results of a hundred runs. According to this approach, the more accurate results are closer to zero.

Goldstein-Price 2D: Expected optimum result for the function is 3. Average result obtained from the Basic Bees algorithms for a hundred runs was found to be 3.0005. The result received from ENSEBRBA on the same problem was found to be 3.0007. The BBA used an average of 504 evaluation numbers to find that result, where average of new algorithms evaluation numbers was 21.496. Both algorithms produced similar average global optima. However performance of the BA was not improved on given problems by applying presented enhancements. Thus average number of evaluations used by the ENSEBRBA was significantly more than number of evaluations used by the BBA. Figure 3.3 illustrates global optima for a hundred runs of BBA and ENSEBRBA on a given problem.

Schwefel 2D: The expected optimum for the function is -837.97. The average global optimum obtained from a hundred runs of the Basic Bees algorithm and ENSEBRBA were -837.144 and -837.964 respectively. BBA used an average of 250049 evaluation numbers to find that optimum, whereas the average for the new algorithm's evaluation was 338.600. As mentioned earlier this test function has complex topography so it is hard to converge to the global optimum but the Bees algorithm with both global and local search found the optimum with no problem. However, ENSEBRBA with early neighbourhood search and enhanced local search was more accurate on the given task. Average global optima for a hundred runs of BBA and ENSEBRBA on the given problem are shown in Figure 3.4.

Schaffer 2D: The expected optimum for the function is 0. Averages of a hundred global optimums of the Basic Bees algorithm was 0,01. The corresponding result obtained from the presented version of the Bees algorithm (a hundred runs) was 0,001. BBA used an average of 121.088 evaluation numbers to find that result, whereas the average for the new algorithm's evaluation was 112.430. The ENSEBRBA performed more accurately and faster than the BBA on this optimisation problem. Figure 3.5 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

Rosenbrock 10 D: The expected answer is 0. The average global optimum obtained from the Basic Bees algorithms (a hundred runs) was 0.0003. The result received from ENSEBRBA was 0.0002. BBA used an average of 116904 evaluations to find that result, whereas the average for the new algorithm was 148193.The BBA found less

accurate global optimum when the presented version of the BA produced more accurate result. The performance of algorithm for given problem was increased. This is related to the extra initialisation during the early phases of the proposed algorithm. Figure 3.6 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

No.	Functions	PSO		EA		ABC		BA		ENSEBRBA	
		Average Absolute Difference	Standard. Deviation.	Average Absolute Difference	Standard. Deviation.	Average Absolute Difference	Standard. Deviation.	Average Absolute Difference	Standard. Deviation.	Average Absolute Difference	Standard. Deviation.
1	Goldstein & Price (2D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0005	0.0006	0.0007	0.0008
2	Schwefel (2D)	4.7376	23.4448	4.7379	23.4448	0.0000	0.0000	0.1492	0.7679	0.0004	0.0057
3	Schaffer (2D)	0.0000	0.0000	0.0009	0.0025	0.0000	0.0000	0.0096	0.0018	0.0009	0.0029
4	Rosenbrock (10D)	0.5998	1.0436	61.5213	132.6307	0.0965	0.0880	0.0003	0.0003	0.0002	0.0003
5	Sphere (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0001	0.0001
6	Ackley (10D)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0,0294	0,0477	0.0001	0.0028
7	Rastrigin (10D)	0.1990	0.4924	2.9616	1.4881	0.0000	0.0000	0.005	0.02	0.0002	0.0003
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	0.3	0.23	0.0000	0.0003
10	Griewank (10D)	0.0008	0.0026	0.0210	0.0130	0.0052	0.0078	0.3158	0.1786	0.0049	0.0019

Table 3.3: Accuracy of proposed algorithm compared with other well known optimisation techniques.

No.	Functions	PSO		EA		ABC		BA		ENSEBRBA	
		Average evaluations	Standard Deviation.	Avg. evaluations	Standard Deviation.	Avg. evaluations	Standard Deviation.	Avg. evaluations	Standard Deviation.	Avg. evaluations	Standard Deviation.
1	Goldstein & Price (2D)	3262	822	2002	390	2082	435	504	211	21496	36855
2	Schwefel (2D)	84572	90373	298058	149638	4750	1197	250049	680	338600	0
3	Schaffer (2D)	28072	21717	219376	183373	21156	13714	121088	174779	112430	66120
4	Rosenbrock (10D)	492912	29381	500000	0	497728	16065	935000	0	148193	116904
5	Sphere (10D)	171754	7732	36376	2736	13114	480	285039	277778	95643.5	89997
6	Ackley (10D)	236562	9,119	50344	3949	18664	627	910000	0	236299	123325
7	Rastrigin (10D)	412,440	67,814	500,000	0	207,486	57,568	885,000	0	53935	44779
8	Martin & Gaddy (2D)	1778	612	1512	385	1498	329	600	259	15,888	16554
9	Easom (2D)	16124	15942	36440	28121	1542	201	5280	6303	1120	1,345
10	Griewank (10D)	290466	74501	490792	65110	357438	149129	4300000	0	316443	97830

Table 3.4: Average evaluation of proposed algorithm compared with other well-known optimisation techniques.

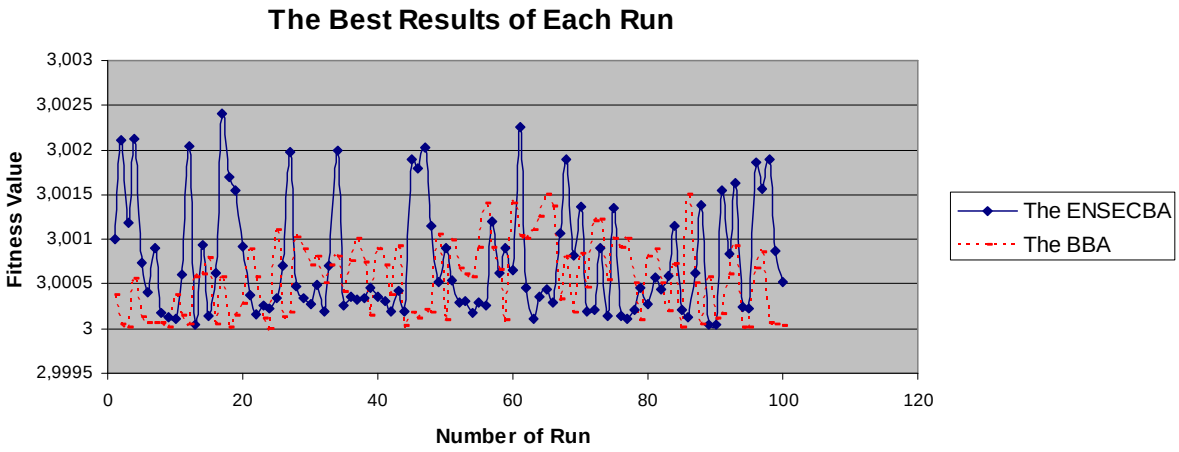

Figure 3.3: The results of a hundred runs for the BBA and the ENSEBRBA on Goldstein & Price (2D).

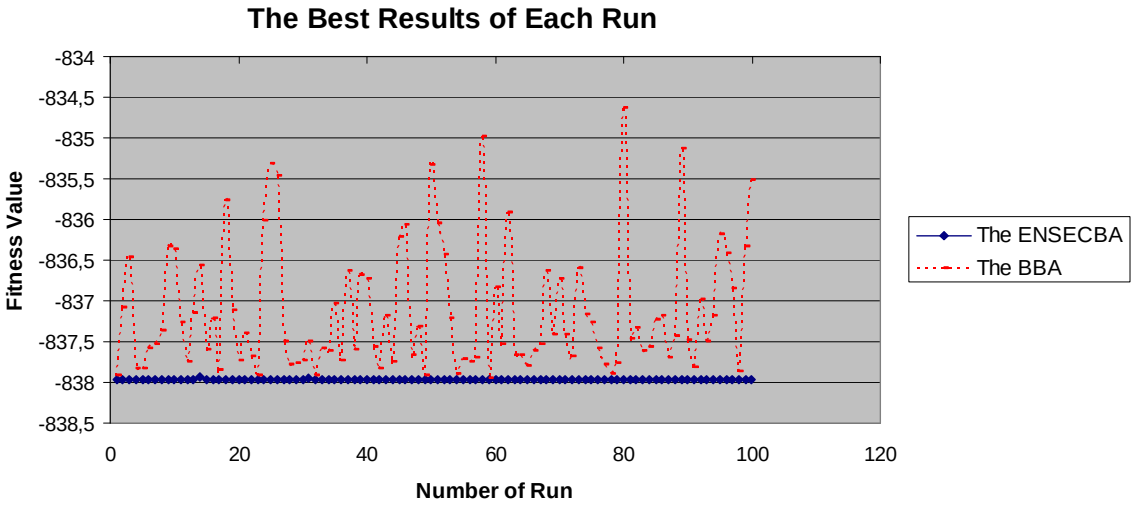

Figure 3.4: The results of a hundred runs for the BBA and the ENSEBRBA on Schewel 2D.

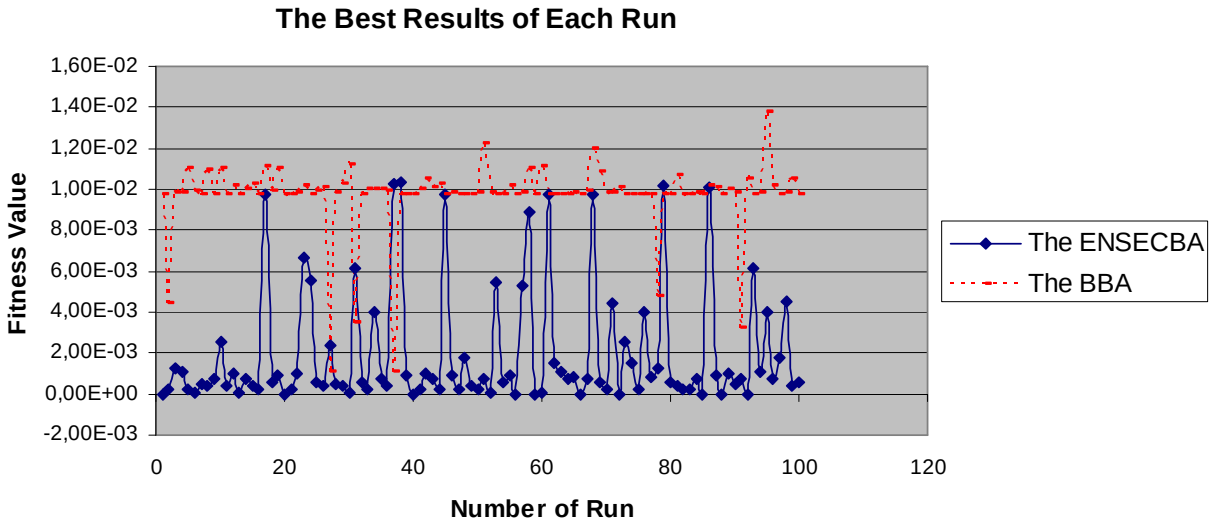

Figure 3.5: The results of a hundred runs for the BBA and the ENSEBRBA on Schaffer 2D.

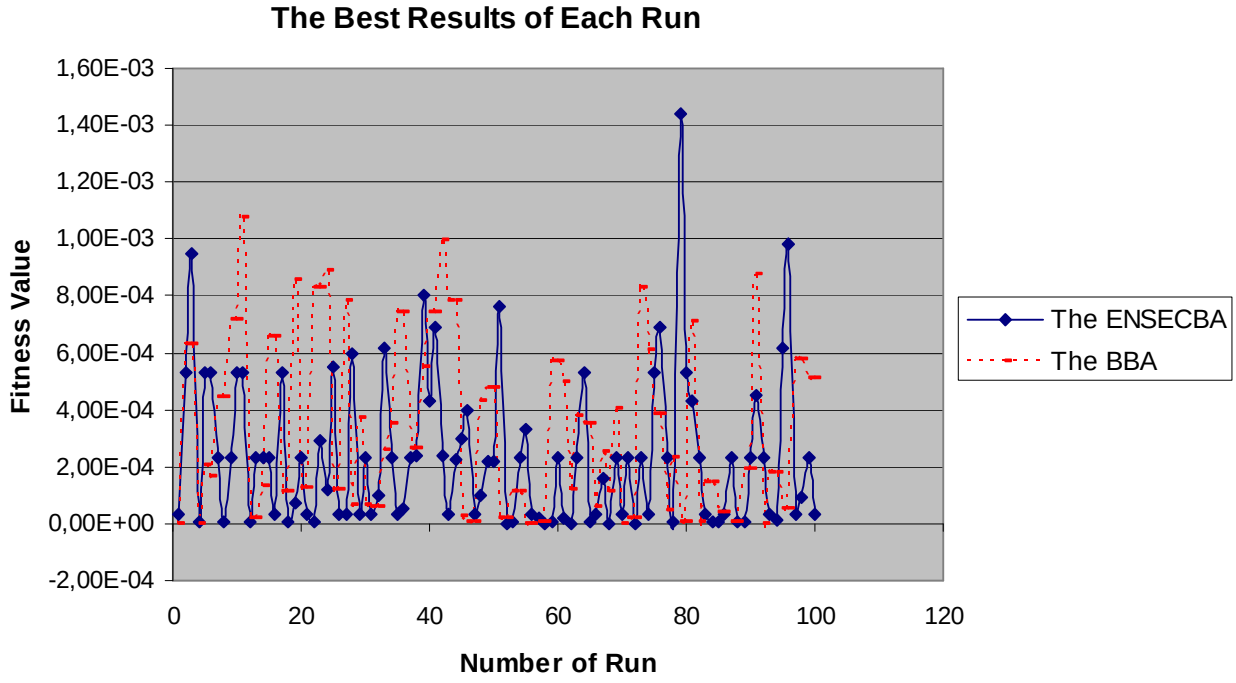

Figure 3.6: The result of a hundred runs for the BBA and the ENSEBRBA on Rosenbrock 10 D.

Hyper Sphere 10D: The expected optimum result for this function is 0. The average result obtained from a hundred runs of The Basic Bees algorithm on the given function was 0.0003. The corresponding result obtained by the presented version of The Bees algorithms for 100 runs was 0.0001. BBA used an average of 285039 evaluations to find that result, whereas the average of the proposed algorithm's evaluation numbers was 95643. The performance of the Bees algorithm was increased significantly for the given function. Figure 3.7 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

Ackley 10D: The expected global optimum for the function is 0. The average optimum result obtained from 100 runs of The Basic Bees algorithms on the given function was 0.029. The corresponding result received from the presented version of The Bees algorithm was 0.0001. BBA used an average of 910000 evaluations to find that result, whereas the average evaluations needed by the new algorithm was 236299. According to the experimental results, ENSEBRBA performance on the Ackley function was significantly better than that of the BBA. Again, it is because of the complex search space of the function that makes algorithms hard to converge to the global optimum with the standard approach. Thus, introduced enhancements empowered the Bees Algorithm to find a more accurate solution. Figure 3.8 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

Rastrigin 10D: The global optimum of the function is 0. The average optimum obtained from The Basic Bees algorithms for a hundred runs was 0.005 and BBA used an average of 885000 evaluations to find that value. The corresponding result received

from the presented version of The Bees algorithm was more accurate (0.0002), and the average of new algorithm's evaluations was only 53935. The Rastrigin function is highly multimodal, which makes it very hard for global optimisation algorithms to find an optimum. Even the BBA, with local and global search strategies, was not very accurate on the Rastrigin function. However the ENSEBRBA with improved local (efficiency-based recruitment) and global search (early neighbourhood search) strategies was successful on this problem. Figure 3.9 illustrates optima of a hundred runs for BBA and ENSEBRBA on the given problem.

Martin & Gaddy 2D: The expected optimum for this function is 0 .Experimental results obtained from The Basic Bees algorithms (a hundred runs) was 0,000 with 600 evaluations.. ENSEBBA found the same result, however the number of evaluations was too high (15,887). This is related to the structure of the new algorithm because it is not necessary to do extra calculations (efficiency rate, early neighbourhood search) for such "easy" functions where it will only increase number of evaluations. Figure 3.10 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

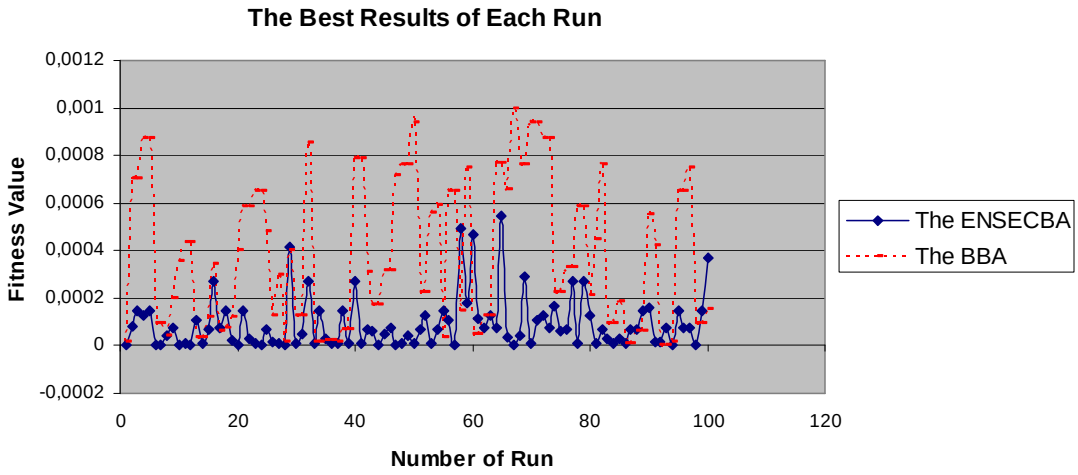

Figure3.7: The results of a hundred runs for the BBA and the ENSEBRBA on Hyper sphere 10D.

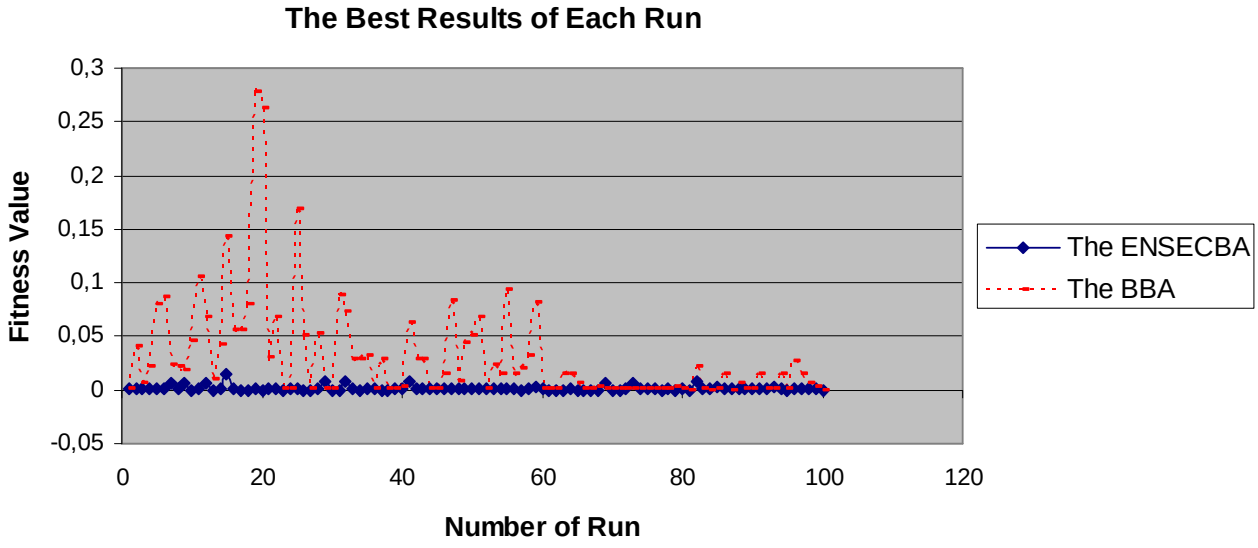

Figure3.8: The results of a hundred runs for the BBA and the ENSEBRBA on Ackley 10D.

Easom 2D: The expected optimum result for this function is -1. BBA performance on the given function was not satisfactory. An average of hundred results obtained from The Basic Bees algorithm was -0.707 while BBA used an average of 5280 evaluations. On the other hand, results obtained from ENSEBRBA were better. Respective results received from ENSEBRBA were -0.9999 and 1120 (evaluations). Easom is another hard optimisation problem. The ENSEBRBA performed better than the BBA on this function. Figure 3.11 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

Inverted Griewank 10D: The expected global optimum for this function is 10. The average result obtained from the Basic Bees Algorithm for a hundred runs was 9.989. The corresponding result received from the presented version of The Bees Algorithm was 9.990. BBA used an average of 4300000 evaluations to find that result, whereas the average of the new algorithm's evaluation numbers was 316443. Experimental results obtained from the BBA on the given function were not satisfactory. Although, the average global optimum was close to the expected one, the number of evaluations to get that result was extremely high. However, the ENSEBRBA performed significantly well on the Griewank function. Both average optimum and number of evaluations for the proposed Bees Algorithm were better than the BBA's corresponding results. The proposed version of algorithm's number evaluations is still too high. Figure 3.12 illustrates global optima for a hundred runs of BBA and ENSEBRBA on the given problem.

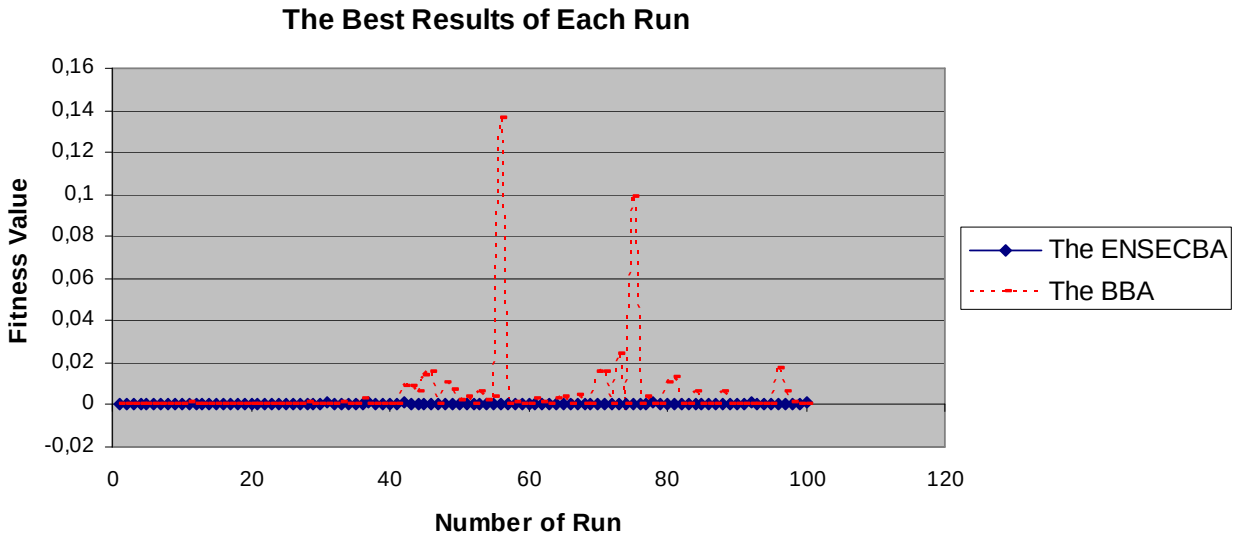

Figure 3.9: The results of a hundred runs for the BBA and the ENSEBRBA on Rastrigin 10D.

