Bats Echolocation-Inspired Algorithms for Global Optimisation Problems

by

Nafrizuan Mat Yahya, PhD.

وَكَأَيِّن مِّن دَابَّةٍ لَّا تَحْمِلُ رِزْقَهَا اللَّهُ يَرْزُقُهَا وَإِيَّاكُمْ وَهُوَ السَّمِيعُ الْعَلِيمُ

"And how many a creature carries not its [own] provision.

Allah provides for it and for you. And He is the Hearing, the Knowing."

- Al Quran, 29:60

Acknowledgements

In the name of God, the Most Gracious, the Most Merciful.

First and foremost, praise and thanks due to Almighty الله the sole Lord of the universe, whose mercy and blessings have constantly had bestowed upon the author. Peace and blessings are upon His final Messenger Muhammad (صلى الله عليه وآله وسلم).

Second, it gives a great pleasure to the author to appreciate and thank the PhD supervisor, Dr M. Osman Tokhi from The University of Sheffield for his invaluable guidance, assistance and support throughout the research. Under his supervision, many aspects regarding the research have been explored. Moreover, the knowledge, ideas and support received from him ultimately lead this work into its present form.

Third, it gives a great gratification to the author to appreciate and thank the ultimate mentor, Dr Wan Azhar Wan Yusoff. He has been and continues to be a constant source of inspiration, motivation and strength.

Finally, it gives a great contentment to the author to appreciate and thank mother, late father, mother-in-law and father-in-law for their support and prayers over years. Special thanks are due to lovely wife, Azlyna, and brilliantly sons, Nafri Nur Ismael and Nafri Nur Ielyas for their extreme patience, understanding and sacrifices.

Thank you.

Table of Contents

Chapter 1

Introduction

A quote by George Bernhard Dantzig [1] :

`"True optimisation is the revolutionary contribution of modern research to decision processes".`

Optimisation according to the definition of Merriam-Webster Dictionary is *an act, process, or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible*.

In general, optimisation is the process of obtaining either the best minimum or maximum result under specific circumstance. The optimisation process engages with defining and examining objective or fitness function that suits some parameters and constraints. Nowadays, a vast range of business, management and engineering applications utilise the optimisation approach to save time, cost and resources while gaining better profit, output, performance and efficiency.

Optimisation problems can be divided into two categories: continuous and combinatorial (discrete). A combinatorial optimisation problem has a finite number of solutions but this is not in the case with a continuous optimisation problem where the number of solutions is infinite. This book concentrates only on continuous optimisation problems. So, here optimisation will refer solely to continuous optimisation problems.

Normally, the optimisation problems can further be classified into two major types namely; single objective optimisation and multi objective optimisation. Naturally, solving a single objective optimisation is about finding an optimised solution to the problem at hand based on the single objective. Multi objective optimisation, on the other hand, is multifaceted and solving the problem is to seek compromised solutions based on a set of conflicting objectives. As there will be no unique solution to a multi objective optimisation problem, a set of 'trade-off' solutions, referred to as Pareto optimum solutions, compromising the objectives is produced. As addition, multi objective optimisation with at least four or more objectives are often referred to as many objective optimisation, although a few researchers specified three objectives also as many objective optimisation.

Meanwhile, the single objective optimisation can be designated as either unconstrained or constrained depending on whether or not the problem contains constraints. The unconstrained single objective optimisation problem (or widely known as single objective optimisation problem) is a problem that has no constraints specified on the variables and usually is less complicated. However, a constrained single objective optimisation problem (or widely referred as constrained opti-

[1]George Bernhard Dantzig (November 8, 1914 – May 13, 2005) was a famous American mathematical scientist who made important contributions to operations research, computer science, economics, and statistics.

misation problem) comes with lack of explicit mathematical formulation but has discrete definition domains, mixed with continuous and discrete design variables and also strong nonlinear objective functions with multiple complex constraints.

Over the past forty years, many techniques have been established to solve different optimisation problems efficiently. Many optimisation problems work with mathematical or numerical linear and nonlinear programming methods and use simple and ideal models to get the optimum result. However, the numerical optimisation method tends to improve the solution locally which is different from a real world problem, often more complex and unpredictable. In addition, due to their computational drawbacks, plus the requirement of substantial gradient information, traditional numerical programming strategies have been incapable of solving any optimisation problem consistently.