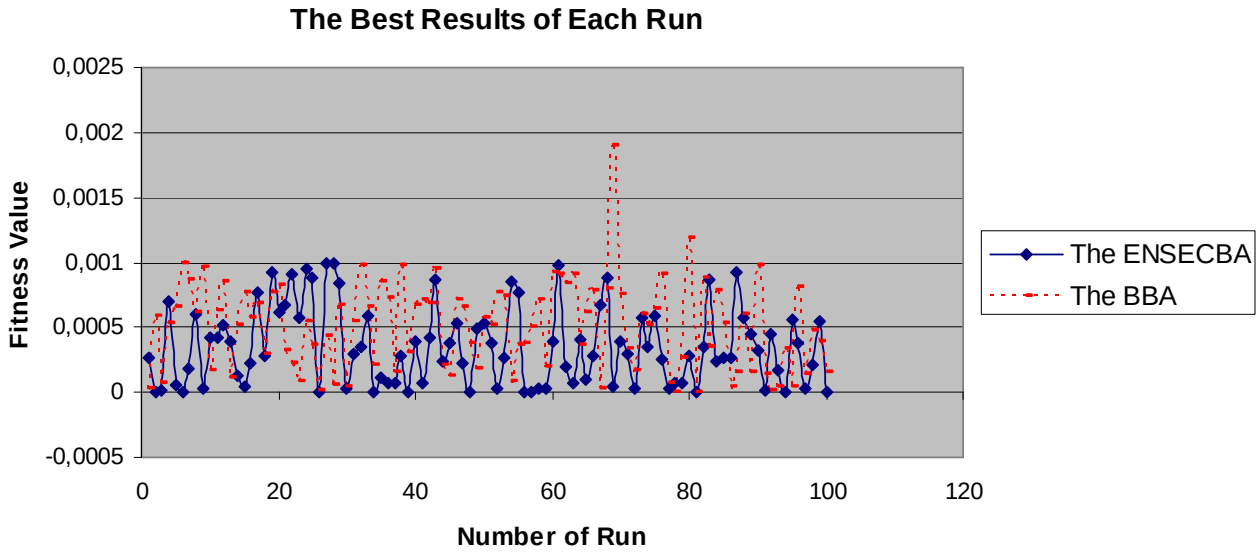

Figure 3.10: The results of hundred runs for the BBA and the ENSEBRBA on Martin & Gaddy 2D.

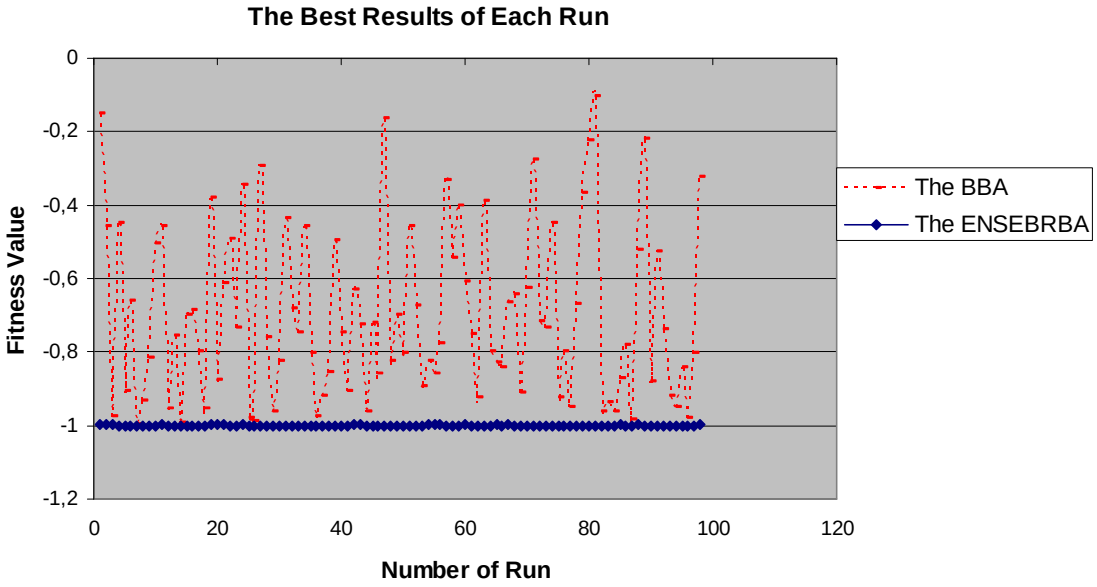

Figure 3.11: The results of hundred runs for the BBA and the ENSEBRBA on Easom 2D.

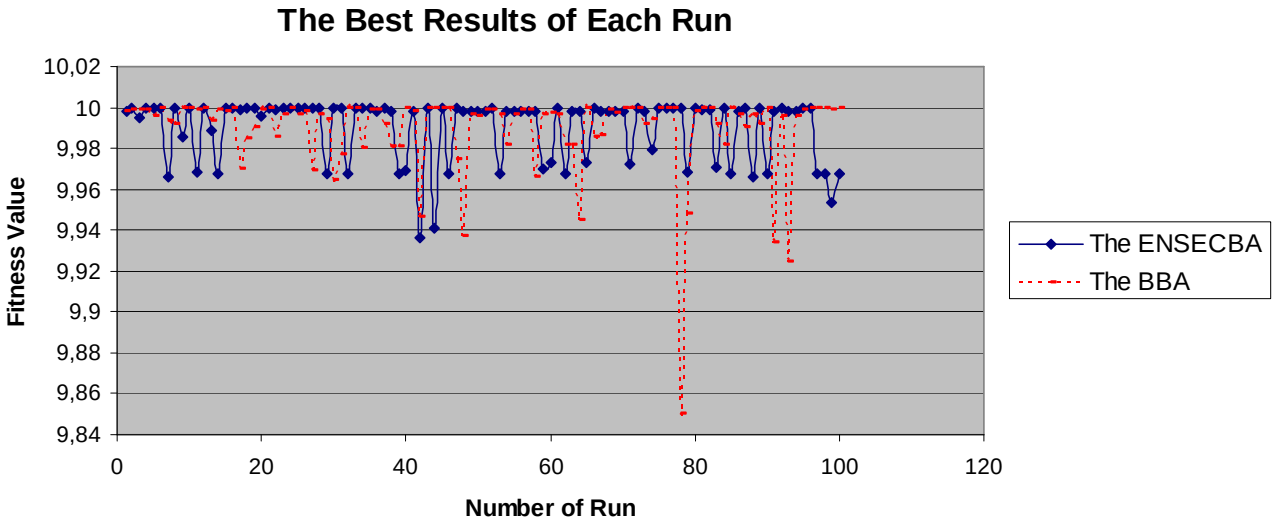

Figure 3.12: The results of a hundred runs for the BBA and the ENSEBRBA on inverted Griewank 10D.

A further t-test was utilised to do statistical analysis of the algorithm where the confidence level was selected to be 95 % ($\alpha < 0.05$). Based on observed results (Table 3.5), the proposed algorithm is statistically significantly better than the Basic Bees on all benchmark functions.

Overall results illustrate that the ENSEBRBA's performance on complex high dimensional functions is better than on lower dimensional ones. Although the proposed algorithm finds an accurate global optimum, the number of evaluations needed to get that result is relatively high. This is due to extra computation performed in the proposed version of the algorithm.

According to the 'no free lunch' theorem, if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems (Wolpert and Macready, 1997).

No.	Function	Significance between the basic Bees Algorithm and the improved Bees Algorithm	
		Significant ($\alpha<0.05$)	α
1	Goldstein & Price (2D)	Yes	0.0004
2	Schwefel (2D)	Yes	3.698 E-18
3	Schaffer (2D)	Yes	6.472 E-52
4	Rosenbrock (10D)	Yes	0.0045
5	Sphere (10D)	Yes	1.9650 E-06
6	Ackley (10D)	Yes	7.150 E-08
7	Rastrigin (10D)	Yes	0.0085
8	Martin & Gaddy (2D)	Yes	0.0010
9	Easom (2D)	Yes	0.0024
10	Griewank (10D)	Yes	0.019

Table 3.5: The statistical analysis between the proposed Bees Algorithm and the basic Bees Algorithm.

3.6 Summary

In this study, two novel enhancements have been presented for the Bees Algorithm. The Basic Bees algorithm was improved both with the early neighbourhood search in the initialisation stage and efficiency-based recruitment in the neighbourhood search stage. The proposed algorithm has been successfully applied to continuous type benchmark functions and compared with other well-known optimisation techniques.

To test the performance of proposed algorithm, the following approaches have been utilised; accuracy analysis, average evaluation and t-test.

According to the accuracy analysis and the average evaluation, the proposed algorithm performed better on higher dimensional than lower dimensional functions. Finally, the statistical significance of the proposed algorithm has been computed with a t-test and the results were compared with the basic Bees Algorithm. Based on t-test results, it can be claimed that the proposed algorithm is statistically significantly better in performance than the basic Bees Algorithm.

Chapter 4

Tabu Bees Algorithm

4.1 Preliminaries

In earlier chapters it was mentioned that the Bees Algorithm has both local and global searches. Global search of the Basic Bees Algorithm considers random exploration of the search space. Because of this random behaviour, the algorithm is unable to avoid visiting already visited sites in order to carry out a local search. Eventually the algorithm converges to the global optimum at the expense of the number of evaluations.

To overcome this site repetition problem, a new algorithm is proposed which is a hybrid of BBA and Tabu Search. The new algorithm is called the Tabu Bees Algorithm (TBA). In TBA, the tabu list was adopted to provide memory to the BBA, memorising unproductive sites and not visiting them again. This shrinks the search space and decreases the number of evaluations needed.

Moreover, a new escape strategy for neighbourhood search is proposed to lead the algorithm out of patches where the fitness values are too similar, due to the Bees Algorithm's nature of getting stuck around the local optima..

The steps and the flowchart for the proposed algorithm are given in Figures 4.1 and 4.2

Start

Initialise population with random solutions.

Evaluate fitness values of each neighbourhood.

While (stopping criterion not met)

Define tabu list

Do for each best patch

Select sites for neighbourhood search.

Recruit bees for selected sites

If Neighbourhood search gets stuck around one point

Use escape strategy

End

Evaluate fitness values from neighbourhood areas

Select the fittest bee from each patch.

End

Update tabu list

Assign remaining bees to search randomly and evaluate their fitnesses.

End

Figure 4.1: Steps of the TBA.

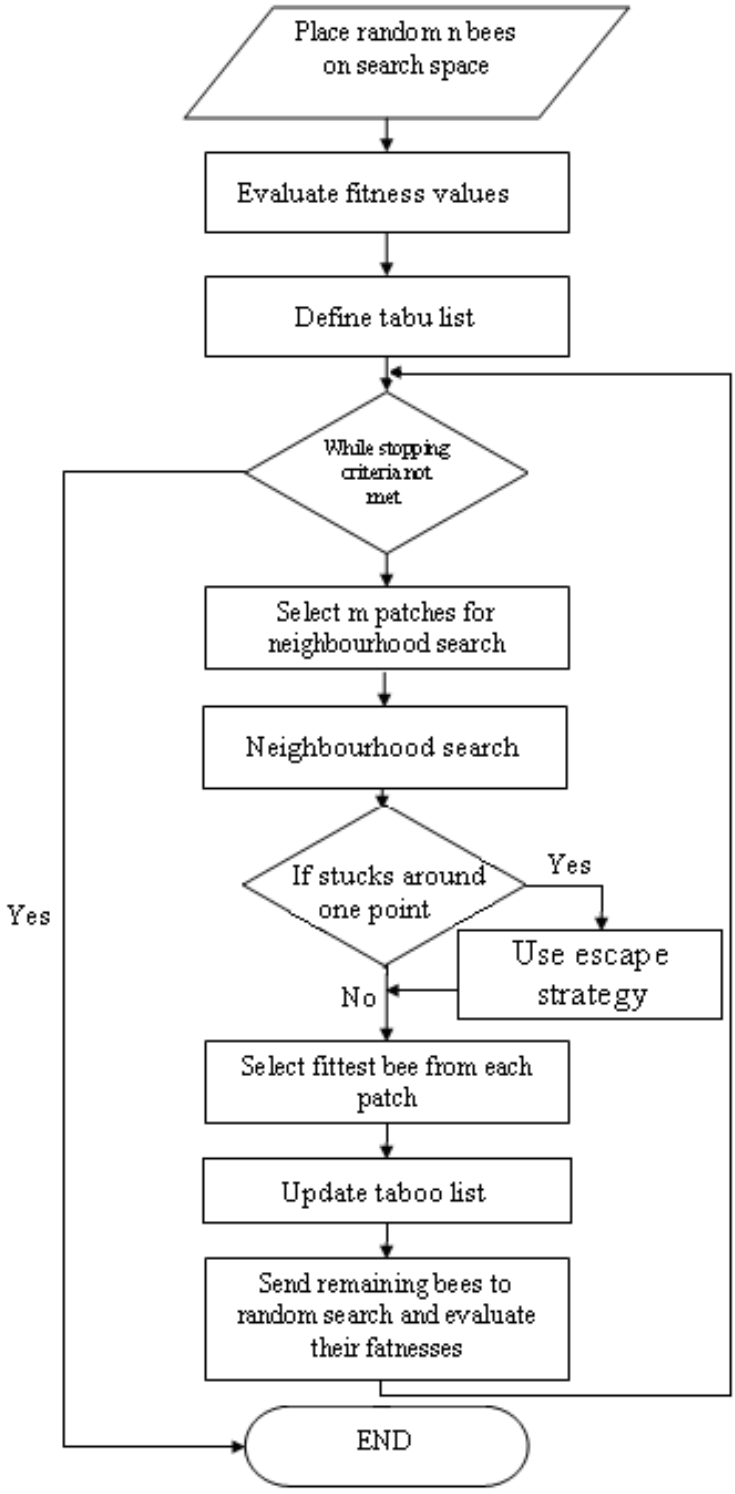

Figure 4.2: Flowchart of the TBA.

4.2 Defining Tabu List

Tabu list is a list of not satisfactory or previously visited solutions which helps an algorithm to avoid those solutions in order to improve its performance. The length of tabu list strongly affects the computational time of the algorithm. Thus, the new solution needs to be verified from the recorded (memorised) tabu list.

To avoid this problem, the length of the tabu list will be updated in every iteration due to having limited size (Rothlauf, 2011).

There are three main strategies to create a tabu list (Pham and Karaboga, 2000), given below:

1. Forbidding strategy: to control new elements entering the existing list.
2. Freeing strategy: to control what exits the tabu list and when.
3. Short-term strategy: to determine a hybrid strategy of forbidding and freeing strategies.

In this study, a short-term strategy-based approach was utilised to create the tabu list.

Tabu Bees Algorithm (TBA) starts the search by randomly placing scout bees in the search space. Then, the fitness values of each allocated site will be evaluated. The next stage is to define the sites for the local and the global search. The local search process will be carried out on “m” best patches. A certain percentage of the “n-m” patches will be utilised for the global search and the rest will be recorded to the tabu

list. These selected patches are the worst patches (w) among the “n-m” sites. The size of the tabu list will be determined empirically. In this study, the tabu list size “t” was determined as equal to the number of scout bees “n”. The next stage of the process is to undertake the neighbourhood search based on an adaptive approach. This will be given in the next section.

4.3 Escape Strategy

There are some problems with a complex search space, where fitness values are too close to each other or even have same value. In this case, the Bees Algorithm performs remarkably slowly during neighbourhood search. It may not be able to escape from local search either. Let us call sites with close fitness values ‘plain’ areas. To escape from ‘plain’ areas an adaptive neighbourhood search is presented. The algorithm tracks the improvement ratio (IR) of fitness values on elite patches to detect ‘plain’ areas. Value of IR must be lower than 0.0001 for the patch to be marked as a ‘plain’ area

$$IR = fitness_{i+1} - fitness_i \qquad (4.1)$$

If no ‘plain’ areas are found, the neighbourhood search process is carried out as for the basic Bees Algorithm. However, if the algorithm detects ‘plain’ areas, it changes its behaviour and shifts the neighbourhood search area into two directions in order to escape. The new neighbourhood search areas are called test neighbourhoods (tnbh). The size of the shifting is ngh/2, which means that the central points for test neighbourhoods areas will be the borders of the initial one (Figure 4.3).

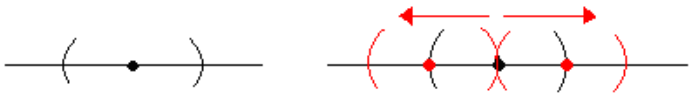

Figure 4.3: Simple example for shifting neighbourhood area.

After evaluating the fitness values of the test neighbourhoods, the algorithm decides on the direction of the search by choosing a more promising 'test neighbourhood' as the actual neighbourhood area. There can be several outcomes of the search on test neighbourhoods as follows:

- Fitness values are not improving for both test neighbourhood areas (Figure 4.4a). In this case, the algorithm shifts test neighbourhood areas further. Shifting is carried out three times and if no improved solutions are found, that patch is added to the tabu list.
- Fitness values are degrading for both test neighbourhood areas (Figure 4.4b). The algorithm adds both patches to the tabu list.
- Fitness values are improving on one of the test neighbourhood areas (Figure 4.4c). The algorithm adds the better site to the tabu list and continues neighbourhood search from the patch where better fitness values were found.
- Fitness values are improving on both test neighbourhood areas (Figure 4.4d). The algorithm compares obtained fitness values. The test neighbourhood with the better fitness is selected in which to continue a neighbourhood search. The worse one will be considered as one of 'm'.

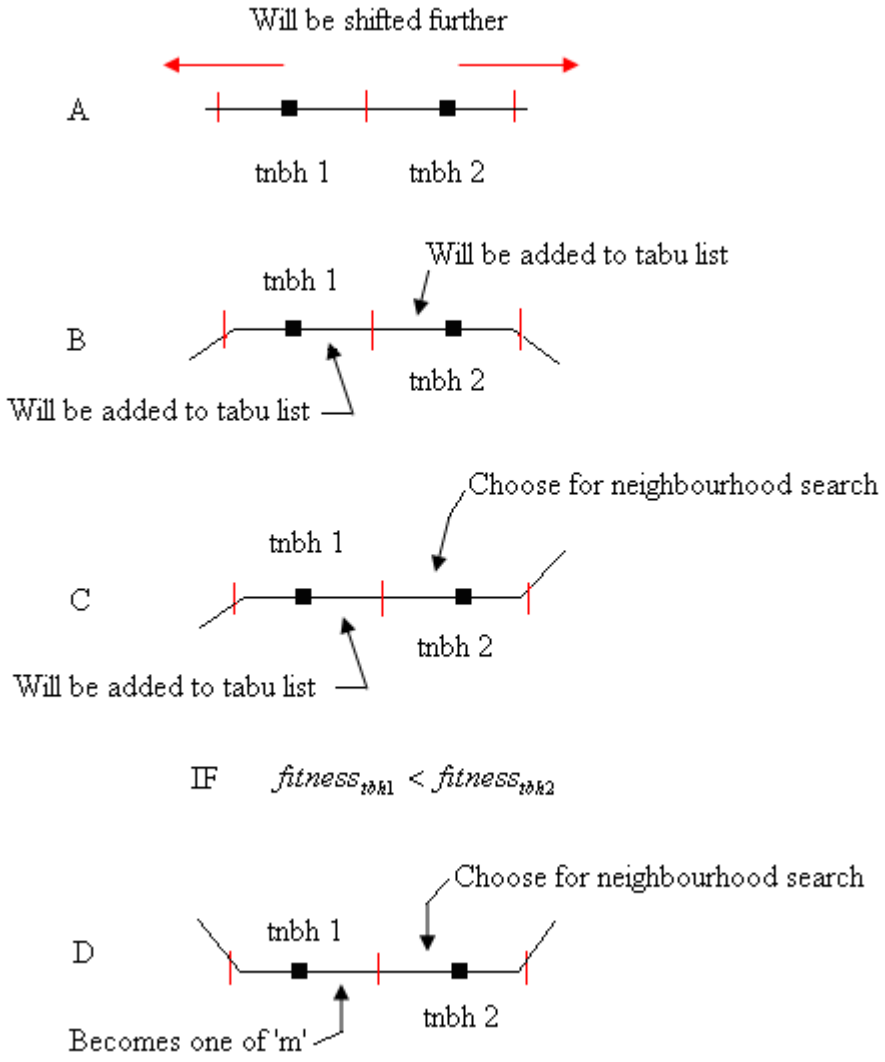

Figure 4.4: Simple example for possible outcomes from 'test neighbourhood' areas.

Further, when neighbourhood search is finished, the TBA evaluates results and updates the tabu list.

4.4 Updating the tabu list

After every neighbourhood search process, the Tabu list is updated using a ‘first in last out’ strategy. New elements enter to the list from the top. As a result, elements which were already in the list move down. One or more elements drop out from the list if there is no space for new incoming data. Algorithm continues search process until one of stopping criteria not met.

Stopping criteria for the proposed version of the algorithm are:

- Global optimum found with acceptable error rate (ER) (In this study error rate was chosen as, ER <0.0001).
- Maximum number of the Evaluations.(In this study this value is chosen as, 5000000)
- Number of repetitions of the global optimum.(In this study this value is chosen as, 100)

4.5 Experiments

To measure the performance of the proposed algorithm, it was tested on ten continuous type benchmark functions. These functions are given in Table 4.1 (Pham and Castellani, 2009 and Ahmad, 2012). Brief information about the used test functions was given in previous chapter.

In previous chapters it was mentioned that the Bees Algorithm requires some parameters to be tuned manually for each optimisation problem. Parameters used for the TBA are given in table 4.2 (Ahmed, 2012). Moreover, value of “w” was chosen empirically to be (n-m)/5 and value “t” was equal to “n” in this study.

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]	$f(\vec{x})=\frac{1}{4000}\sum_{i=0}^{10}(x_i-100)^2-\prod_{i=0}^{i=10}\cos\left(\frac{x_i-100}{\sqrt{i+1}}\right)+1$	$\vec{x}=(\overrightarrow{100})$ $f(\vec{x})=0$

Table 4.1: Test functions (Pham and Castellani, 2009and Ahmad, 2012).

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 4.2: Parameters used for the TBA (Ahmad, 2012).

4.6 Results and Discussion

The performance of the algorithm was assessed as defined in previous chapter, which will be based on the accuracy and the average evaluation numbers (Tables 4.3-4.4). Results were compared to BBA and ENSEBBA.

The experimental results for each function are given below.

Goldstein-Price 2D: The expected global optimum is 3. According to the experiments, the computed results were obtained by evaluation of a hundred runs. The BBA and the ENSEBRBA found the average global optimum at 3.0005 and 3.0007, respectively. The average evaluation numbers found for each algorithm were 504 with BBA and 21496 with ENSEBRBA. The TBA is utilised to solve this benchmark function and the average global optimum was found as 3.0002 in 761 evaluations. The results of a hundred runs for BBA, ENSEBRBA and TBA are given in Figure 4.5. Although the number evaluation of the proposed algorithm is better than the ENSEBRBA, the performance of the BBA is better. Moreover, the proposed algorithm found the global optimum better than all others. According to this comparison, the proposed algorithm performed better than all.

No.	Functions	ENSEBRBA		BA		TBA	
		Average Absolute Difference	Standard Deviation.	Average Absolute Difference	Standard Deviation.	Average Absolute Difference	Standard Deviation.
1	Goldstein & Price (2D)	0.0007	0.0008	0.0005	0.0006	0.0002	0.0002
2	Schwefel (2D)	0.0004	0.0057	0.1500	0.7679	0.0004	0.0212
3	Schaffer (2D)	0.0009	0.0029	0.0096	0.0018	0.0000	0.0000
4	Rosenbrock (10D)	0.0002	0.0003	0.0003	0.0003	0.0000	0.0000
5	Sphere(10D)	0.0001	0.0003	0.0003	0.0003	0.0001	0.0001
6	Ackley (10D)	0.0001	0.0028	0.0294	0.0477	0.0001	0.0003
7	Rastrigin (10D)	0.0003	0.0003	24.8499	8.3306	0.0000	0.0000
8	Martin & Gaddy (2D)	0.0000	0.0003	0.0000	0.0003	0.0000	0.0000
9	Easom (2D)	0.3	0.23	0.0003	0.0003	0.0000	0.0002
10	Griewank (10D)	0.0049	0.0019	0.3158	0.1786	0.0000	0.0001

Table 4.3: Accuracy of proposed algorithm compared with the BBA and the ENSEBRBA.

No.	Functions	ENSEBBA		BA		TBA	
		Average Evaluations	Standard Deviation.	Average Evaluations	Standard Deviation.	Average Evaluations	Standard Deviation.
1	Goldstein & Price (2D)	219496	36855	504	211	761	330
2	Schwefel (2D)	338600	0	250049	0	62054	0
3	Schaffer (2D)	112430	66120	121088	174779	6.309	2165
4	Rosenbrock (10D)	148193	116904	935000	0	9821	3333
5	Sphere (10D)	95644	89997	285,039	277,778	2,972	1,963
6	Ackley (10D)	236299	123325	910000	0	9199	3651
7	Rastrigin (10D)	53935	44779	885000	0	7559	7093
8	Martin & Gaddy (2D)	15888	16554	600	259	1065	1.517
9	Easom (2D)	1120	1345	5280	6303	1063	1130
10	Griewank (10D)	316443	97830	4300000	0	8294	2586

Table 4.4: Average evaluation of proposed algorithm compared with the BBA and the ENSEBRBA.

Schwefel 2D: The expected optimum result for the function is -837.97. The average result obtained from a hundred runs of the BBA and the ENSEBRBA were -837.144 and -837.964 respectively. BBA used an average of 250049 evaluations to find that optimum, whereas the average for the ENSEBRBA was 338.600. The TBA used to solve same optimisation problem and the algorithm found the average global optimum as -837.93 in 62054 evaluations. Figure 4.6 illustrates global optima for a hundred runs of BBA, ENSEBRBA and TBA on the given problem. Although all versions of the Bees algorithm produced fairly accurate results for this optimisation problem, the TBA found the global optimum in lower number of evaluations. This is because TBA has memory and the algorithm avoids revisiting already visited sites.

Schaffer 2D: The expected global optimum is 0. The experimental results computed by evaluation of a hundred runs were 0.01 and 0.001 respectively. BBA used an average of 250049 evaluations to find that result, whereas the average of ENSEBRBA evaluations was 112430. The TBA is utilised to solve this benchmark function and the average global optimum was found as 0 in 6309 evaluations. The results of a hundred runs for BBA, ENSEBBA and TBA are given in Figure 4.7. Based on experimental results, the TBA found a more accurate global optimum for the problem in lower number of evaluations than other described versions of the Bees Algorithm.

Rosenbrock 10 D: The expected optimum is 0. An average of a hundred experimental results obtained from the BBA and the ENSEBRBA were 0.0003 and 0.0002 respectively. BBA used an average of 11690 evaluations to find that result, whereas the average of ENSEBRBA evaluations was 148193. Moreover, TBA was applied on the same function and the new algorithm found the average global

optimum as 0 in 9821 evaluations. Figure 4.8 illustrates global optima for a hundred runs of BBA, ENSEBRBA and TBA on the given problem. Although ENSEBRBA was better than the BBA in terms of results on this complex high dimensional problem, due to the memory unit used in TBA, results were improved even further. This is because Rosenbrock's global optimum is located at the flat valley, which has a long narrow parabolic shape and the TBA, with a local escape strategy, converges to the global optimum easily.

Hyper Sphere 10D: The global optimum for this function is 0. Experimental results computed by evaluation of a hundred runs were 0.0003 and 0.0001 respectively. The BBA needed an average of 285039 evaluations to find that result, when the average of the proposed ENSEBRBA evaluations was 95643. Experimental results obtained from the TBA on the same benchmark function were 0.0001 (global optimum) in 2972 evaluations. The results of a hundred runs for BBA, ENSEBRBA and TBA are given in Figure 4.9. The proposed version of the Bees Algorithm produced the same global optimum as the ENSEBRBA, which was already better than the optimum obtained using the BBA. However, due to utilised memory, the TBA used fewer evaluations to get that result.

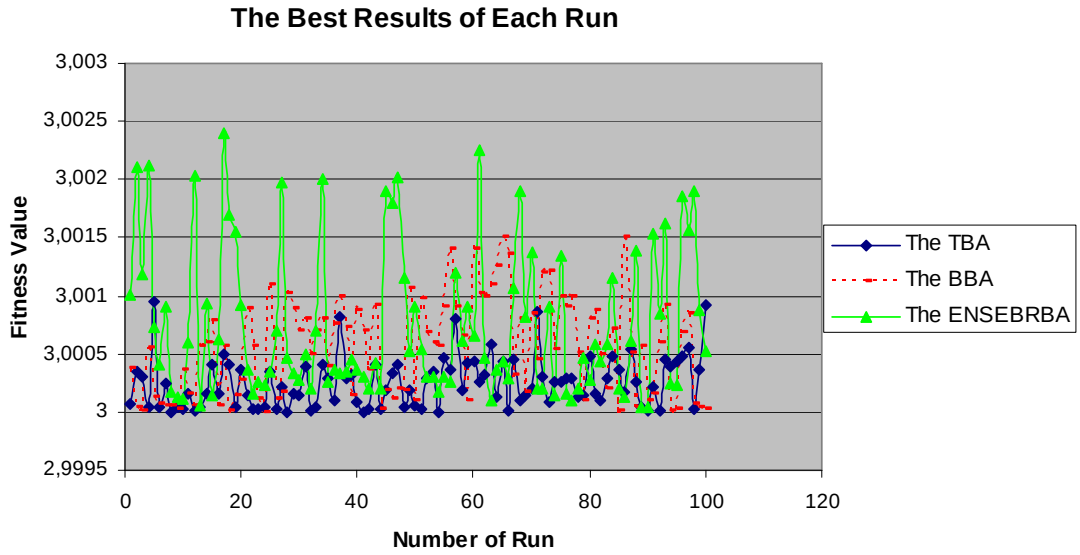

Figure 4.5: The results of a hundred runs for BBA, ENSEBRBA and TBA on Goldsein-Price 2D function.

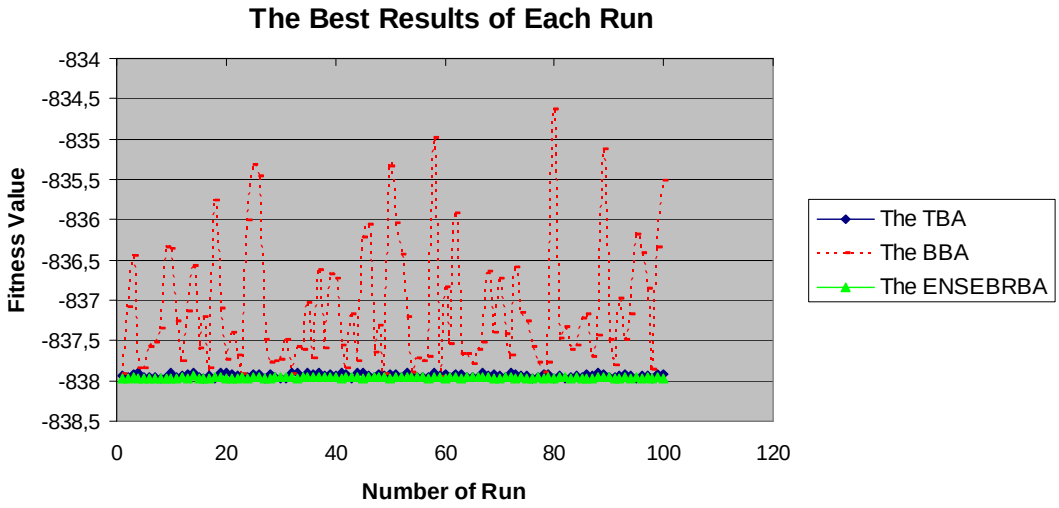

Figure 4.6: The results of a hundred runs for BBA, ENSEBRBA and TBA on Schewel 2D function.

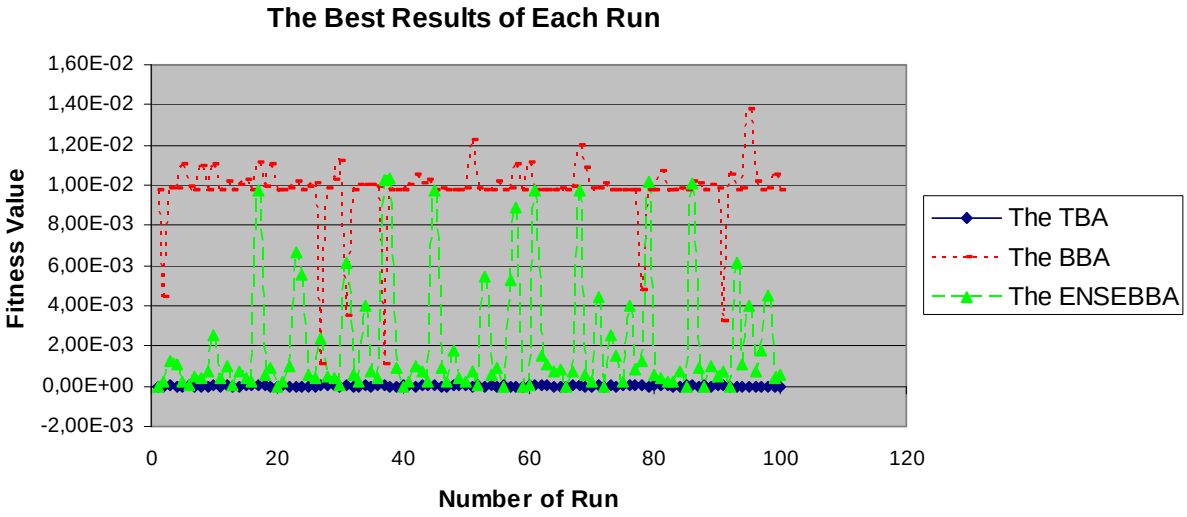

Figure 4.7: The results of a hundred runs for BBA, ENSEBRBA and TBA on Schaffer 2D function.

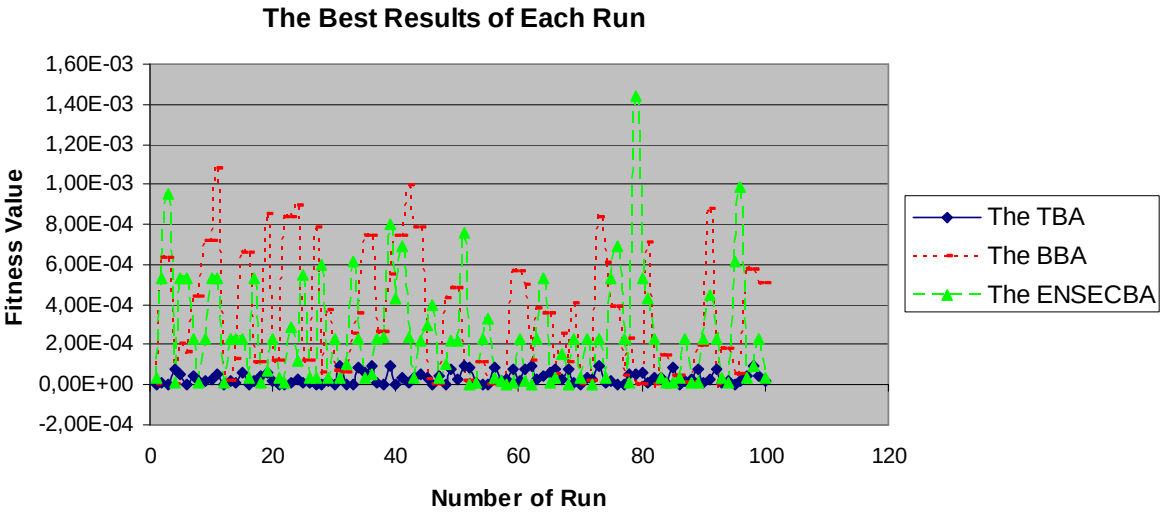

Figure 4.8: The results of a hundred runs for BBA, ENSEBRBA and TBA on Rosenbrock 10 D function.

Ackley 10D: The expected optimum of this is 0. According to the experiments, the computed global optimum obtained by the evaluation of a hundred runs were 0.029 for the BAA and 0.0001 for the ENSEBRBA. The BBA needed an average of 910000 evaluations to find that result and the average number of evaluations of the ENSEBRBA was 236299. The TBA was utilised to solve the same optimisation problem and the algorithm found an average global optimum as 0,0001 in 2972 evaluations. The results of a hundred runs for BBA, ENSEBRBA and TBA are given in Figure 4.10. As for the previous function, the proposed version of the Bees Algorithm produced the same global optimum as the ENSEBRBA which was already better than optimum found by the BBA. Because of the memory factor, the TBA used a lower number of evaluations to converge to a global optimum.

Rastrigin 10D: The global optimum of this function is 0. An average of a hundred optima of the BBA and the ENSEBRBA were 0.005 and 0.0002 respectively. An average of a hundred evaluations for the BBA was of 885000, when the corresponding result for the ENSEBRBA was 53935. The result of the same experiment using the TBA was an average global optimum as 0 in 7559 evaluations. The results of a hundred runs for BBA, ENSEBBA and TBA are given in Figure 4.11. On the Rastrigin function, the proposed algorithm performed better than the other two. The TBA was better than the BBA in all aspects and as it was expected that TBA would surpass ENSEBRBA on number of evaluations.

Martin & Gaddy 2D: The expected global optimum is 0**.** According to the experiments, the computed results were obtained by evaluation of a hundred runs. The average optimum obtained from the BBA and the ENSEBRBA were both 0. BBA

needed an average of 600 evaluations to find that result, whereas the average for ENSEBRBA was 15888. The average of a hundred global optima found by using the TBA to solve the same optimisation problem was 0 and average number of evaluations was 1065. Figure 4.12 illustrates global optima for a hundred runs of BBA, ENSEBRBA and TBA on the given problem. On the Martin and Gaddy function, the Basic version of the Bees algorithm performed better than both proposed versions. However, the TBA obtained global optimum in fewer evaluations than the ENSEBRBA.

Easom 2D: The expected optimum result for this function is -1. The average result obtained from a hundred runs of the BBA and the ENSEBRBA were -0.707 and -0.9999 BBA needed an average of 5280 evaluations. The corresponding results for ENSEBRBA were 1120. TBA was used to solve the same optimisation problem and the algorithm found the average global optimum as -0.9999 in 1063 evaluations. Figure 4.13 illustrates global optima for a hundred runs of BBA, ENSEBRBA and TBA on the given problem. On the given function the TBA found the same global optimum as the ENSEBRBA, which was already better than the optimum found by the BBA. This is because the Easom function's optimum is in a small area compared to a large search space. Therefore, the BBA with the standard approach is unable to converge to the global optimum.

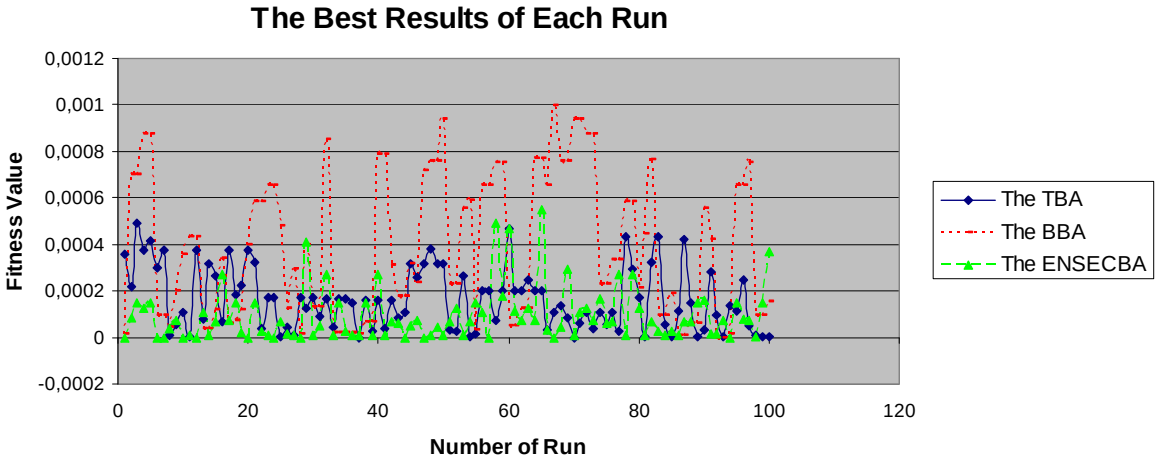

Figure 4.9: The results of a hundred runs for BBA, ENSEBRBA and TBA on Hyper sphere 10D function.

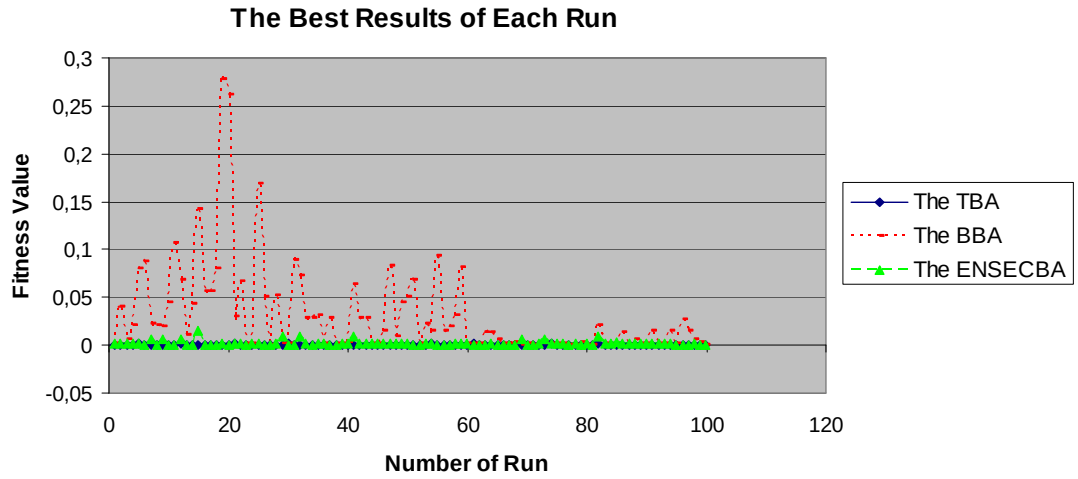

Figure 4.10: The results of a hundred runs for BBA, ENSEBRBA and TBA on Ackley 10D function.

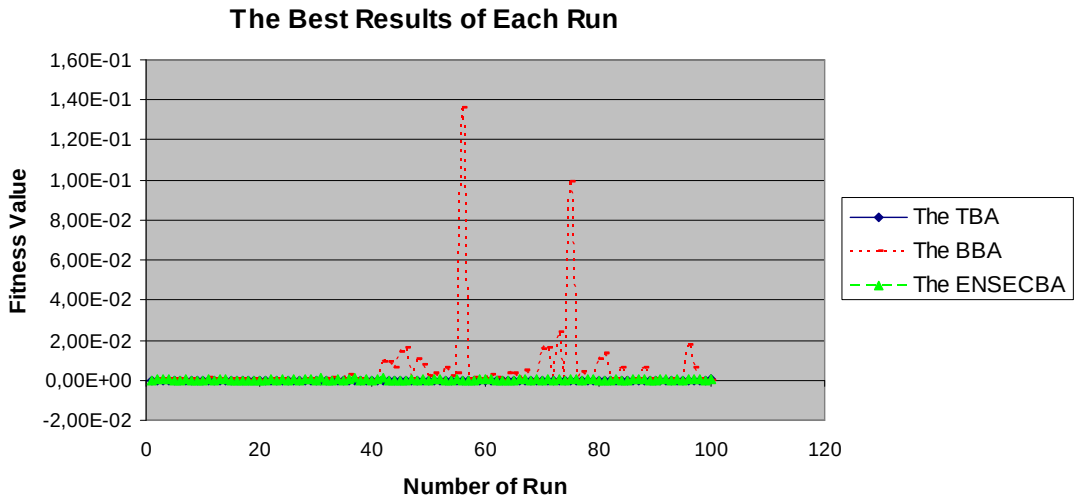

Figure 4.11: The results of a hundred runs for BBA, ENSEBRBA and TBA on Rastrigin 10D function.

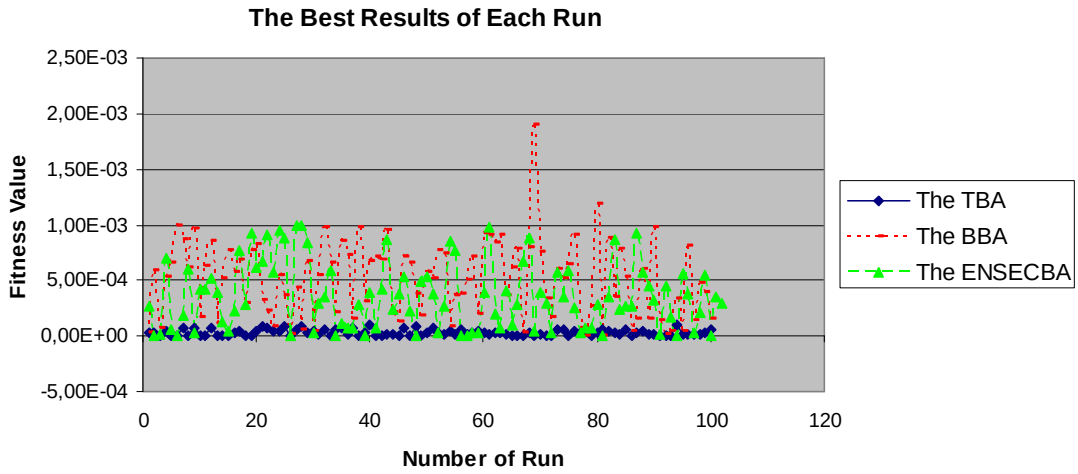

Figure 4.12: The results of a hundred runs for BBA, ENSEBRBA and TBA on Martin & Gaddy 2D function.

Inverted Griewank 10D: The expected global optimum for this function is 10. The average result obtained from the Basic Bees algorithms over hundred runs was 9.989. The corresponding result obtained by ENSEBRBA was 9.990. BBA needed an average of 4300000 evaluations to find that result, whereas the average of ENSEBRBA's evaluations was 316443. TBA was used to solve the same optimisation problem and the algorithm found an average global optimum as 9.9999 in 8294 evaluations. The experimental results of a hundred runs for BBA, ENSEBBA and TBA are given in Figure 4.14. Due to having widely spread local optima, the BBA has performed poorly on the Griewank function. The ENSEBRBA found a fairly accurate global optimum but in a high number of evaluations. However, the TBA found the most accurate global optimum in fewer evaluations because this algorithm has poor location avoidance mechanisms.

Although main reason to develop both introduced strategies was to decrease the number of evaluations used by the BBA to find the global optimum, overall results illustrate that the accuracy of the BBA was significantly increased in the process as well.

Further statistical analysis was carried out by using t-test, where the confidence level was selected to be 95 % ($\alpha < 0.05$). The T- test results are illustrated in table 4.5. From the t-test results between the Tabu Bees Algorithm and the Basic Bees Algorithm it is clearly seen that TBA performs statistically significantly better.

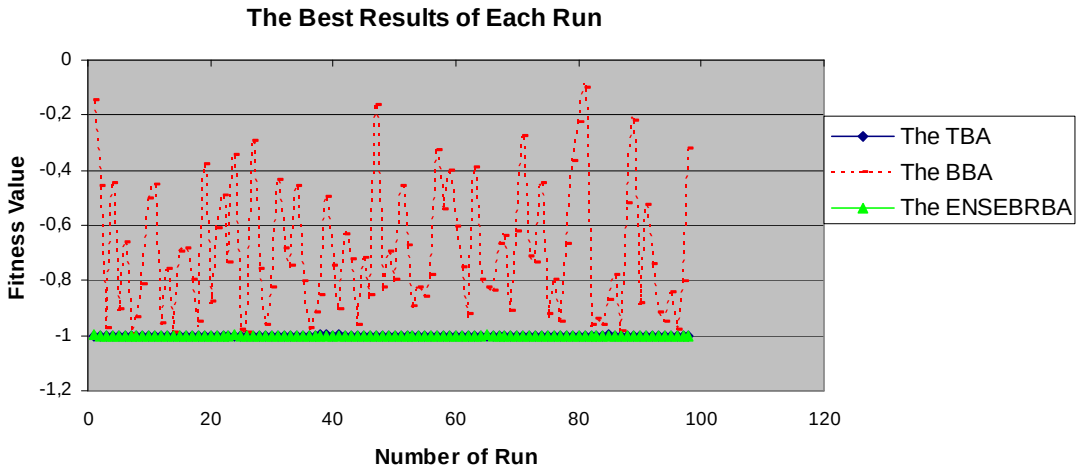

Figure 4.13: The results of a hundred runs for BBA, ENSEBRBA and TBA on Easom 2D function.