Due to stated limitations and other downsides, the alternative prospect to solve an optimisation problem is by heuristic[2] or metaheuristic[3] method. Even though the metaheuristic methods are computationally laborious and give no guarantee of the quality of the results, the methods are still in the top ranking of optimisation solving tools. Metaheuristic methods offer significant advantages such as; easy to develop and implement, with a broad range of applicability, able to give a global perspective to the problem domains that are needed to be solved and the convergence rate of the global or nearly global optimum results are better than other optimisation approaches.

For the past decades, evolutionary algorithms that are part of metaheuristic methods have become popular among the researchers to deal with the complexity of a wide variety of single and multi objective optimisation problems. Evolutionary algorithms have been derived from a combination of a set of rules or restrictions and randomness by populations in generations. Evolutionary algorithms imitate or simulate the successful characteristics of natural phenomena of physical systems (e.g. simulated annealing algorithm) or biological systems (e.g. animal behaviours-based algorithms).

Evolutionary algorithms offer some advantages. The major advantages of evolutionary algorithms are that they are very good in general applicability that cover the vast range of problems as well as prior knowledge of the problem considered as inessential. An evolutionary algorithms only needs an explicit or implicit objective function to optimise the problem. An evolutionary algorithm kicks off with some guessed solutions, updates solutions in a synergistic manner then navigates the search agents to balance between exploitation of good found-so-far positions and exploration of new anonymous search positions toward the optimum global solution. The evolutionary algorithms are divided to some sub-fields. The subfields include genetic algorithm (GA) by Holland in 1975, evolutionary strategy (ES) by Rechenberg in 1965, evolutionary programming (EP) by Fogel et al. in 1966, genetic programming (GP) by Koza in 1992 and differential evolution (DE) by Storn and Price in 1995.

Among most popular evolutionary algorithms that have already captured the attention of researchers today are swarm intelligence algorithms. Swarm intelligence algorithms are inspired by the collective behaviour of swarms through a complex interaction between individuals and their neighbourhood with nature such as a colony of ants, bacteria, bees, bats, birds and fishes . In general, swarms have self-organisation and decentralised control features and all the swarm follows the same system where a population of swarm cooperates and interacts with each other in the group and the environment under certain rules during foraging or socialising . The most remarkable features of any swarm intelligence algorithm are that it has advantages of memory, diverse multi-characters capability, rapid solution improvement mechanism and is adaptable to internal and external changes .

There are some well-known swarm intelligence algorithms developed over the past two decades. In 1995, Kennedy and

[2]a way of trial and error to produce acceptable solutions to a complex problem in a reasonably practical time.
[3]*meta* means 'beyond' or 'higher level' and generally perform better than simple heuristic.

Eberhart pioneered particle swarm optimisation (PSO) algorithm that simulates the social behaviour and choreography of a bird flock. It was followed by ant colony optimisation (ACO) algorithm by Dorigo in 1999. The algorithm simulates the activity of ants while seeking a path to a food source. In micron scale of swarm intelligence algorithms, the characteristics and behaviour of the vertebrate immune system have led Hofmeyr and Forrest to introduce an artificial immune system (AIS) algorithm in 2000. In the same year, Passino successfully imitated the social foraging behaviour of *Escherichia coli* ($E.-coli$) for search of nutrients with the bacterial foraging optimisation (BFO) algorithm.

In 2007, the artificial bee colony (ABC) optimisation method that was modelled from a colony of bee raised attention of research community after explored by Karaboga and Basturk. Then, in year 2008, Havens initiated roach infestation optimisation (RIO) algorithm that was inspired from social characteristics of an intrusion of cockroaches. Later, Yang introduced bat algorithm (BA) in 2010 which imitated the echolocation of bats to find prey with different levels of pulse and loudness emitted. The algorithm was the third from Yang after a cuckoo search (CS) algorithm encouraged from compellation of social parasitism practised by a group of cuckoo and the firefly algorithm (FA) idealised from the flashing behaviour of fireflies. Both algorithms introduced a year before.