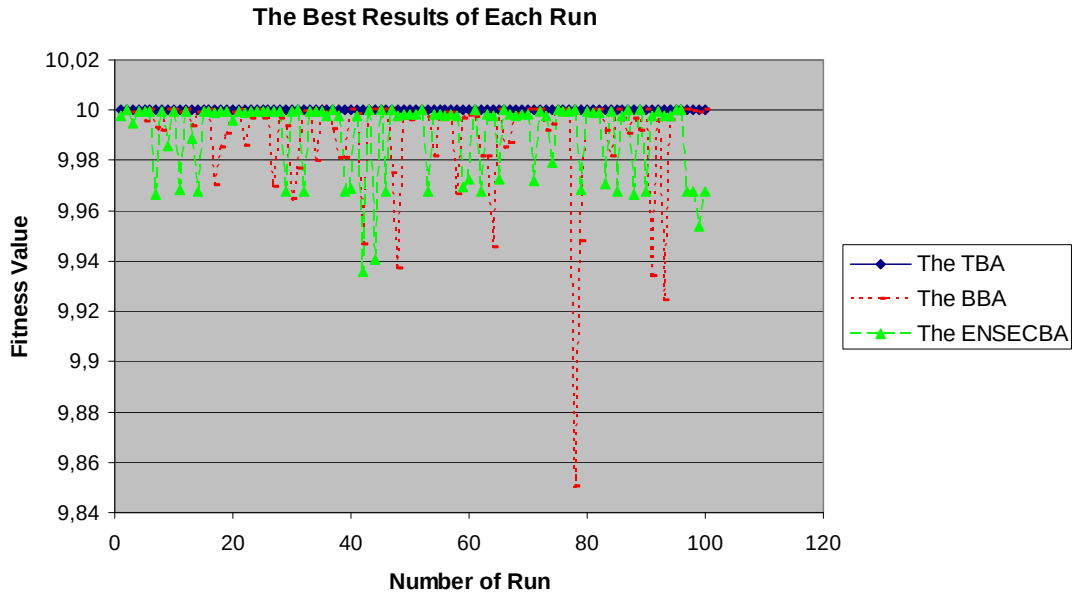

Figure 4.14: The results of a hundred runs for BBA, ENSEBRBA and TBA on inverted Griewank 10D functions.

No.	Function	Significance between the TBA and the BBA	
		Significant ($\alpha<0.05$)	α
1	Goldstein & Price (2D)	Yes	0,0021
2	Schwefel (2D)	Yes	2,59325E-17
3	Schaffer (2D)	Yes	2,24632E-14
4	Rosenbrock (10D)	Yes	1,42986E-15
5	Sphere (10D)	Yes	3,82E-09
6	Ackley (10D)	Yes	2,09428E-08
7	Rastrigin (10D)	Yes	0,0057
8	Martin & Gaddy (2D)	Yes	4,94477E-25
9	Easom (2D)	Yes	2,53E-22
10	Griewank (10D)	Yes	3,06308E-06

Table 4.5: The statistical analysis between the TBA and the basic Bees Algorithm.

4.7 Summary

In this study a novel algorithm was proposed which is hybrid between the BBA and Tabu. The new algorithm is called Tabu Bees algorithm. In this algorithm, tabu list was utilised to give memory to the BBA to solve the site repetition problem. In addition, a new adaptive neighbourhood strategy was proposed to overcome the issue of getting stuck around local optima with similar fitness values. The proposed algorithm has been successfully applied on continuous type benchmark functions and compared with the BBA and ENSEBRBA.

Accuracy analysis, average evaluation and t-test were utilised compute the performance of the proposed algorithm.

According to the Experimental results it can be concluded that the number of evaluations needed both on lower and higher dimensional problems were dramatically decreased. On the other hand, the proposed improvements increased the accuracy of algorithm as well. Based on t-test results, it can be concluded that the proposed algorithm is statically significantly better performing than the basic Bees Algorithm.

Chapter 5

The Autonomous Bees Algorithm

5.1 Preliminaries

In this chapter, the Autonomous Bees Algorithm (ABA) is presented as a solution for the below mentioned problem.

Various weaknesses of the BBA were discussed in previous chapters and some enhancements were introduced to solve these problems. This chapter focuses on the one of the biggest issues for the BBA, which is the large number of parameters to be set manually. These parameters must be tuned to produce accurate results. Although the BBA is a relatively easy algorithm to apply on different optimisation problems, the large number of parameters makes it hard for new users.

As a concept, autonomy is the capacity of an individual to make an informed, un-coerced decision. It is widely used in fields like politics, sociology, religion and engineering. Autonomy has applications in artificial intelligence as well. For example: Autonomous Genetic Algorithm for Functional Optimisation (Meng, 2007).

In the literature several studies on parameter tuning for the BBA have been presented. However, these studies did not provide the BBA with full independence. The ABA is a self-directed version of the BBA where interaction between the user and the process is on a minimal level.

The block diagram of the ABA is given in Figure 5.1.

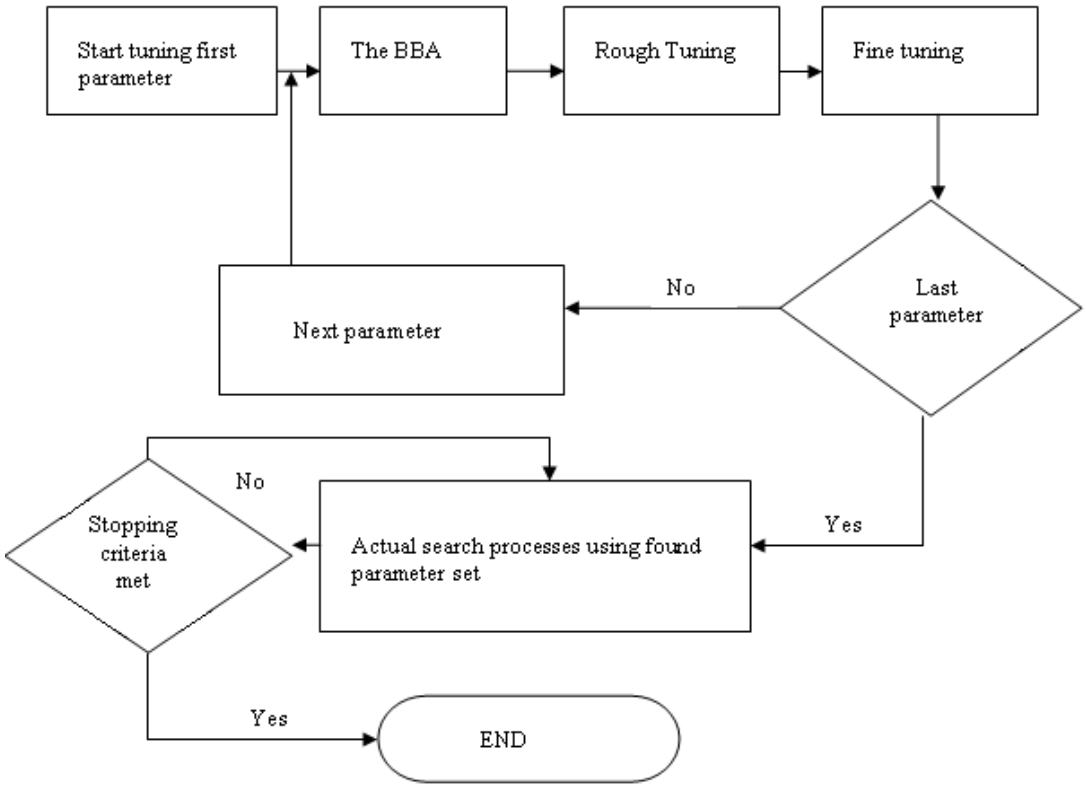

Figure 5.1: Block diagram of the ABA.

5.2 Autonomous Behaviour

In this section autonomous behaviour of the ABA is explained in detail. To illustrate every step of algorithm, a ten dimensional Hyper Sphere function was chosen. The definition of the used function was given in Chapter 2.

The ABA starts search with a set of predefined parameters. It is then guided, based on previous information, toward a better parameter set. The default values of the parameters are given below:

- Number of scout bees. *n =10;*
- Number of sites selected out of n visited sites. *m = 3;*
- Number of best sites out of m selected sites. *e =1;*
- Number of bees recruited for best e sites. *nep =8;*
- Number of bees recruited for the other (m-e) selected sites. *nsp =4;*
- Patch size around of a selected best location. *ngh=1;*

In previous research on the BBA, the parameters were tuned as given numbers empirically to solve many different optimisation problems. Therefore, it is quite promising to start searching with these parameters. The ABA tunes parameters one at time and there are two steps for each of them: rough tuning and fine tuning.

Determining n: Number of scout bees is the first parameter to be tuned. It is an important parameter because if “n” is too low, the algorithm will fail to find the

optimum and if n is too large the number of evaluations needed will be high. Based on previous experience, it can be said that the number of scout bees alters between 0 and 100 depending on the structure of the problem. The algorithm creates ten equal groups of numbers in this range and randomly picks one value from each group *[1-10; 11-20; 21-30; 31-40; 41-50; 51-60; 61-70; 71-80; 81-90; 91-100]*.

Further, algorithm uses these values as the number of scout bees to do a search on the optimisation problem. After running the search for each of the scout bees, the ABA evaluates the obtained results to choose the most promising group of numbers. The algorithm assesses the results based on fitness values and the number of evaluations prioritised on the fitness values. In our experiment, the algorithm chose 69 as most promising number of scout bees, as shown in Figure 5.2a. The fitness value obtained using 69 bees to do the scouting was 0, which is the expected result for the Hyper Sphere function. The same result was obtained while using other numbers of scout bees as well. However, the number of evaluations needed to achieve that result was the lowest for 69 scout bees (Figure 5.2b). Finding this value is considered as rough tuning of parameter “n” (number of scout bees).

After determining the rough value of “n”, the algorithm carries out the fine tuning of the parameter, using every member of the group where the rough value of “n” was found *[61; 62; 63; 64; 65; 66; 67; 68; 69; 70]*. Furthermore, ABA evaluates the results of the fine tuning using the same strategy (based on global optimum and evaluation numbers) as for rough tuning. The number of scout bees was selected to be 66 after fine tuning, as illustrated in figures 5.3a and 5.3b.

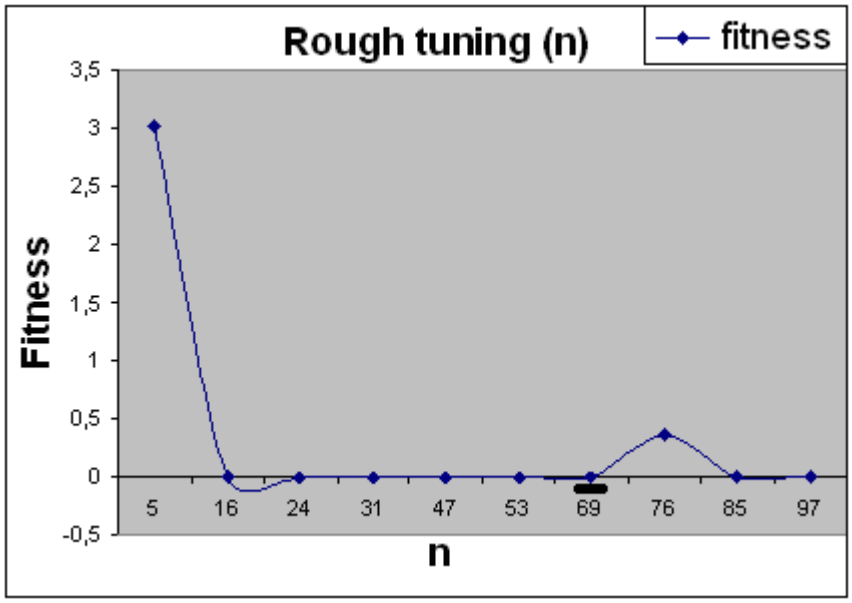

Figure 5.2a: Fitness values obtained after Rough Tuning of "n".

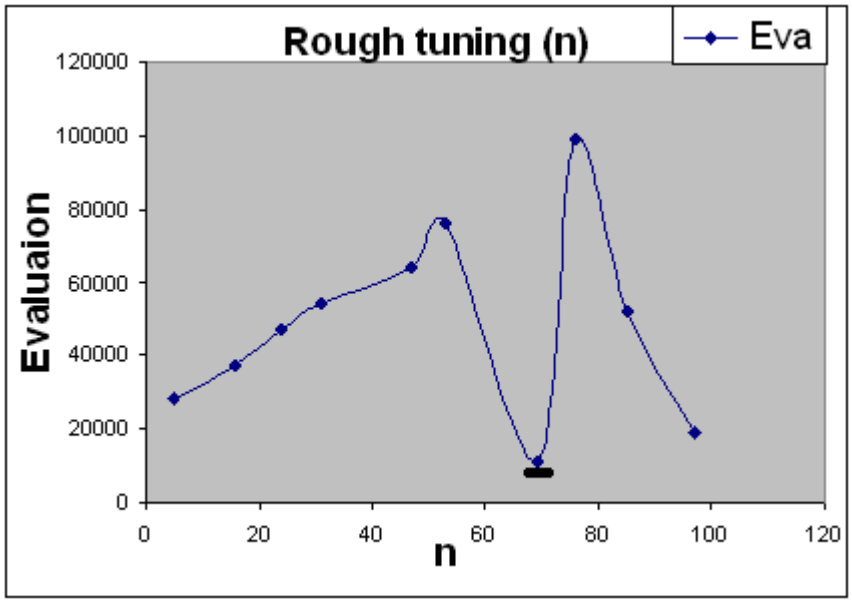

Figure 5.2b: Number of Evaluations obtained after Rough Tuning "n".

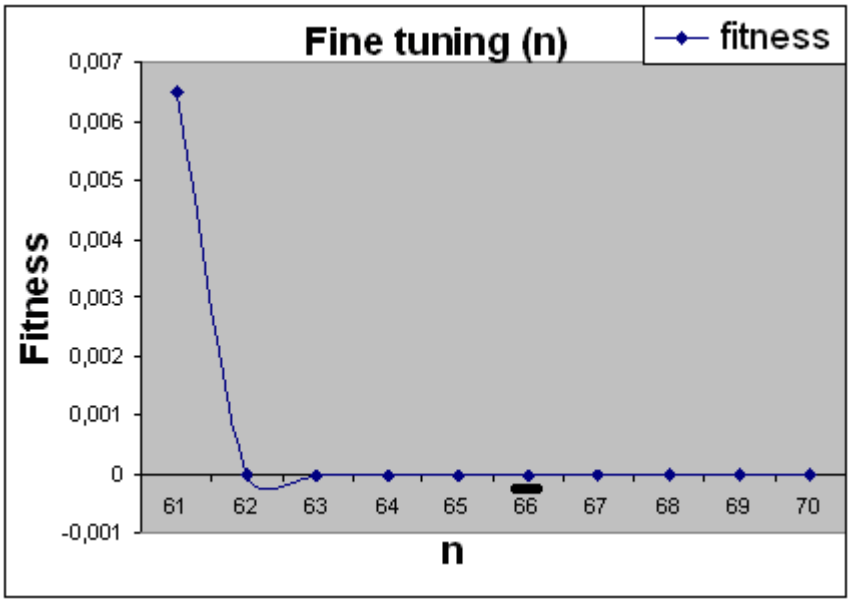

Figure 5.3a: Fitness values obtained after Fine Tuning of "n".

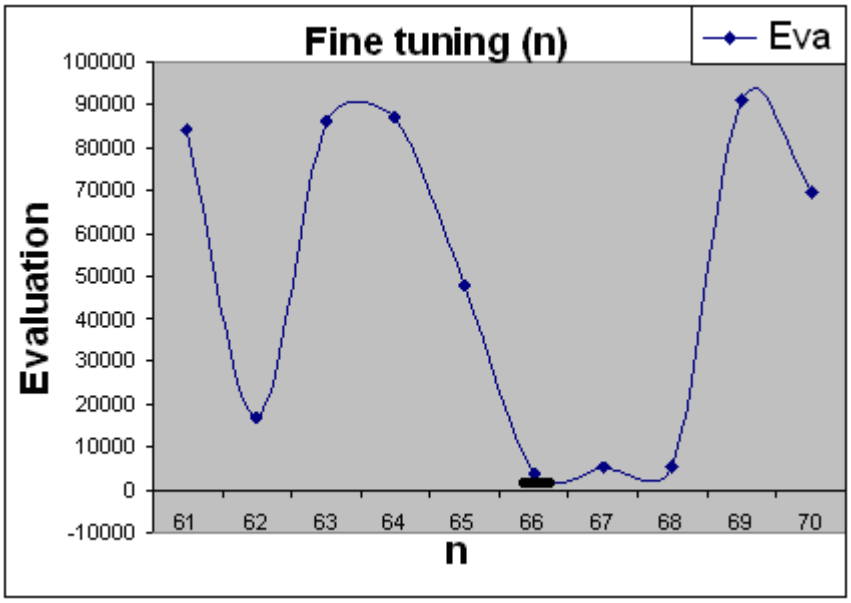

Figure 5.3b: Number of Evaluations obtained after Fine tuning of "n".

Determining m: Number of sites selected for local search is another very important parameter to be determined accurately. The **v**alue of "m" changes the core behaviour of the algorithm, such as:

- 0<m<n : Classical global and local search of the Bees Algorithm
- 0=m<n: Only global search of the Bees Algorithm.
- 0<m=n : Only local search of the Bees Algorithm

After finding the value of "n", the algorithm starts tuning the next parameter, which is "m". Accordingly, "m" can not exceed "n". Therefore the value of "m" will be between 0 and 66. The ABA creates ten groups of numbers in that range and chooses random numbers from each of them *[0-6; 7-13; 14-20; 21-27; 28-35; 36-42; 43-49; 50-56; 57-63; 64-66;]*.

Based on the best fitness and evaluations, the algorithm selects the rough value of "m", which is 9 for our problem, as shown in Figures 5.4a and 5.4b.

Fine tuning can then be performed when the rough number of "m" is found. Fine tuning is carried out using ten numbers from the group to which 9 belongs. The algorithm needs 10 numbers from that group to undertake fine tuning. If the number of elements in that group is lower than ten, the algorithm adds random elements from same range to the group [*7; 8; 9; 10; 11; 12; 13; (7; 10; 11;)]*.

After evaluating results obtained, "m" was chosen to be 8, as illustrated in Figures 5.5a and 5.5b.

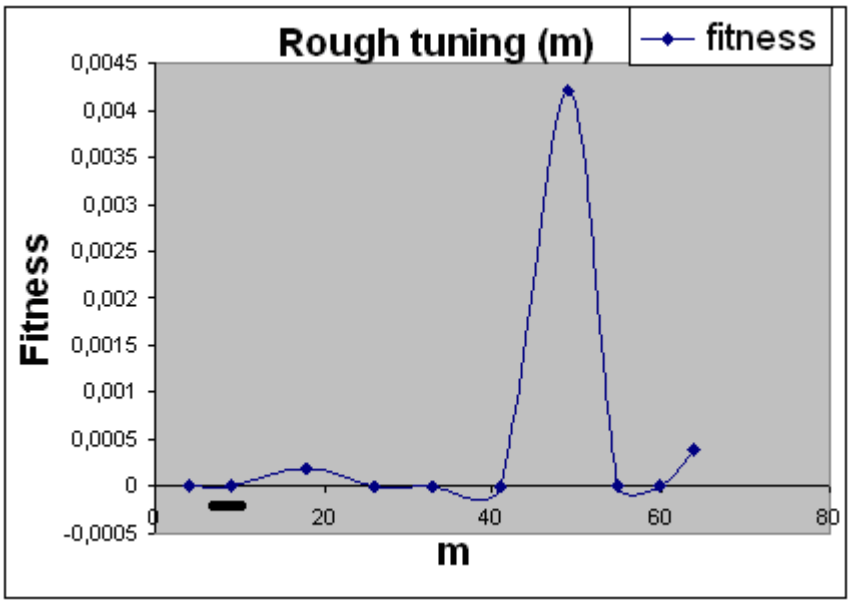

Figure 5.4a: Fitness values obtained after Rough Tuning of "m".

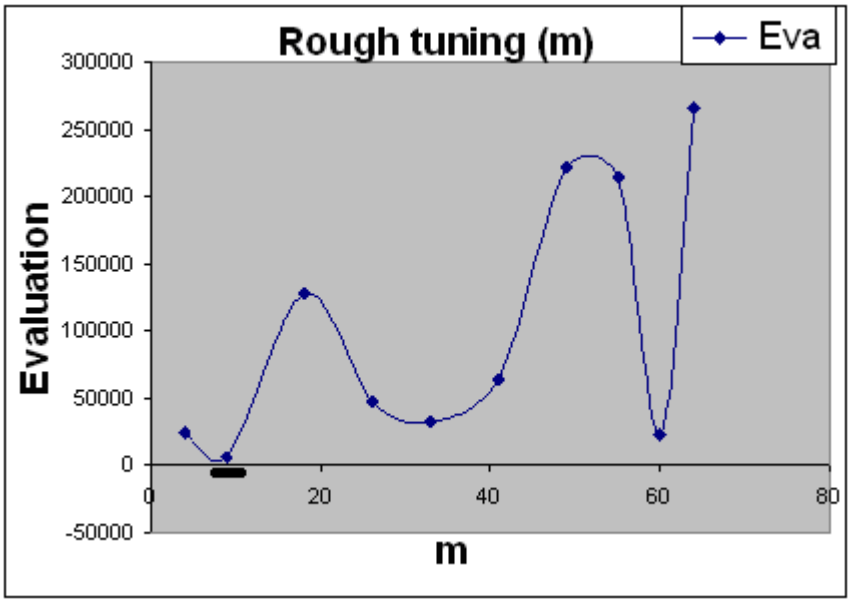

Figure 5.4b: Number of Evaluations obtained after Rough Tuning of "m".

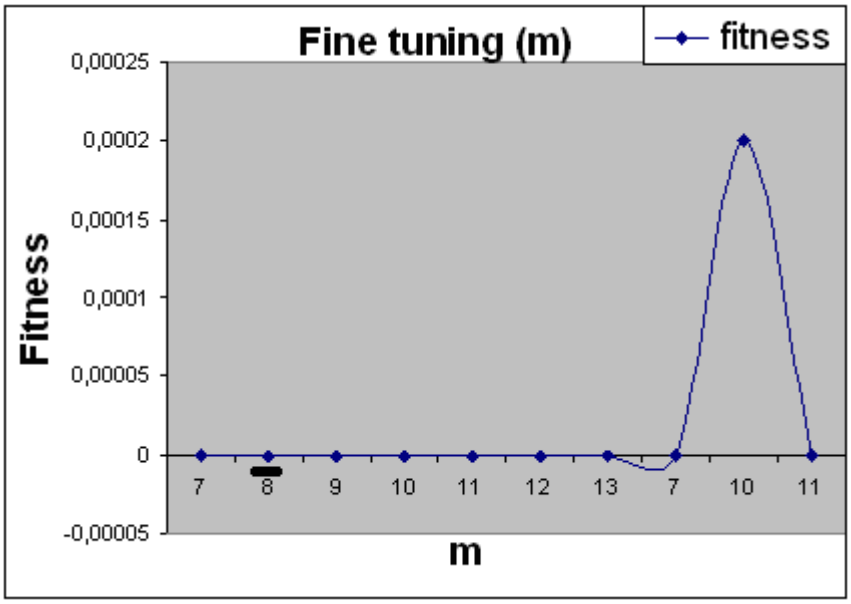

Figure 5.5a: Fitness values obtained after Fine Tuning of "m".