In 2012, Tawfeeq utilised the concept of echolocation of bats to find prey to design a new swarm intelligence algorithm. Different from the algorithm investigated previously by Yang, this algorithm models the principles of bats sonar used in echolocation to search for the optimum solution to a specific problem. It is worth mentioning, to strengthen the swarm intelligence algorithms or to cater for a specific problem, the versions of swarm intelligence algorithms hybridised between each other or with other conventional approaches have also existed.

Chapter 2

Optimisation problems in brief

2.1 Optimisation problem

Optimisation theory is a division of mathematics about the study of techniques, methods, procedures or algorithms to find the optimum solution to the problem considered. The optimisation problem takes place in most disciplines. In the engineering field, optimisation problem occurs in modelling and characterising; design of devices, circuits and systems; design of tools, instruments and equipment; design of structures and buildings, production planning and scheduling, quality, inventory and process control as well as maintenance and repair of equipment or systems.

The optimisation process starts by formulating the problem first. A performance criterion F must be derived in terms of n variables $x_1, x_2, \ldots, x_n$ as:

$$\text{Optimise } F = F(x_1, x_2, \ldots, x_n) \tag{2.1}$$

F is usually referred to the objective function or fitness function or cost function that can be assumed in numerous forms. It can be the cost of a product in a manufacturing environment or the difference between the desired performance and the actual performance in a system. On the other hand, the variables $x_1, x_2, \ldots, x_n$ need to be adjusted in such a way to optimise F. Variables $x_1, x_2, \ldots, x_n$ are the parameters that influence the product cost in the first case or the actual performance in the second case. They can be independent variables, decision variables, design variables or control parameters.

The word ‘optimise’ is more specific than the word ‘improve’ in delivering the meaning to achieve an optimum. In accession to that, ‘optimum’ is a technical term appropriately referring to quantitative measurement and is better than daily-use-word ‘best’. Subject to circumstances, optimise is taken to mean ‘minimise’ or ‘maximise’.

2.2 Single objective optimisation problem

2.2.1 Background

A single objective optimisation is an objective function of n numbers of variables (x) that tie to lower bound and upper bound variables as:

$$\begin{aligned} &\text{Optimise } F(x),\ x = (x_1, x_2, \dots, x_n) \\ &\quad \text{where} \\ &\qquad x_i^{(L)} \le x_i \le x_i^{(U)},\ i = 1, 2, \dots, n \end{aligned} \tag{2.2}$$

Here $x_i^{(L)}$ represent the lower bounds and $x_i^{(U)}$ represent the upper bounds of variable x_i with n variables respectively.

The purpose of an optimisation algorithm is to find a solution of variable(s), x for which the function $F(x)$ is optimum. There are two categories of solutions:

1. Local optimum solution: A point or solution x^* is said to be a local optimum solution if there exists no point in the neighbourhood of x^* which is better than x^*. For minimisation problem, a point x^* is a local minimum solution if no point in the neighbourhood had a function value smaller than $F(x^*)$.

2. Global optimum solution: A point or solution x^{**} is said to be a global optimum solution if there exists no point in the entire search space which is better than the point x^{**}. Similarly, a point x^{**} is a global minimum solution if no point in the entire search space has a function value smaller than $F(x^{**})$.

There are three major techniques available to solve the single objective optimisation problem. The first technique is calculus-based techniques or numerical methods. This technique uses a set of local requisite and sufficient condition to satisfy the solution of problem. Example of algorithms that use this technique are direct search methods and indirect search methods. The technique is excellent to solve a small class of unimodal problems but is inefficient to apply to many real life applications.

Enumerative techniques are the second set to solve the single objective optimisation problem. These techniques evaluate each and every point in the search space to arrive at the optimum solution. Most of the algorithms that apply these techniques, for example, dynamic programming, will break the considered problem into a smaller size and lower complexity because it is difficult to search all the points in the search space.

The guided random techniques are other techniques to solve single objective optimisation problems. These techniques are enumerative methods-improved where additional information about the search space is used to lead to potential solution points. The randomly guided techniques are further classified into single-point search and multi-point search. Swarm intelligence algorithms as part of evolutionary algorithms utilise the multi-point search where a highly explorative searching process with a random choice of parameters are adopted to search for several points at a time. These robust techniques have advantages to find acceptable near-optimum solution of the problems that have large search space, and are multimodal and discontinuous.