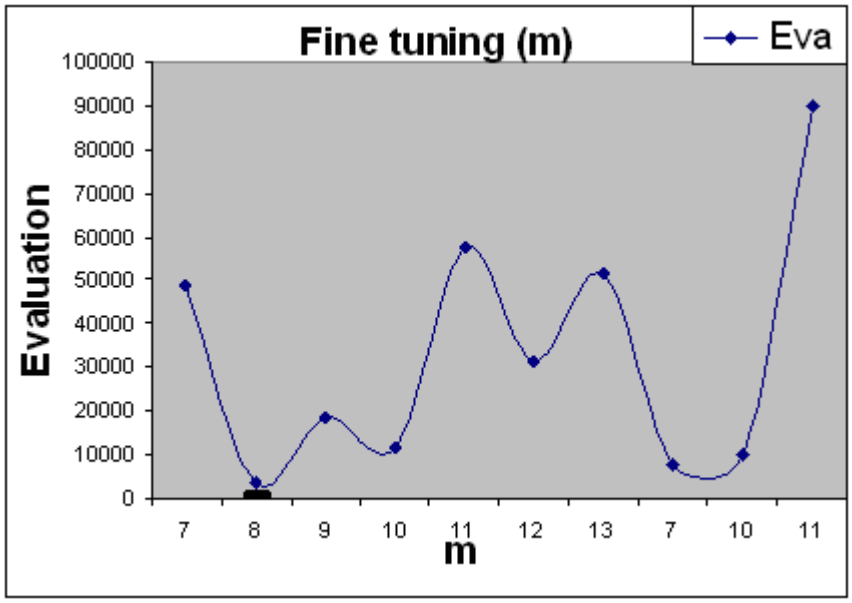

Figure 5.5b: Number of Evaluations obtained after Fine tuning of "m".

Determining e: The same methodology as for determining “m” was used to find the number of elite sites. “e” must be lower or at least equal to “m”. The algorithm will create ten groups of numbers between 0 and 8 and random elements from each group will be selected as “e” while solving the optimisation problem *[0; 1; 2; 3; 4; 5; 6; 7; 8;(0;)]*.

Figures 5.6a and 5.6b illustrate the results of rough tuning where 2 was selected as “e” for further fine tuning.

The fine tuning of “e” in this experiment was relatively easy because there was only one element in the group from where algorithm chooses values of elite sites to perform fine tuning. The results obtained from fine tuning on the given problem are illustrated in Figures 5.7a and 5.7b.

Determining nsp: In general, the number of recruited bees for neighbourhood search on selected sides has no direct relations with number of patches or scout bees. Because of this the parameter is tuned independently from “n”, “m” or “e”. Maximum number of recruit bees is assumed to be 50. This number is divided into five groups of numbers and 2 random values are selected from each group in order to have 10 well distributed values of nsp for comparison [*1-10; 11-20; 21-30; 31-40; 41-50]*.

The rough number of “nsp” is selected to be 3 as shown in Figures 5.8a and 5.8b. The group of numbers which 3 represents is selected for fine tuning of the parameter. After fine tuning of the parameter, the value “nsp” was found to be 2 (Figures 5.9a and 5.9b).

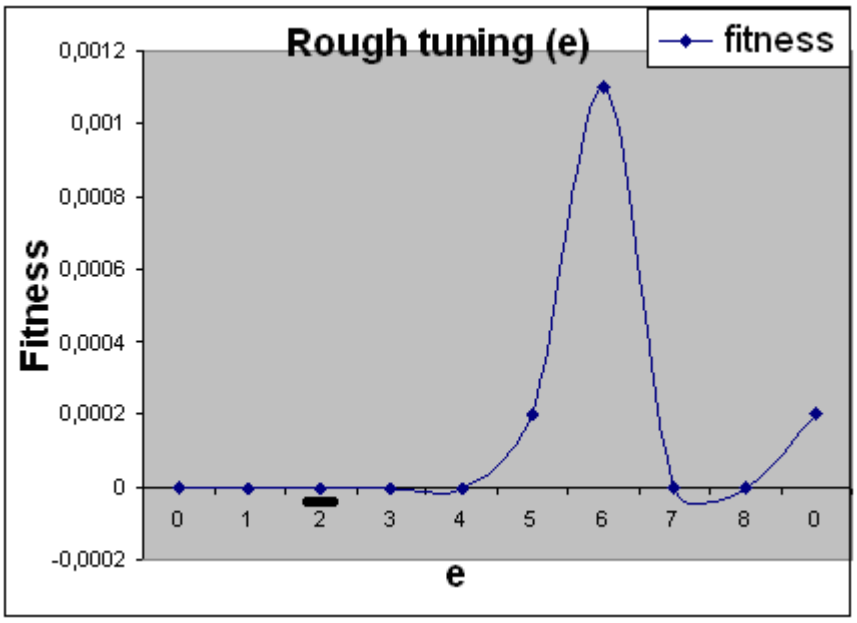

Figure 5.6a: Fitness values obtained after Rough Tuning of "e".

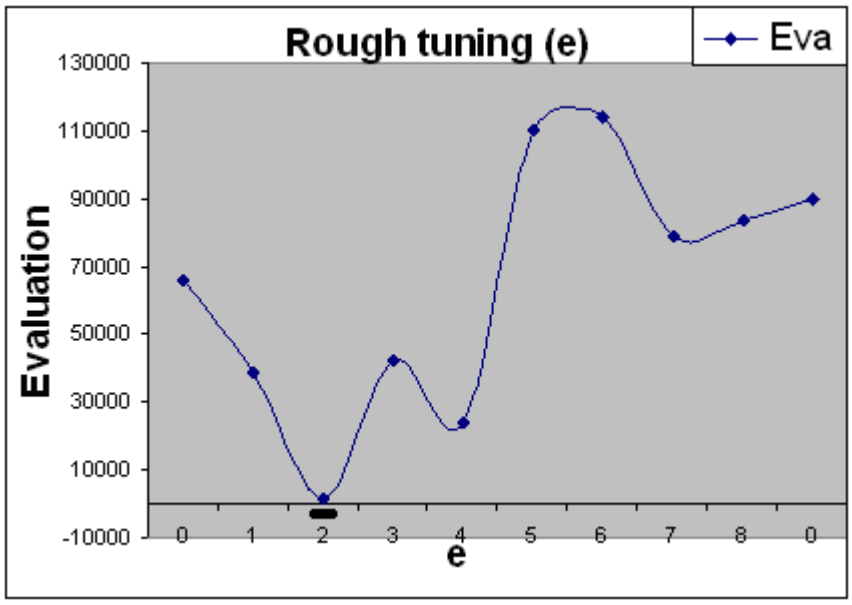

Figure 5.6b: Number of Evaluations obtained after Rough Tuning "e".

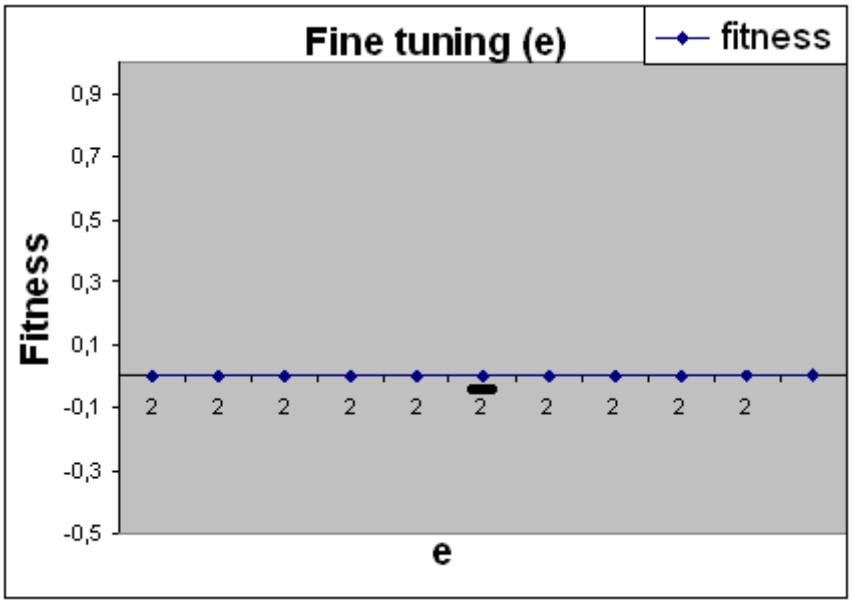

Figure 5.7a: Fitness values obtained after Fine Tuning of "e".

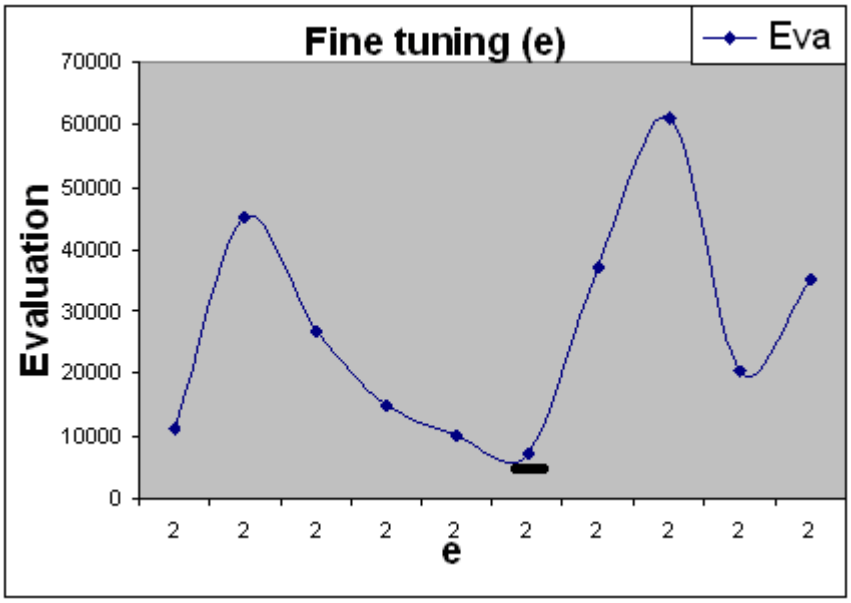

Figure 5.7b: Number of Evaluations obtained after Fine tuning of "e".

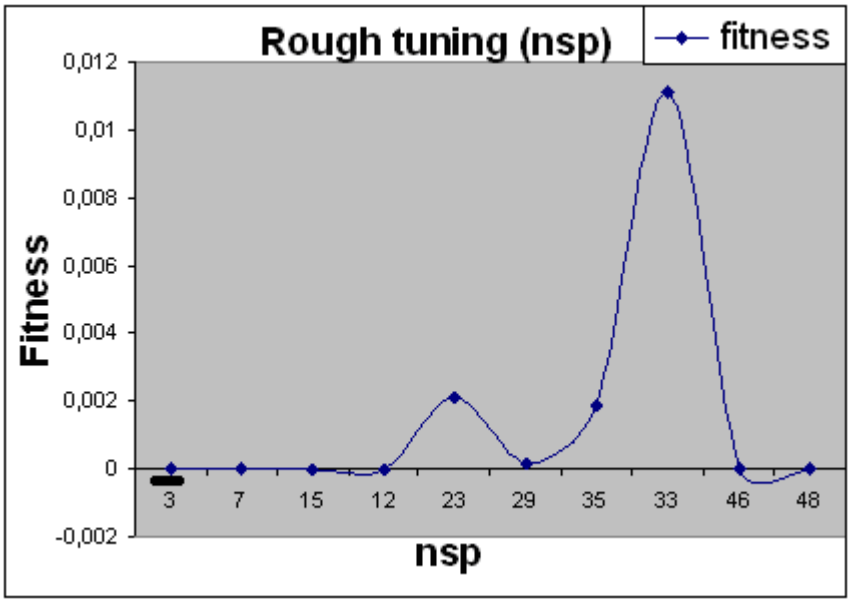

Figure 5.8a: Fitness values obtained after Rough Tuning of "nsp".

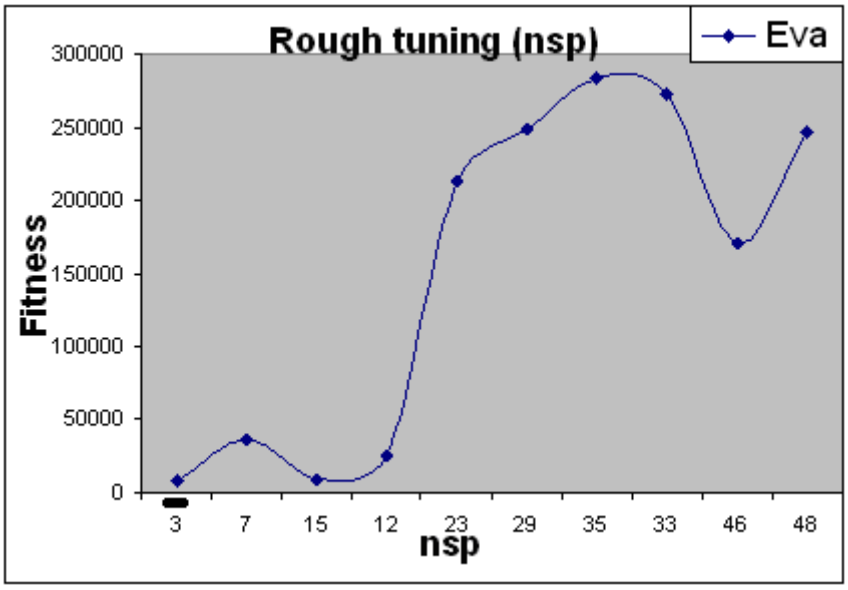

Figure 5.8b: Number of Evaluations obtained after rough tuning of" nsp".

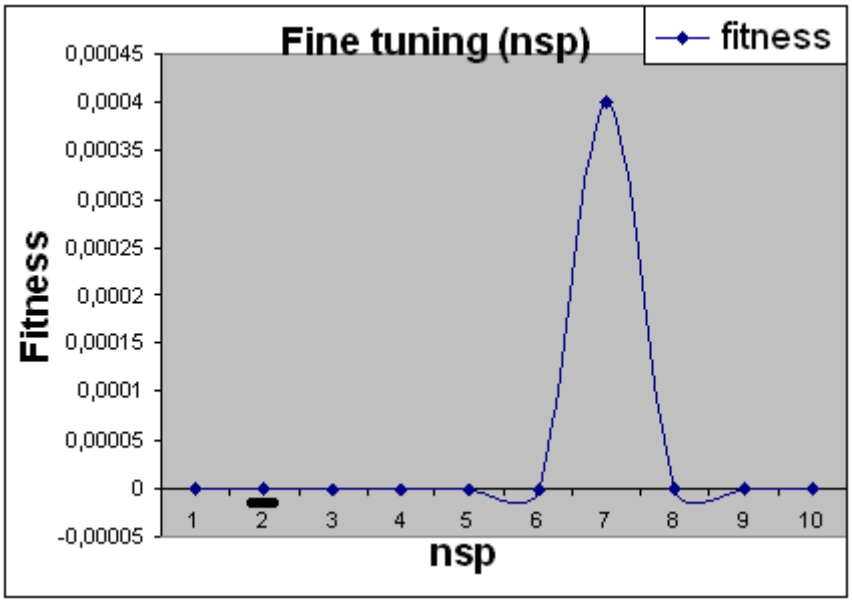

Figure 5.9a: Fitness values obtained after Fine Tuning of "nsp".

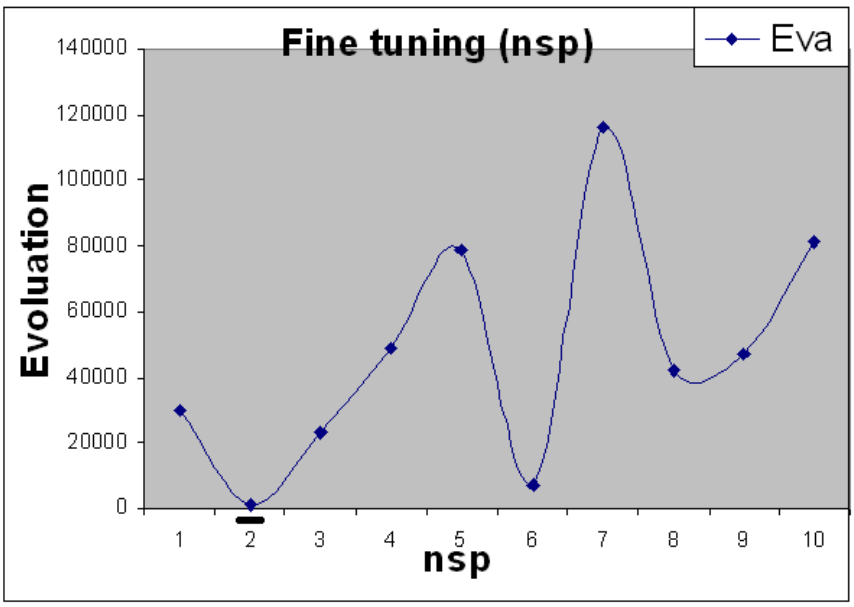

Figure 5.9b: Number of Evaluations obtained after Fine tuning of "nsp".

Determining nep: The next parameter to be tuned is the number of recruit bees for elite sites. In most studies, the value of this parameter is greater than nsp. However, in this study both these parameters are considered to be in the same range. So, the same approach as for nsp was used to tune nep. The results of rough tuning are given in Figures 5.10a and 5.10b. Figures 5.11a and 5.11b illustrate the results of fine tuning.

Determining ngh: Size of neighbourhood search is the last parameter to be tuned. There is no need to do rough tuning for "ngh". In this study, the maximum size of the neighbourhood search was chosen to be 1 and decreased by half for ten times, [*1; 0.5; 0.25; 0.125; 0.062; 0.031; 0.015; 0.007; 0.003; 0.001]*, and the results were compared. Results are given in Figures 5.12a and 5.12b.

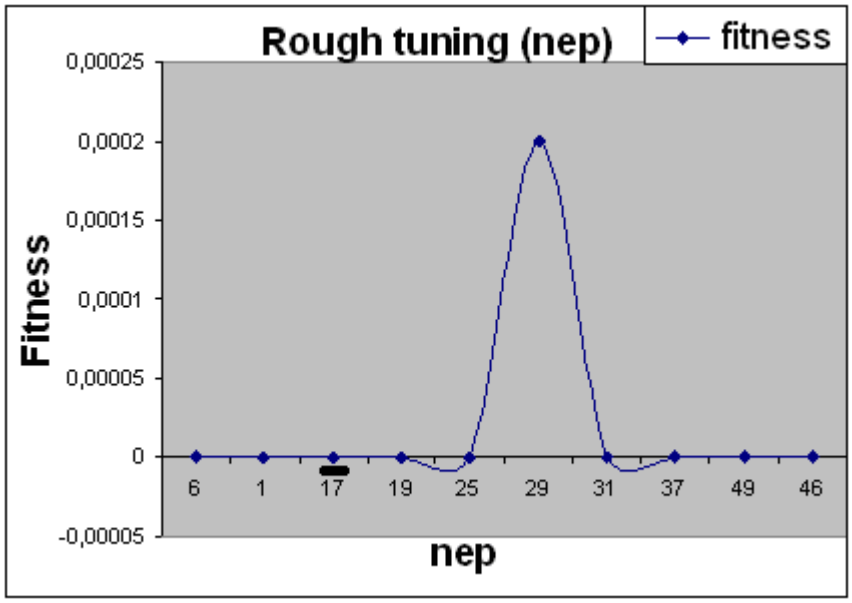

Figure 5.10a: Fitness values obtained after Rough Tuning of "nep".

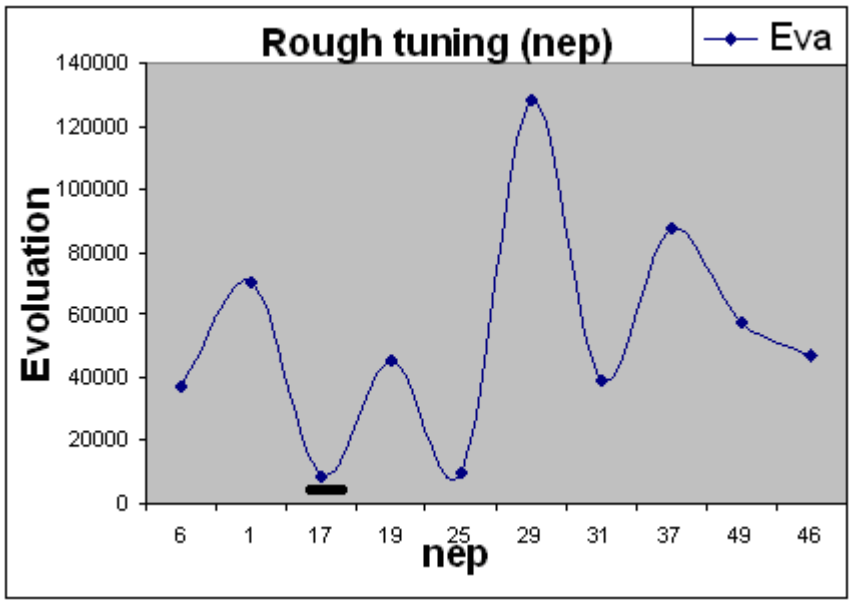

Figure 5.10b: Number of Evaluations obtained after Rough tuning of "nep".

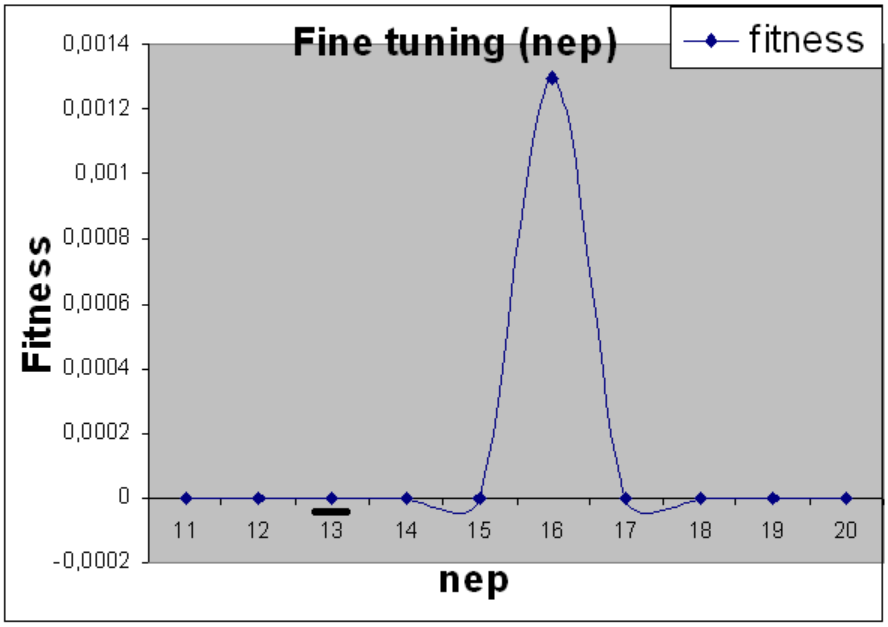

Figure 5.11a: Fitness values obtained after Fine Tuning of "nep".

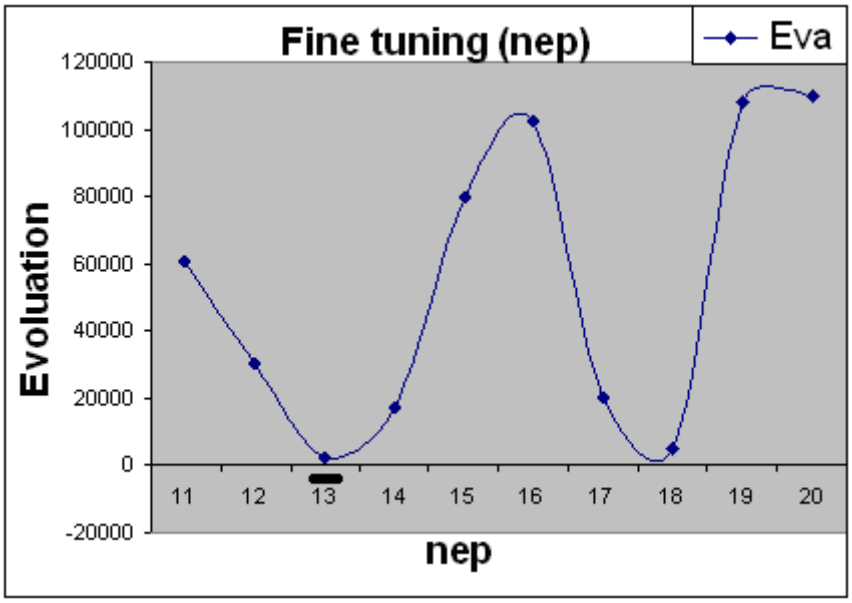

Figure 5.11b: Number of Evaluations obtained after Fine tuning of" nep".

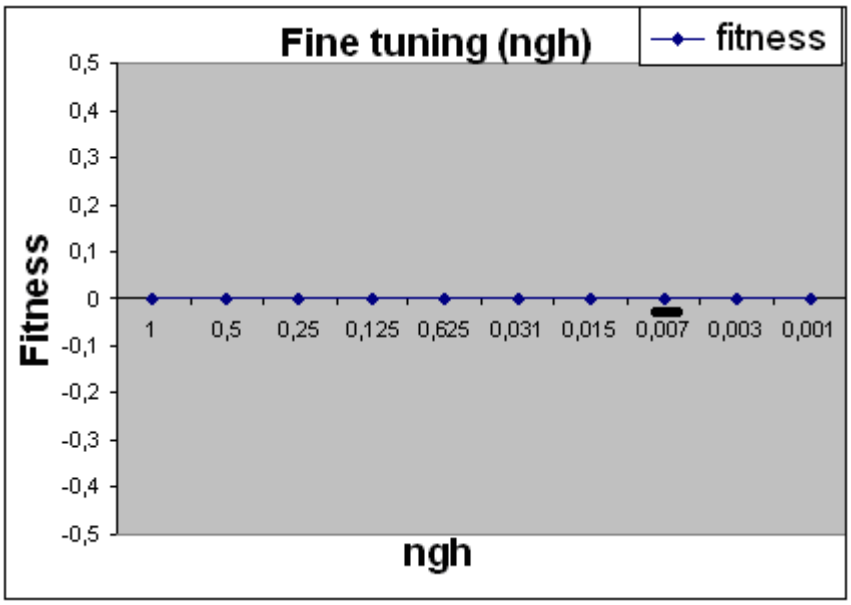

Figure 5.12a: Fitness values obtained after Fine Tuning of "ngh".

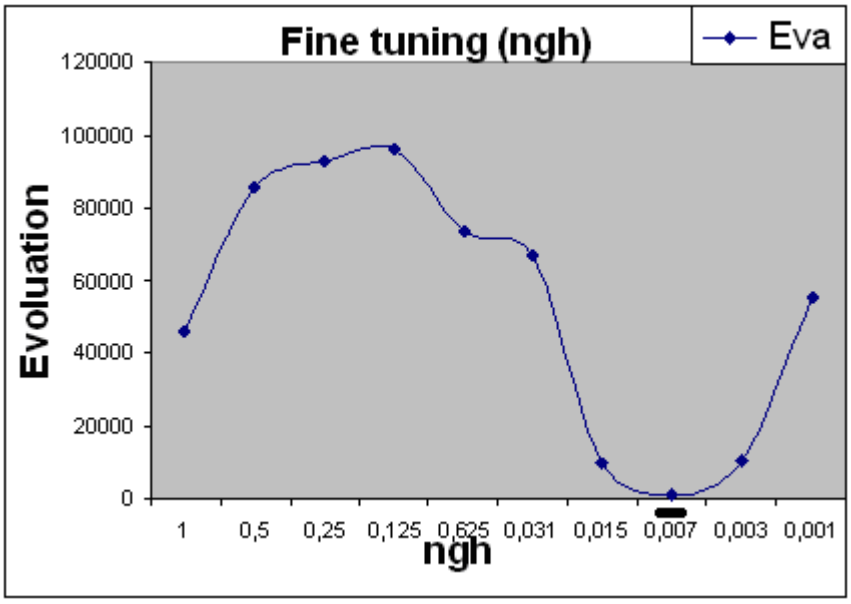

Figure 5.12b: Number of Evaluations obtained after Fine tuning of "ngh".

After tuning the last parameter, which is ngh, the algorithm gets the parameter set to start the actual search. In the case of the Hyper Sphere function, the algorithm has generated the given parameter set:

n=66; m=8; e=2; nsp=2; nep=13; ngh=0,007;

In the following section experimental results obtained from the ABA will be presented.

5.3 Experiments

Ten continuous type benchmark functions were used for experiments to test the productivity of the proposed algorithm. These functions are given in Table 4.1 in chapter four (Pham and Castellani, 2009 and Ahmad, 2012). Brief information about the used test functions was given in chapter 2. The algorithm was applied to all problems as described in the previous section.

5.4 Results and Discussion

The performance of the algorithm was assessed according to the global optima found and the average number of evaluations needed (Table 5.1 and 5.2). Further, T test was utilised to check the significance of the algorithm. Results obtained from the ABA were compared with results of the BBA on the same functions.

Goldstein-Price: The expected global optimum is 3. The parameter set used for BBA to solve this problem was:

n=10; m=3; e=1; nsp=2; nep=13; ngh=0,005;

The average global optimum obtained from the BBA was 3.0005 and the algorithm needed an average of 504 evaluations to find that optimum. The parameter set found by the ABA was:

n=12; m=3; e=2; nsp=4; nep=9; ngh=0,005;

The average of 100 global optima produced by the ABA was 3.0002 and the average of evaluations was 654. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.13. The experimental results obtained from the ABA were better than the results obtained from the BBA. Thus, with a better parameter set, the algorithm becomes more accurate and efficient.

The two dimensional **Schwefel** function was selected for experiment. The expected optimum for the function is -837.97. The parameter set used for BBA to solve this problem was:

n=10; m=2; e=1; nsp=5; nep=6; ngh=0,05;

The average of results obtained from the BBA was -837,144 and the algorithm used an average of 250049 evaluations to find that optimum. The parameter set found by the ABA was:

n=53; m=17; e=3; nsp=8; nep=41; ngh=0.25;

The average of 100 global optima produced by the ABA was -837,711 and the average of evaluations was 163053. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.14. Experimental results show that the performance of the ABA is better than the Basic Bees. However, even the global optimum found by the ABA is not very accurate. This is because the BBA was used as an "engine" in the ABA which already failed to find an accurate global optimum. To overcome this problem, more accurate versions of the Bees Algorithm can be used as a core for the ABA.

The two dimensional **Schaffer** function was selected for experiment. The expected answer for the function is 0. The parameter set used for BBA to solve this problem was:

n=100; m=4; e=2; nsp=10; nep=30; ngh=3;

The average of results obtained from the BBA was 0.01 and the algorithm used an average of 121088 evaluations to find that optimum. The parameter set found by the ABA was:

n=60; m=23; e=5; nsp=5; nep=15; ngh=0.5;

The average of 100 global optima produced by the ABA was 0.0005 and the average of evaluations was 9370. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.15.

No.	Functions	BA	ABA
		Average Evaluations	
1	Goldstein & Price (2D)	504	654
2	Schwefel (2D)	250049	163053
3	Schaffer (2D)	121088	9370
4	Rosenbrock (10D)	935000	529045
5	Sphere (10D)	285039	29906
6	Ackley (10D)	910000	700870
7	Rastrigin (10D)	885000	148960
8	Martin & Gaddy (2D)	600	840
9	Easom (2D)	5280	3137
10	Griewank (10D)	4300000	750020

Table 5.1: Average evaluations obtained from hundred runs of the BBA and the ABA.

No.	Functions	BA	ABA
		Global optimum	
1	Goldstein & Price (2D)	0.0005	0.0002
2	Schwefel (2D)	-837.144	-837,711
3	Schaffer (2D)	0.01	0.0005
4	Rosenbrock (10D)	0.0003	0.0004
5	Sphere (10D)	0.0003	0.0000
6	Ackley (10D)	0.029	0.02
7	Rastrigin (10D)	0.0048	0.0004
8	Martin & Gaddy (2D)	0	0
9	Easom (2D)	-0.707	-0.8168
10	Griewank (10D)	9.9895	9.9949

Table 5.2: Global optimums obtained from hundred runs of the BBA and the ABA.

The **Rosenbrock** function was selected for experiment. The expected answer for the function is 0. The parameter set used for the BBA to solve this problem was:

n=15; m=8; e=5; nsp=10; nep=30; ngh=0,0015;

The average of results obtained from the BBA was 0.0003 and the algorithm used an average of 935.000 evaluations to find that optimum. The parameter set found by the ABA was:

n=23; m=17; e=6; nsp=5; nep=44; ngh=0,003;

The average of 100 global optima produced by the ABA was 0.0004 and the average of evaluations was 529045. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.16. Due to the function's nature, accurate local search is required to find the global optimum. Both the BBA and the ABA found fairly accurate global optima because of utilised local search. However, the ABA found the optimum in fewer evaluations because the proposed algorithm performed a local search on more patches.

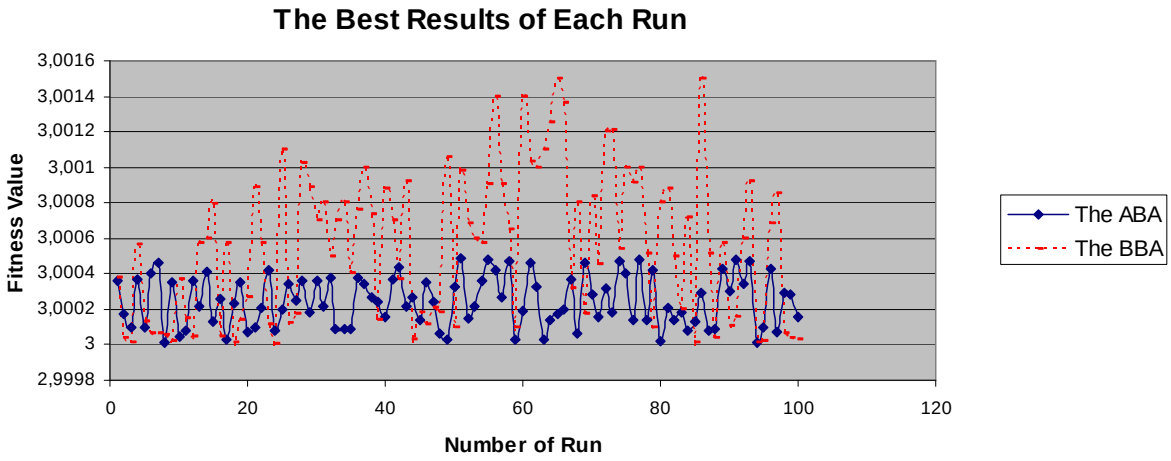

Figure 5.13: The results of a hundred runs for the BBA and the ABA on Goldstein and Price 2D.

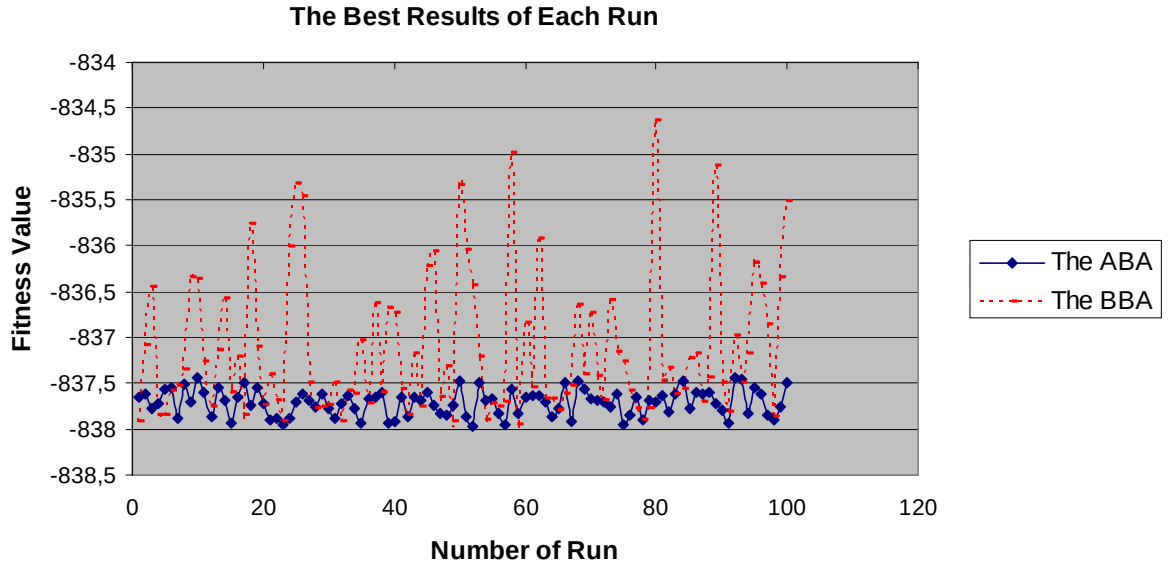

Figure 5.14: The results of a hundred runs for the BBA and the ABA on Schwefel 2D.

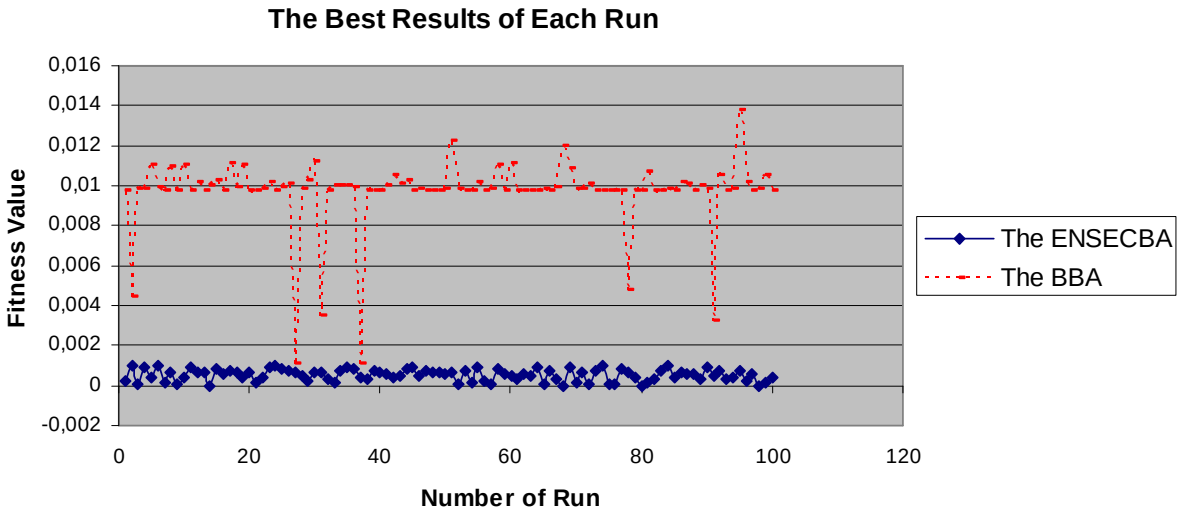

Figure 5.15: The results of a hundred runs for the BBA and the ABA on Schaffer 2D.

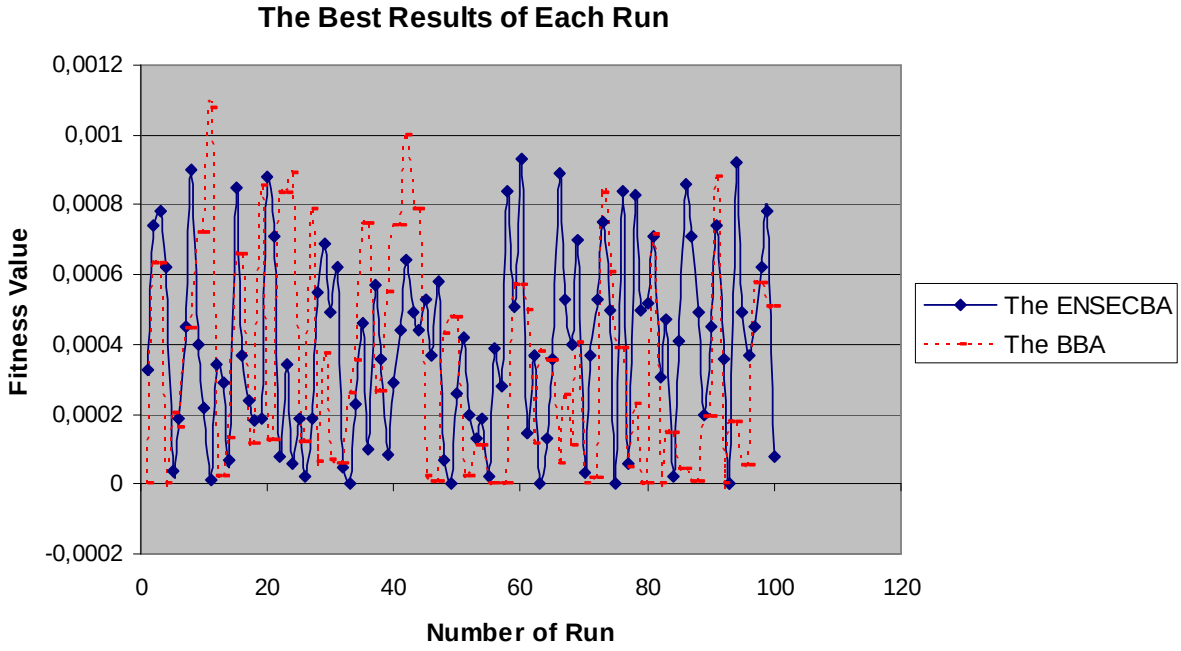

Figure 5.16: The results of a hundred runs for the BBA and the ABA on Rosenbrock 10D.

The **Hyper Sphere** function was selected for experiment. The expected answer for the function is 0. The parameter set used for the BBA to solve this problem was:

n=10; m=7; e=1; nsp=20; nep=30; ngh=0,05;

The average of results obtained from the BBA was 0.0003 and the algorithm used an average of 285039 evaluations to find that optimum. The parameter set found by the ABA was:

n=66; m=8; e=2; nsp=2; nep=13; ngh=0,007;

The average of 100 global optima produced by the ABA was 0 and the average evaluations was 29906. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.17. The influence of the accurate parameter set on the performance of the algorithm can be observed from the experimental results, thus, with more precise parameters, the algorithm obtained better results.

The ten dimensional **Ackley** function was selected for experiment. The expected answer for the function is 0. The parameter set used for BBA to solve this problem was:

n=100; m=8; e=1; nsp=10; nep=20; ngh=0,7;

The average of results obtained from The BBA was 0,029 and the algorithm used an average of 910.000 evaluations to find that optimum. The parameter set found by the ABA was:

n=54; m=6; e=3; nsp=15; nep=24; ngh=1;

The average of 100 global optima produced by the ABA was 0.02 and the average of evaluations was 700870. Global optima obtained from the ABA and the BBA's hundred runs are given in Figure 5.18. Ackley is another hard type benchmark function. Both the ABA and the BBA failed to find a precise global optimum and the number of evaluations needed to get results was not far from each other. However, both algorithms got similar results, so it can be concluded that the ABA is as effective as the BBA for a given optimisation problem.

The ten dimensional **Rastrigin** function was selected for experiment. The expected answer for the function is 0. The parameter set used for BBA to solve this problem was:

n=10; m=3; e=1; nsp=20; nep=30; ngh=0,01;

The average of results obtained from the BBA was 0.048 and the algorithm used an average of 885000 evaluations to find that optimum. The parameter set found by the ABA was:

n=70; m=13; e=7; nsp=8; nep=21; ngh=0.31;

The average of 100 global optima produced by the ABA was 0.0004 and the average of evaluations was 148960. Global optima obtained from the ABA's and the BBA's hundred runs are given in Figure 5.19. In the BBA "n" was chosen too low. The algorithm used the maximum number of evaluations available and stopped searching before converging to an actual global optimum. However, the ABA chose a higher number of initial scout bees, which lead to a more accurate result. Another factor, which affects result on such problems, is the number of sites for local search. On

functions like Rastrigin, algorithms utilising local search are more productive than those that the use only global search.

The two dimensional **Martin & Gaddy** function was selected for experiment. The expected answer for the function is 0. The parameter set used for BBA to solve this problem was:

n=10; m=5; e=1; nsp=10; nep=30; ngh=0,1;

The average of results obtained from the BBA was 0 and the algorithm used an average of 600 evaluations to find that optimum. The parameter set found by the ABA was:

n=13; m=4; e=3; nsp=17; nep=36; ngh=0.625;

The average of 100 global optima produced by the ABA was 0 and the average of evaluations was 840. Global optima obtained from the ABA's and the BBA's hundred runs are given in Figure 5.20. On the given function, performances of both algorithms were approximately the same. As mentioned in previous chapters, the BBA is already enough to solve relatively easy optimisation problems.

The two dimensional **Easom** function was selected for experiment. The expected answer for the function is -1. The parameter set used for the BBA to solve this problem was:

n=100; m=10; e=2; nsp=4; nep=30; ngh=0, 5;

The average of results obtained from the BBA was -0,707 and the algorithm used an average of 5280 evaluations to find that optimum. The parameter set found by the ABA was:

n=69; m=11; e=8; nsp=2; nep=48; ngh=0,5;

The average of 100 global optima produced by the ABA was -0.8168 and the average of evaluations was 3137. Global optima obtained from the ABA's and the BBA's hundred runs are given in Figure 5.21. Both the Basic Bees Algorithm and the Autonomous Bees Algorithm failed to find a global optimum but again, both algorithms generated similar results.

A modified version of the ten dimensional **Griewank** function was selected for experiment. The expected answer for the function is 10. The parameter set used for the BBA to solve this problem was:

n=100; m=40; e=20; nsp=10; nep=30; ngh=1,5;

Average of results obtained from The BBA was 9.989 and algorithm used average of 4300000 evaluations to find that optimum. Parameter set found by the ABA was:

n=48; m=32; e=19; nsp=6; nep=9; ngh=1;

The average of 100 global optima produced by the ABA was 9.9949 and the average of evaluations was 750020. Global optima obtained from the ABA's and the BBA's hundred runs are given in Figure 5.22. For this function "n" was chosen too high for the BBA, which caused an unusually high number of evaluations. Although the ABA found a similar global optimum, due to well tuned parameters, number of evaluations to find the global optimum was significantly lower.

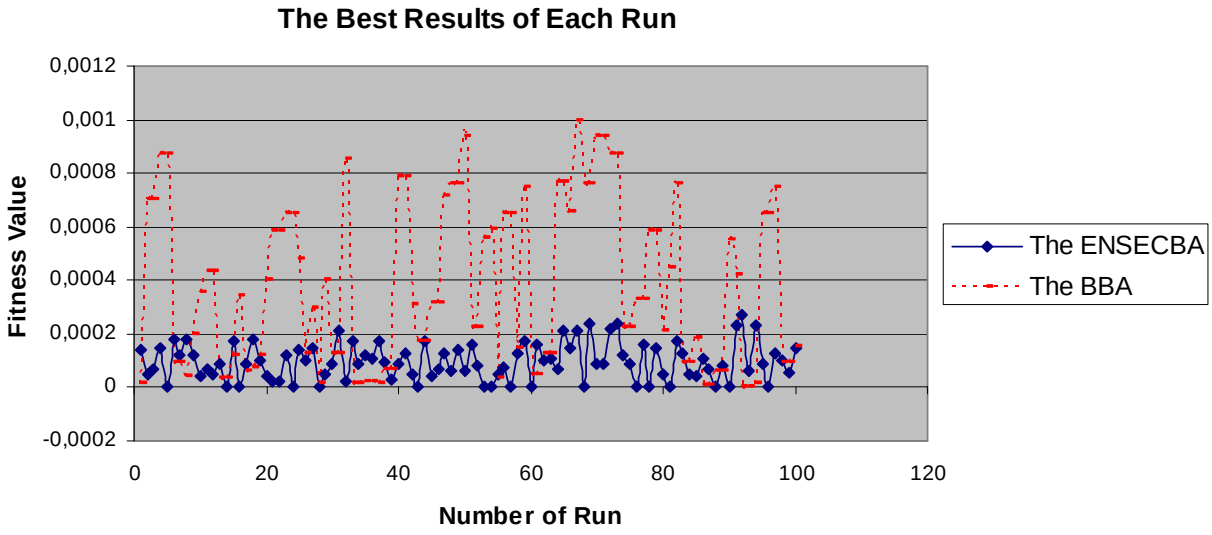

Figure 5.17: The results of a hundred runs for the BBA and the ABA on Hyper Sphere 10D.

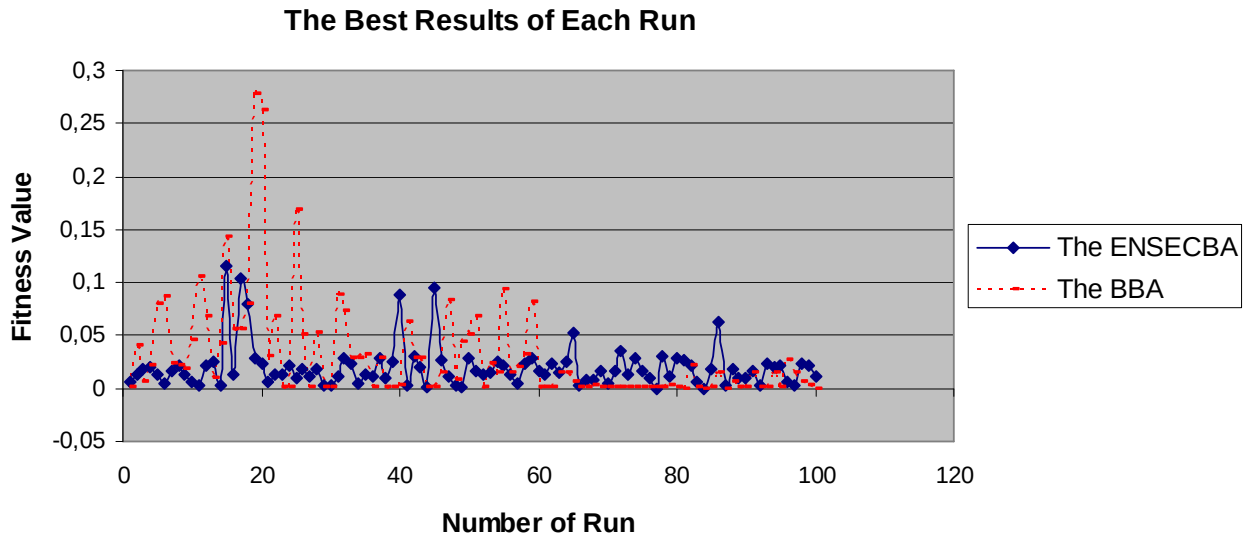

Figure 5.18: The results of a hundred runs for the BBA and the ABA on Ackley 10D.

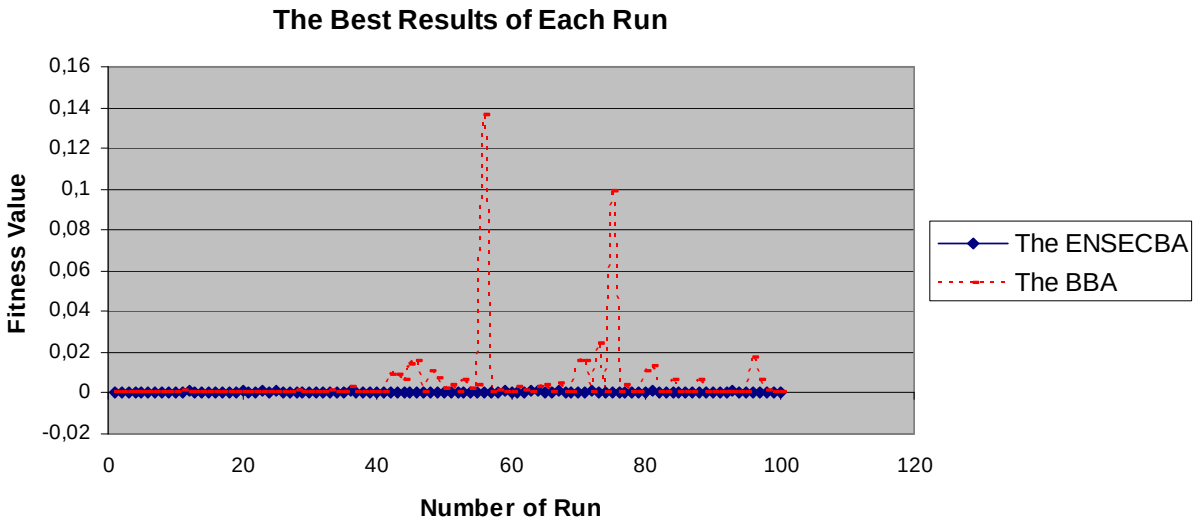

Figure 5.19: The results of a hundred runs for the BBA and the ABA on Rastrigin 10D.

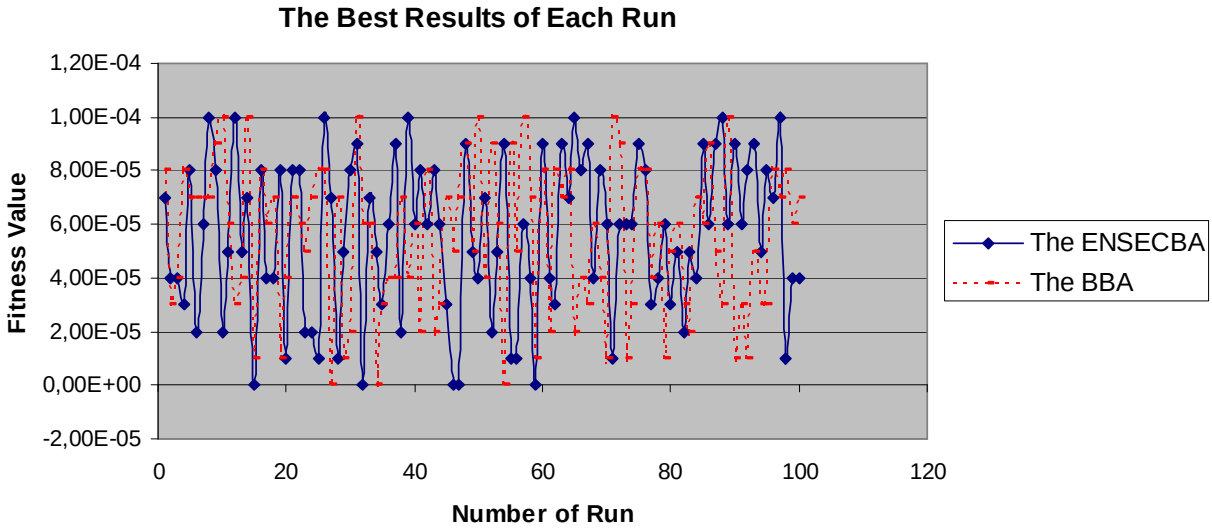

Figure 5.20: The results of a hundred runs for the BBA and the ABA on Martin and Gaddy 2D.

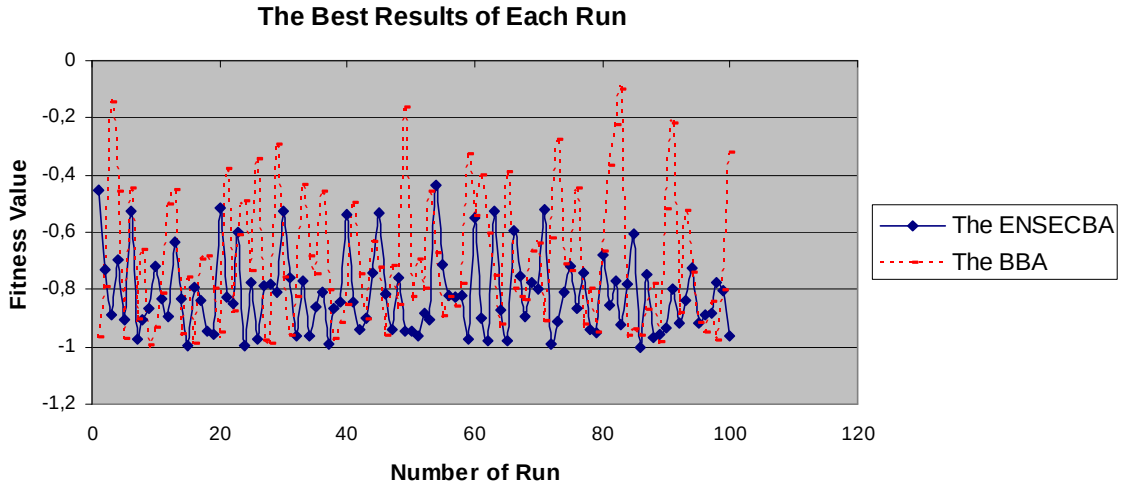

Figure 5.21: The results of a hundred runs for the BBA and the ABA on Easom 2D.

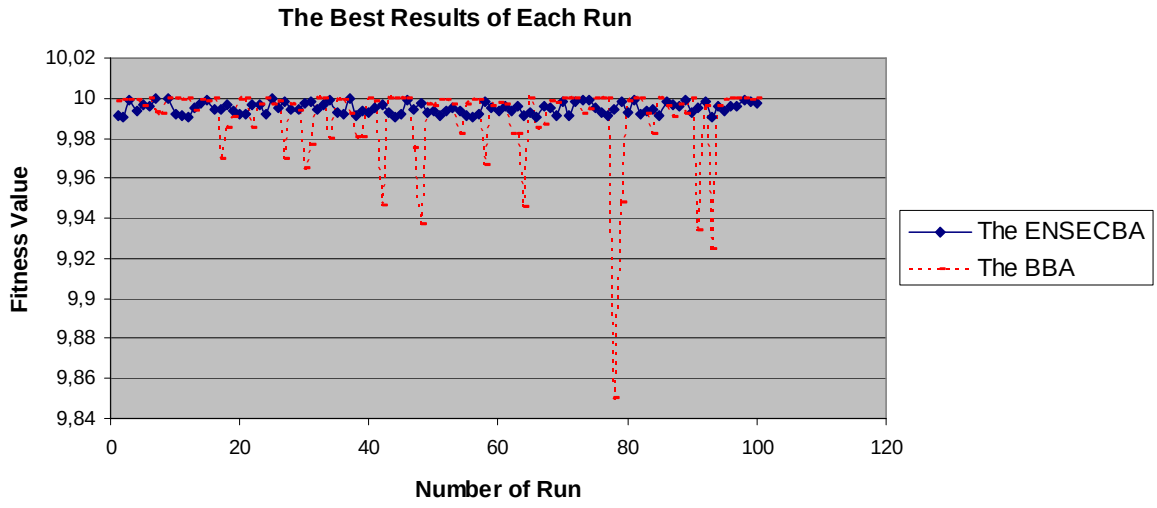

Figure 5.22: The results of a hundred runs for the BBA and the ABA on Griewank 10D.

Moreover, Statistical analysis has been carried out using t-test. The confidence level was selected to be 95 % ($\alpha <$ 0.05). T- test results are illustrated in table 5.3. According to results the ABA is more significant than the BBA on most benchmark functions. Which means the ABA is better than the BBA

No.	Function	Significance between the Basic Bees Algorithm and the Autonomous Bees Algorithm	
		Significant (α<0.05)	α
1	Goldstein & Price (2D)	Yes	4,09009E-10
2	Schwefel (2D)	Yes	6,89091E-11
3	Schaffer (2D)	Yes	6,23132E-74
4	Rosenbrock (10D)	No	0,06113
5	Sphere (10D)	Yes	3,28918E-14
6	Ackley (10D)	No	0,06612
7	Rastrigin (10D)	Yes	0,0111
8	Martin & Gaddy (2D)	No	0,72923
9	Easom (2D)	Yes	7,74352E-05
10	Griewank (10D)	Yes	0,0132

Table 3.5: The statistical analysis between the Autonomous Bees Algorithm and the Basic Bees Algorithm.

5.5 Summary

In this study, the Autonomous Bees Algorithm was presented. The aim of the research was to create an independent version of the BBA where there is no need to tune the initial parameters manually.

The proposed algorithm has been successfully tested on continuous type benchmark functions and the results observed were compared with the results obtained from the experiment on the Basic Bees Algorithm. Results of the experiments proved that the ABA can autonomously tune parameters without human interaction and produce at least similar or better results than The Basic Bees algorithm.

All experimental results were illustrated in the previous section. Moreover, statistical analysis has been employed using t-test and the results have been shown in this chapter.

Chapter 6

Conclusion and Future Work

6. Conclusion

This chapter summarises the main contributions and conclusions of this study. It also provides suggestions for the future work.

6.1 Contributions

This study has introduced new enhancements to the Bees Algorithm. The following enhancements are given below:

- Early neighbourhood search to improve initialisation stage of the Bees Algorithm.

- Efficiency based recruitment for the neighbourhood search to improve performance of the algorithm on high dimensional problems.

- Hybridisation of the Tabu search and the Bees algorithm to provide memory for the Bees Algorithm to decrease number of evaluations.

- Novel strategy to escape from local patches with similar fitness values

- Provide autonomy for the Bees Algorithm to minimise the human interaction with the search process.

6.2 Conclusions

The objectives stated in chapter one have all been achieved.

This thesis has proposed three enhanced the Bees Algorithms. Each new algorithm was tested on continues type benchmark functions. Further statistical analysis was carried out using T-test. All experimental results were provided in related chapters. The conclusions are given below:

1. Early neighbourhood search and efficiency based recruitment for the neighbourhood search were utilised to create new version of the Bees Algorithm which was called the Early Neighbourhood Search and Efficiency-based Recruitment Bees Algorithm (ENSEBRBA). Proposed algorithm was tested on ten different types of continues benchmark functions. Results were assessed based on average absolute difference technique and average number of evaluations. From experimental results it can be concluded that performance of the Bees Algorithm on high dimensional problems was improved due to proposed modifications. However, performance of the proposed algorithm was not satisfactory on easy low dimensional benchmark functions. This can be related with high computational calculation of the efficiencies of each best patch. Such calculations are not necessary for "easy" problems. Thus it will only increase the number of evaluations. The proposed enhancements improved the overall performance of the algorithm. Moreover results of statistical analysis proved that the proposed algorithm is significantly better than the Basic Bees Algorithm. First and second objectives described in chapter 1 were achieved by using ENSEBRBA.

2. The Hybrid Tabu Bees Algorithm (TBA) was proposed by combining the Tabu search and the Bees Algorithms. This is first version of the Bees Algorithm which utilises the memory unit. Moreover new strategy to escape from locals with similar fitness values. The new algorithm was also tested on continues type benchmark functions and the results were compared with the BBA and the ENSEBRBA. Experimental results were again assessed based on average absolute difference and average number evaluations. According to the generated results the proposed modifications decreased the number of evaluations needed for the Bees Algorithm go converge to the global optimum. Although the TBA was introduced to decrease number of evaluations, it also improved accuracy of the Bees Algorithm. Utilised t-test proved that proposed algorithm is significantly better than the Basic Bees Algorithm. The third and forth objectives met by proposing the Hybrid Tabu Bees Algorithm.

3. Concept of autonomy was utilised to develop version of the Bees Algorithm where interaction between user and the search process was minimised. The proposed algorithm was called the Autonomous Bees Algorithm (ABA). The proposed algorithm was also tested on continues type benchmark functions. The generated results were compared to the results of the BBA. The experimental results were assessed based on average of global optimums and number of evaluations. Observed results proved that the ABA generated optimal parameter set and produced at least same or better results than the BBA. Moreover, t-test based statistical analysis was carried out. According to this experiment the ABA was significantly better than the BBA on seven functions out of ten. Results observed from those three functions were similar to results of the BBA. From t-test result it can be concluded that utilised autonomy not only provided the

independency to the Bees Algorithm but also improved accuracy. Objective five proposed in chapter 1 was achieved by developing the ABA.

6.3 Future work

There are a number of issues which can be investigated in order to improve the Bees Algorithm and widen its potential.

- Early neighbourhood search was introduced as a solution for the poor initialisation stage of the Bees Algorithm. However, this search was carried out in its simplest form using minimum number of recruit bees. In the future different search strategies can be applied to improve efficiency of this approach in the initialisation stage.

- Efficiency based recruitment was suggested to improve the performance of the Bees Algorithms on high dimensional problems. However, the performance of the algorithm was degraded on simple low dimensional problems due to the computational complexity. This can be investigated to find more productive approach to calculate efficiency of the patches with minimum number of evaluations.

- Different tabu list strategies can be investigated for the Hybrid Tabu Bees Algorithm.

- Various enhancements were proposed in this study. It can be investigated to have various combinations of those enhancements.

- In future, it is possible to focus on the BA parameter reduction to run the algorithm with less parameters.

- Most studies in the BA were carried out to improve the neighbourhood search stage (local search). The future research studies on the BA may focus on the global search process stage.

- The new research trend on the BA is to enhance the algorithm with hybrid approaches using Tabu Search, Genetic Algorithm and PSO. It is possible to investigate the availability of using other hybrid combinations.

REFERENCES

Ahmad S., 2012, "A study of search neighbourhood in the bees algorithm", *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Al-Jabbouli H., 2009, "Data Clustering Based on Bees and Trees", *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Back T., Fogel D. B. and Michalewicz Z., 1997, "Handbook of Evolutionary Computation (Computational Intelligence Library)", *Taylor and Francis*, New York.

Baris Y., 2012, "Novel Computational Technique for Determining Depth Using the Bees Algorithm and Blind Image Deconvolution", *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Blondin J., 2009, "Particle Swarm Optimization: A Tutorial", http://cs.armstrong.edu/saad/csci8100/pso_tutorial.pdf, August 2013.

Blum C. and Merkle D., 2008, "Swarm Intelligence-Introduction and Applications", *Springer*, Berlin.

Bramier M. F. and Banzhaf W., 2006, "Linear Genetic Programming", *Springer*, New York.

Chinneck J. W., 2000, “Practical Optimization: A Gentle Introduction”, *Carleton University*.

David P., and Alan M., 2010, “Artificial Intelligence: Foundations of Computational Agents”, *Cambridge University Press.*

Davis L., 1991, “Handbook of genetic algorithms”, *Van Nostrand Reinhold, New York.*

Dieterich J.M., Hartke B., 2012, “Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization”, *Applied Mathematics*, vol. 3, pp. 1552-1564.

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.

Dorigo M. and Stutzle T., 2004, “*Ant Colony Optimization*”, A Bradford Book The MIT Press, London.

Eberhart R. C. and Kennedy J., 1995, "A New Optimiser using Particle Swarm Theory”, *Proceedings of Sixth International Symposium on Micromachine and Human Science*, pp. 39-43.

Eberhart R. C. and Yuhui S., 2001, "Particle swarm optimization: developments, applications and resources," *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol.1, pp.81-86.

Eiben A. E. and Smit S. K., 2012, "Evolutionary Algorithm Parameters and Methods to Tune Them", *In Autonomous Search*, Eds.Hamadi Y.,Monfroy E. and Saubion F., pp. 15-36, *Springer Verlag*, Berlin.

Ernesto M., Baris Y., Alfredo L. and Michael S. P., 2013, "A Multi-Objective Optimization for Supply Chain Network Using the Bees Algorithm", *International Journal of Engineering Business Management*, InTech, DOI: 10.5772/56754.

Ghanbarzadeh A., 2007, "The Bees algorithm. A novel optimisation tool", *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Glover F., 1990, "Tabu Search, Part II – ORSA", *Journal on Computing* 2, pp. 4-32.

Goffi L. W., Ferrier G. D. and Rogers J., 1994, "Global Optimisation of Statistical Function with Simulated Annealing", *Journal of econometrics*, vol. 60 pp. 65-99.

Goldstein A.A., Price J.F., 1967, "An effective algorithm for minimization", *Numerische Mathematik*, vol. 10, pp.184-189.

Gould J.L. and Gould C.G., 1988, "The Honey Bee", *Scientific American Library*,

New York.

Jason B., 2011, “Clever Algorithms: Nature-Inspired Programming Recipes”, http://www.cleveralgorithms.com/nature-inspired/stochastic/tabu_search.html, August 2013.

Ho, C. W., Lee, K. H. and Leung, K. S., 1999, "A genetic algorithm based on mutation and crossover with adaptive probabilities," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol.1, pp.,768-775.

Karaboga D., 2005, “An idea based on honey bee swarm for numerical optimization”, *Technical Report TR06*, Erciyes University, Turkey.

Klipp D.L., 2001, “A Study in Polynomial Motion”,http://www.isa.org/

InTechTemplate.cfm?Section=Communities2&template=/TaggedPage/DetailDisplay.cfm&ContentID=7043, August 2013.

Koc E., 2010, “The Bees Algorithm Theory, Improvements and Applications”, *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Koppen M., Abraham A. and Schaefer G., 2011, “Intelligent Computational Optimization in Engineering Techniques and Applications”, *Springer, Berlin*.

Koziel S. and Yang X. S., 2011, “Computational Optimization, Methods and

Algorithms", *Springer, Berlin*.

Kristoffersen T. K., 2007, "Stochastic programming with applications to power Systems", *PhD Thesis,* University of Aarhus.

Land H. and Doig A. G., 1960, "An automatic method of solving discrete programming problems", *Econometrica* 28 (3), pp. 497–520.

Lee J.Y., 2010, "Multi-Objective Optimisation Using the Bees Algorithm", *PhD Thesis,* Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Li J. and Rhinehart R.R., 1998, "Heuristic Random Optimisation", *Computers and Chemical Engineering* 22(3), pp. 427-444.

Li L. and Liu F., 2011, "Group Search Optimisation for Application in Structural Design", *Springer*, Berlin.

Meng Z., 2007, "Autonomous genetic algorithm for functional optimization", *Progress In Electromagnetics Research* 72, pp 253-268.

Mishra S., 2009, 'Engineering Economics and Costing", *PHI Learning Private Limited*, New Dehli.

Molga M., 2005, "Test functions for optimization needs", http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, August 2013.

Onwunalu, J. E. and Durlofsky L. J., 2010, "Application of a particle swarm

optimization algorithm for determining optimum well location and type ", *Comput. Geosci*. 14, pp.183–198.

Otri S., 2011, "Improving the Bees Algorithm for Complex Optimisation Problems", *PhD Thesis*, Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Panigrahi B. K., Lim M. H. and Shi Y., 2011, "Handbook of Swarm Intelligence - Concepts, Principles and Applications", *Springer*, Berlin.

Passino K. M. and Seeley T. D., 2006, "Modeling and analysis of nest-site selection by honeybee swarms: the speed and accuracy trade-off", *Behav. Ecol. Sociobiol.* 59(4), pp. 27-42.

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

Pham D. T., Darwish A. H. and Eldukhri E.E., 2009, "Optimisation of a Fuzzy Logic Controller Using the Bees Algorithm", *International Journal of Computer Aided Engineering and Technology* 1, pp. 250-264.

Pham D.T. and Castellani M., 2009a, "The Bees Algorithm: Modelling Foraging Behaviour to Solve Continuous Optimization Problems", *Proceedings of IMechE, Part C* 223(12), pp. 2919-2938.

Pham D.T. and Karaboga D., 2000, "Intelligent optimisation techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks", *Springer-Verlag, London*.

Pham D.T., Castellani M. and Fahmy A.A., 2008, "Learning the Inverse Kinematics of a Robot Manipulator Using the Bees Algorithm", *Proceedings of INDIN 2008*, pp. 493-498.

Pham D.T., Ghanbarzadeh A., Koc E. and Otri S., 2006d, "Application of the Bees Algorithm to the training of radial basis function networks for control chart pattern recognition", *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering, CIRP ICME*, Ischia, Italy, pp. 711-716.

Pham D.T., Ghanbarzadeh A., Koc E., Otri S., Rahim S. and Zaidi M., 2005., "The Bees Algorithm", *Technical Report: MEC 0501*, Manufacturing Engineering Centre, Cardiff University.

Pham D.T., Ghanbarzadeh A., Koc E., Otri S., Rahim S. and Zaidi M., 2006a, "The Bees Algorithm: A Novel Tool for Complex Optimisation Problems", *Proc. 2nd Int. Virtual. Conf. on Intelligent Production Machines and Systems (IPROMS 2006), Elsevier Oxford*, pp: 454-459.

Pham D.T., Koc E., Ghanbarzadeh A. and Otri S., 2006c, "Optimisation of the Weights of Multi-Layered Perceptrons Using the Bees Algorithm", *5th International*

Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey.

Pham D.T., Koc E., Lee J.Y. and Phrueksanant J., 2007b, "Using the Bees Algorithm to Schedule Jobs for a Machine. LAMDAMAP", *8th International Conference on Laser Metrology, CMM and Machine Tool Performance*, Cardiff, Euspen, UK, pp. 430-439.

Pham D.T., Otri S., Afify A., Mahmuddin M. and Al-Jabbouli H., 2007c, "Data Clustering Using the Bees Algorithm", *40th CIRP International Manufacturing Systems Seminar*, Liverpool, UK.

Pham D.T., Otri S., Ghanbarzadeh A. and Koc E., 2006b, "Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition", *2nd IEEE International Conference on Information and Communication Technologies: From Theory to Applications, Damascus, Syria*, pp. 1624-1629.

Rosenbrock H.H., 1960," An Automatic Method for Finding the Greatest or Least lue of a Function", *The Computer Journal*, vol. 3(3), pp. 175-184.

Rothlauf F., 2011, "Design of Modern Heuristics", *Springer*, Berlin.

Rutkowski L., 2008, "Computational Intelligence: Methods and Techniques", *Springer, Berlin*.

Schmidhuber J., and Zhao J., 1999, “Direct Policy Search and Uncertain Policy Evaluation”, *Technical Report: SS-99-07*, California: AAAI.

Seanm L., 2009, “Essentials of Metaheuristics”, http://cs.gmu. edu/~sean/ book/ metaheuristics/ , August 2013.

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

Sholedolu M.O., 2009, “Nature-inspired Optimisation: Improvements to the Particle Swarm Optimisation Algorithm and the Bees Algorithm”, *PhD Thesis,* Institute of Mechanical and Manufacturing Engineering, Cardiff University.

Shtovba, S. D., 2005, " Ant Algorithms: Theory and Applications", *Translated from Programmirovanie , Programming and Computer Software* 31(4), pp. 167–178.

Talbi E., 2009, **“**Metaheuristics: From Design to Implementation”, *Wiley.*

Tereshko V. and Loengarov A., 2005, “Collective Decision-Making in Honey Bee Foraging Dynamics”, *Computing and Information Systems Journal* 9(3), pp 1-7.

Tsubakitani S., Evans J. R, 1998, "Optimising tabu list size for the travelling salesman problem.", *Computers Ops. Res.* 25, pp.91-97.

Weise T., 2009, “Global Optimization Algorithms: Theory and Application”,

http://www.it-weise.de/projects/book.pdf, August 2013.

Wolpert D. H. and Macready W. G., 1997, “No Free Lunch Theorems for Optimisation”, *IEEE Transactions on Evolutionary Computation* 1(1), pp. 67-82.

APPENDIX A

2 dimensional graphic illustrations of the Benchmark functions are given below (Molga, 2005):

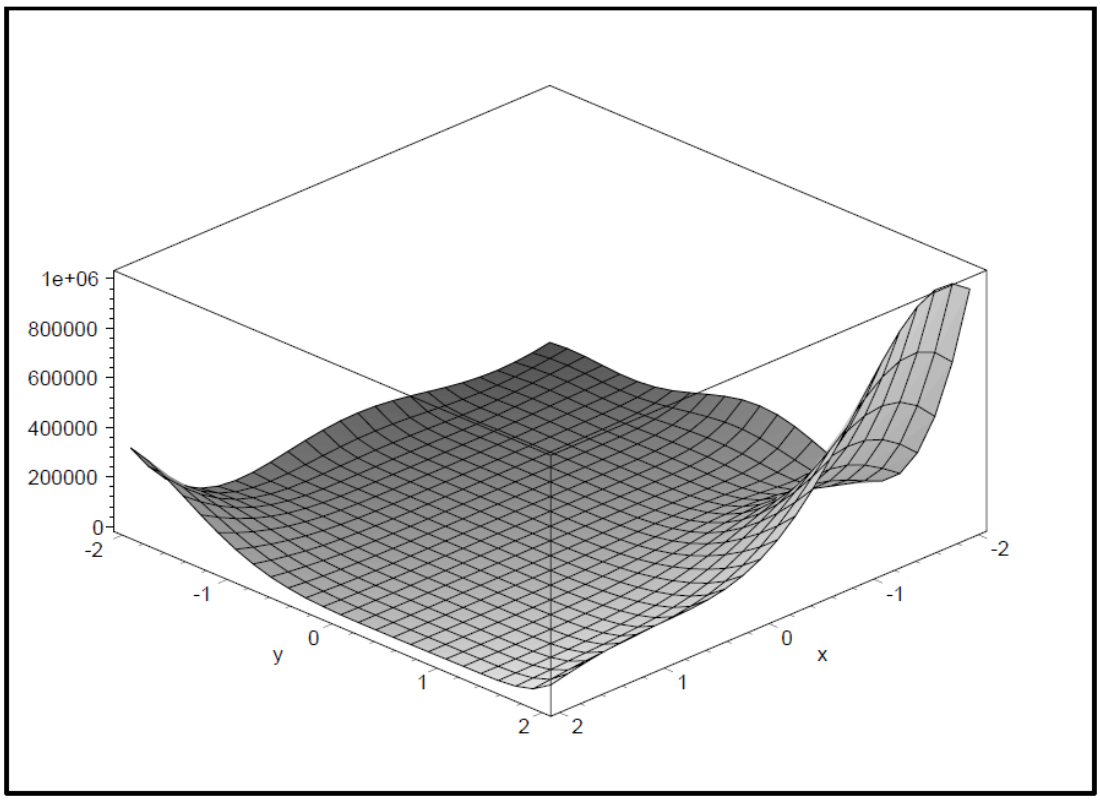

Figure A1: Graphic illustration of the Goldstein and Price's function (Molga, 2005).

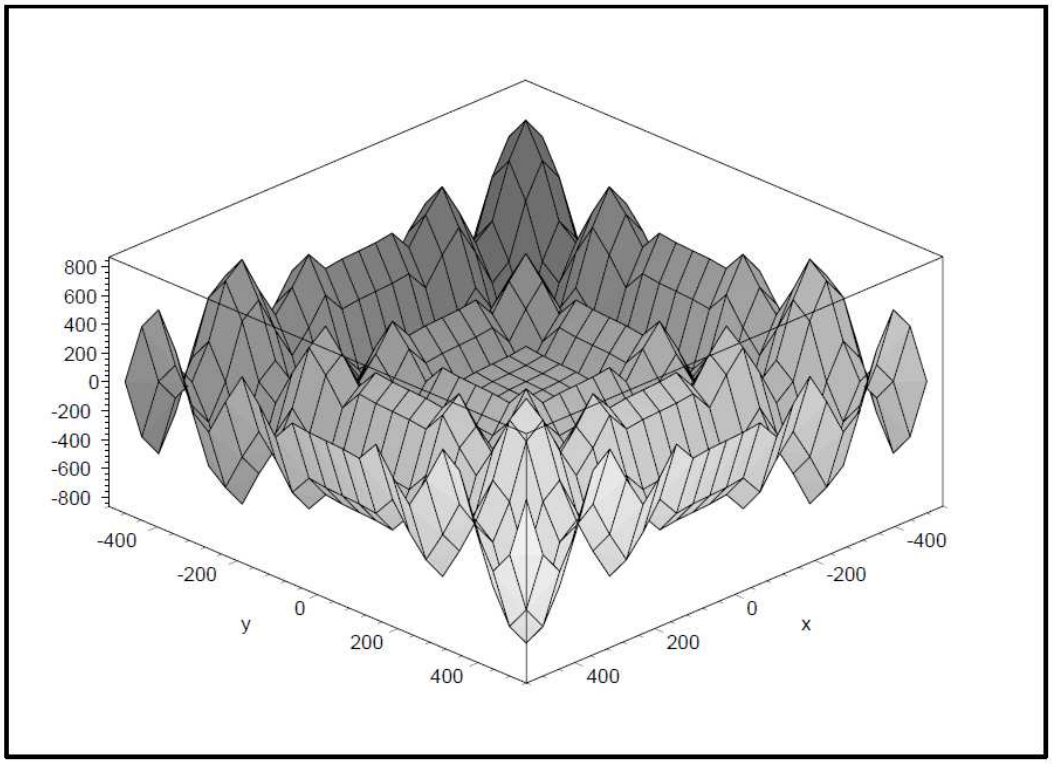

Figure A2: Graphic illustration of the Schwefel function (Molga, 2005).

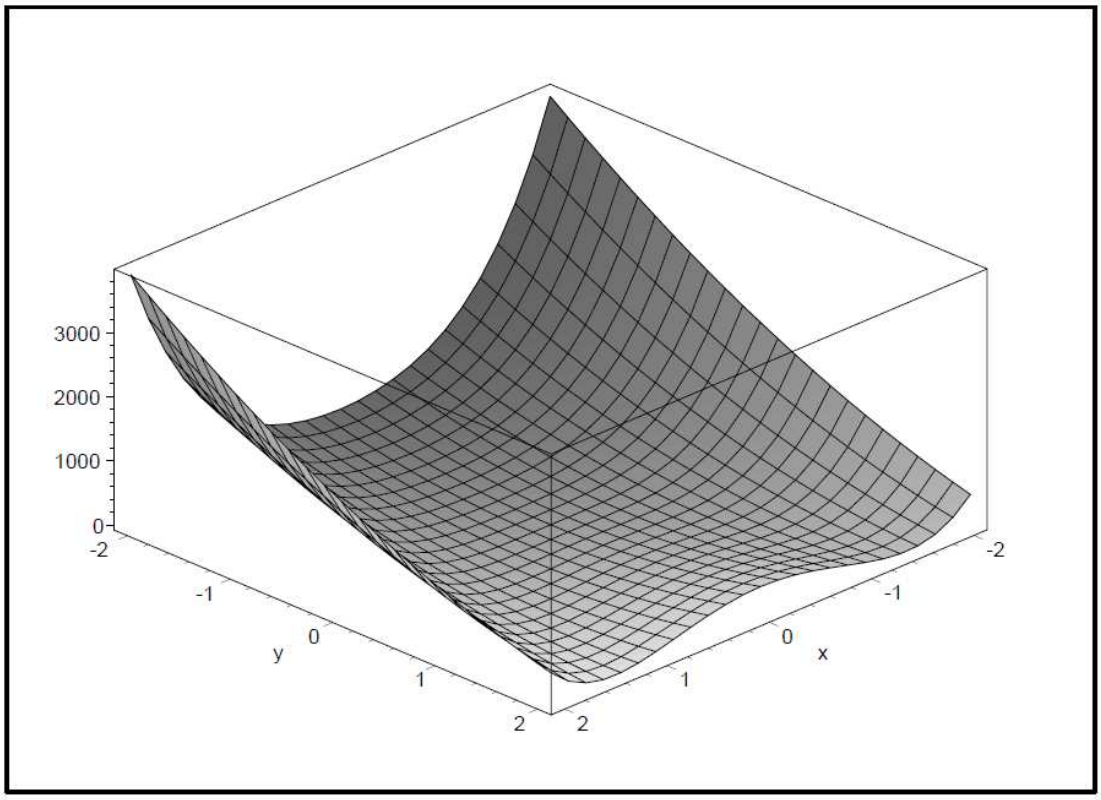

Figure A3: Graphic illustration of the Rosenbrock function (Molga, 2005).

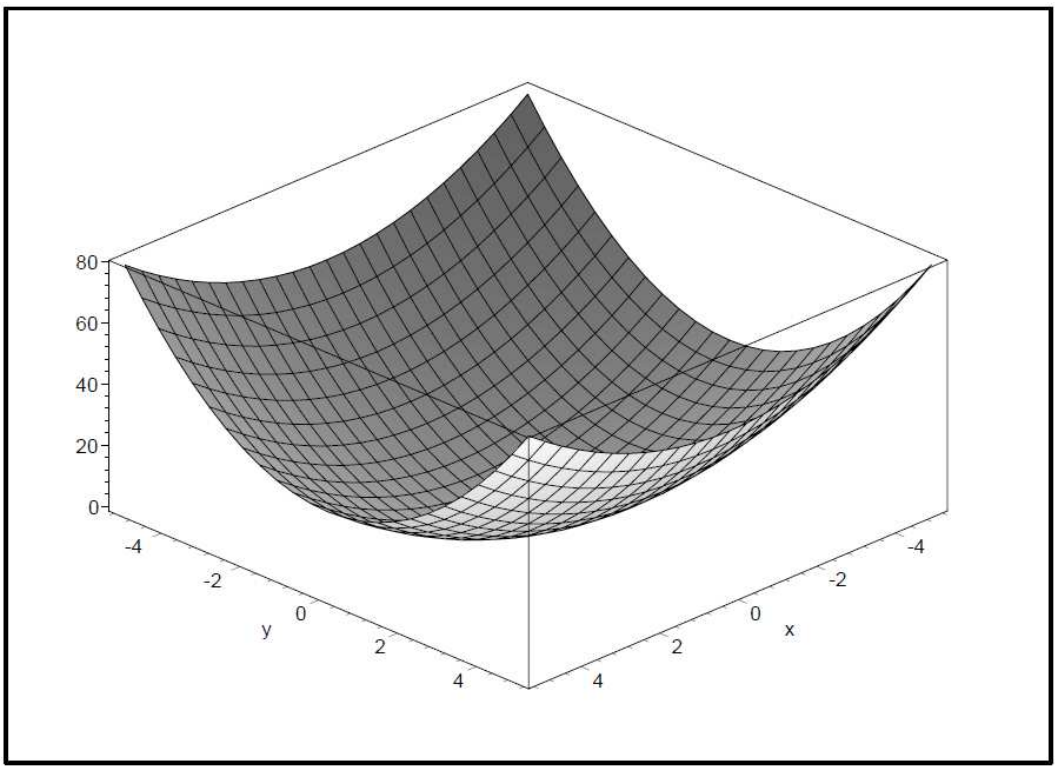

Figure A4: Graphic illustration of the Hyper Sphere function (Molga, 2005).

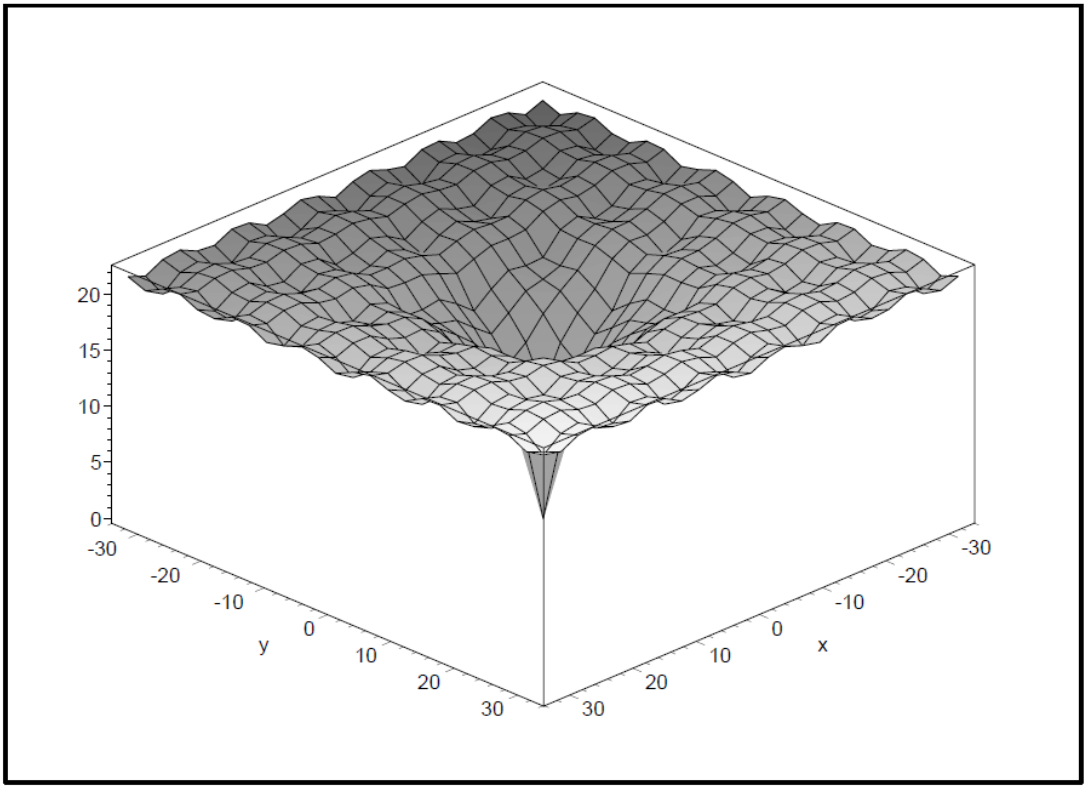

Figure A5: Graphic illustration of the Ackley function (Molga, 2005).

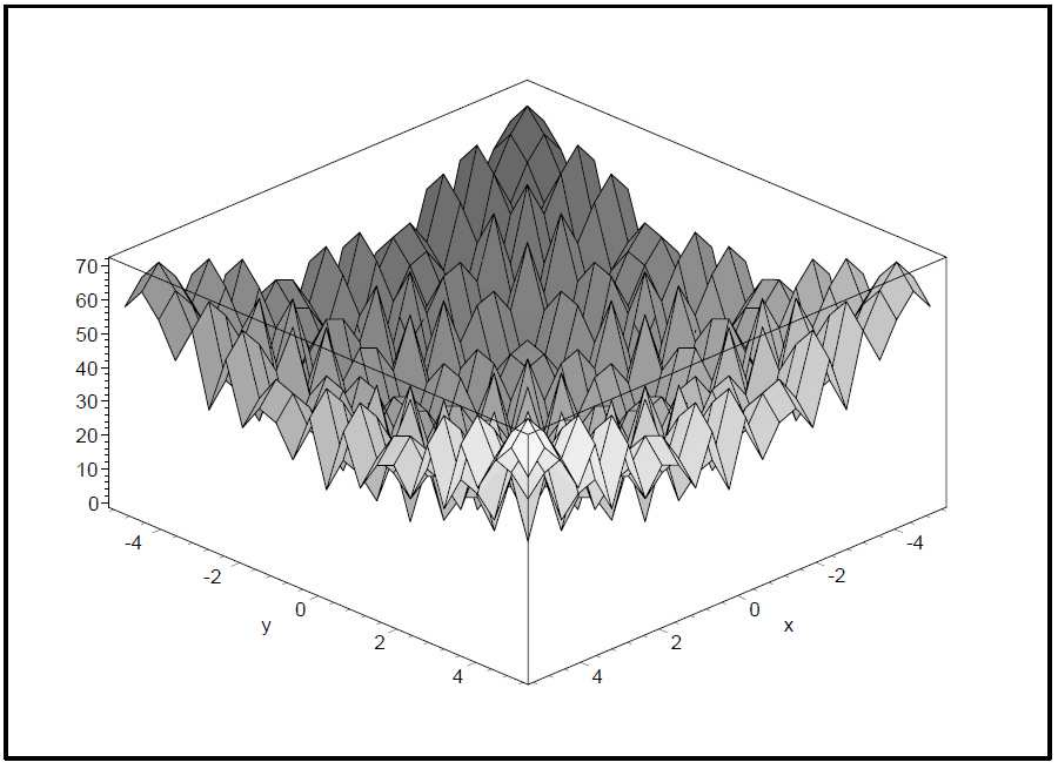

Figure A6: Graphic illustration of the Rastrigin function (Molga, 2005).

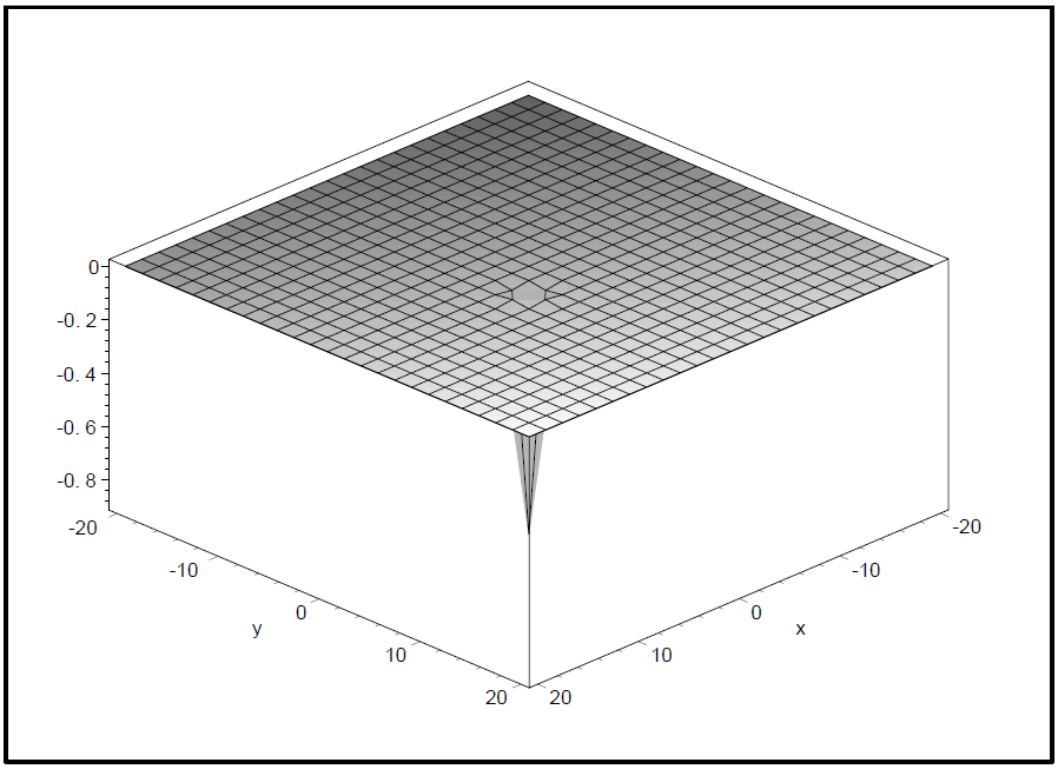

Figure A7: Graphic illustration of the Easom function (Molga, 2005).

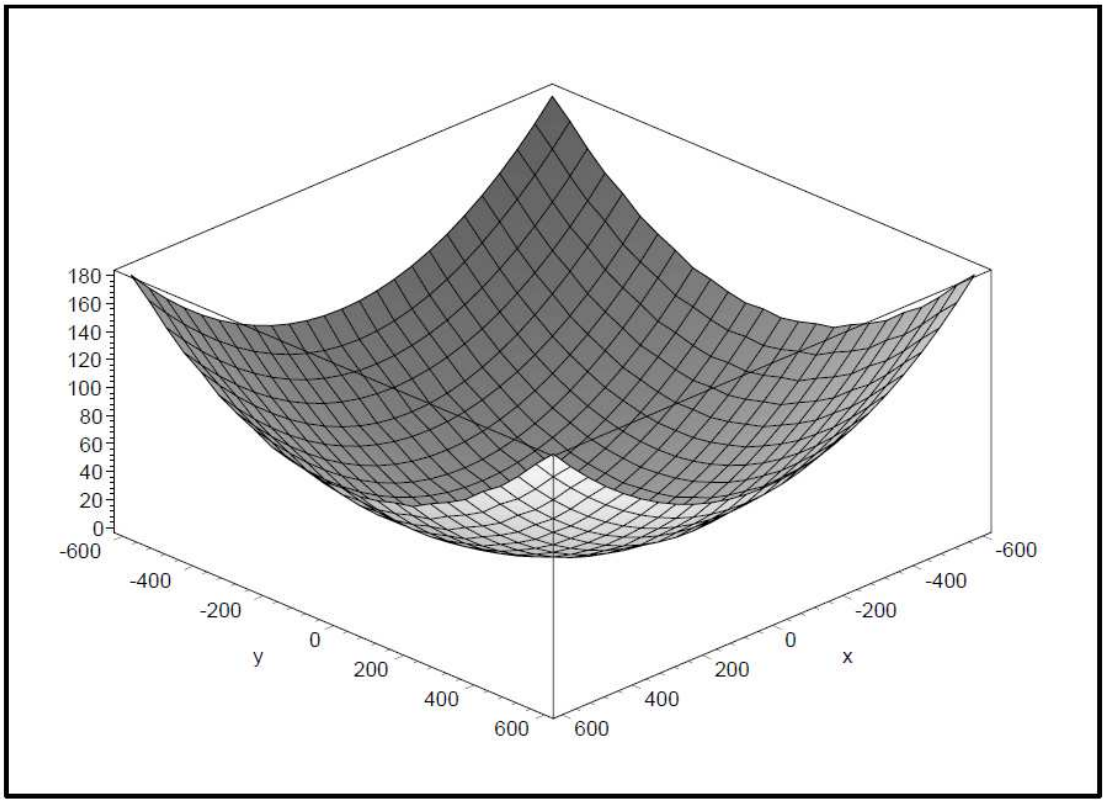

Figure A8: Graphic illustration of the Griewank function (Molga, 2005).