2.2.2 Approaches for single objective optimisation problems

In general, the algorithms used to solve for single objective optimisation problems can be divided into two categories, namely direct methods and gradient-based methods. Direct methods are solely depend on the objective function values to guide the search process and do not utilise any derivative information of the objective function. Meanwhile, gradient-based methods utilise derivative information (either first or second-order) to guide the search process.

Swarm intelligence algorithm which is a type of direct methods has attracted a lot of attention of the research community. There are many swarm intelligence algorithms that have been introduced over the last decade for solving single objective optimisation problems. In 2005, Yang introduced a virtual bee algorithm (VBA) that simulates the swarm interactions of social honey bees. The VBA follows the process of bees searching for honey as: a bee finds a food source; brings back honey to the hive; recruits others by performing 'waggle dance'; recruits bee to learn the distance and direction from the dance; forages the same source and becomes the favoured path. The performance of VBA has been compared with genetic algorithm (GA) to optimise De Jong's test function and Keane's multi-peaked bumpy test function. The results suggested that VBA works better than GA due to the parallelism factor inside the algorithm.

Then, in 2007, Yang has proposed a hybrid algorithm of PSO and GA to solve single objective optimisation problems. The hybrid algorithm is a combination of the flying behaviour of particles and population diversity of PSO that are enhanced by the genetic mechanism of GA. The hybrid algorithm divided the searching process into two stages. The first stage utilised the PSO procedures while the second stage adopted the GA procedures. The hybrid algorithm can improve the performance of PSO and GA as well as it is able to avoid premature convergence. The hybrid algorithm was tested on three single objective optimisation benchmark test functions, namely Sphere function, Rosenbrock function and Rastrigrin function. The hybrid algorithm recorded a better performance as compared to PSO and GA.

In the same year, Karaboga and Basturk introduced an artificial bee colony (ABC) algorithm that was also inspired from a nectar searching process by a bee colony. The ABC algorithm divided bees into three groups; the employed bees go to the located food source, the onlooker bees wait on the dance area to choose a food source and the scouts bees search for food randomly. The ABC algorithm has been compared with GA, particle swarm optimisation (PSO) and hybridised algorithm of particle swarm-inspired evolutionary algorithm (PS-EA) on five high dimensional multi modal single objective optimisation benchmark test functions. The simulation results concluded that the ABC algorithm can escape from local optimum as well as can be used to solve multivariable and multi modal function optimisation problems.

Next, Havens has introduced roach infestation optimisation (RIO) algorithm in 2008 that motivated from the collective and individual behaviours of cockroaches. The RIO algorithm works based on three behaviours of intrusion of cockroaches: like the darkest location, enjoy to socialise with a company of friends and periodically become hungry. The RIO algorithm was tested on eight single objective optimisation benchmark test functions. The results showed that the RIO algorithm could find global optimum as well as perform on par with PSO.

In 2009, Yang again presented a firefly algorithm (FA) that was encouraged from the unique pattern of flashing light by a swarm of fireflies. The FA was idealised from three rules; all fireflies are unisex, attractiveness is proportional to their brightness and objective function landscape determines the brightness. The performance of FA has been compared with GA and PSO on ten single objective optimisation benchmark test functions. The results indicated that FA outperformed both of the algorithms in terms of the efficiency and success rate.

In the same year, Yang also introduced a cuckoo search (CS) algorithm that was based on the obligate brood parasitic behaviour of some cuckoo species. This algorithm is also integrated with the Lévy flight behaviour of some birds and fruit flies. The CS algorithm operates based on three rules inspired by cuckoo breeding behaviour. The rules are: each cuckoo lays one egg in a random nest at a time, the best nest with the highest quality of eggs will bring forward to next generations and fixed number of available host nests. The CS algorithm has been verified and compared with GA and PSO on ten single objective optimisation benchmark test functions. The simulation results showed that CS performed better as compared to both established algorithms especially for multi modal objective functions.

Then, Kang have proposed a Rosenbrock artificial bee colony (RABC) algorithm in 2011. The algorithm has integrates a Rosenbrock's rotational direction method for an exploitation phase with original ABC algorithm for exploration phase.

The Rosenbrock method is a classical derivative-free local search technique with adaptive search orientation and size while the ABC algorithm is a swarm intelligence algorithm that is inspired from a colony of bee searching for nectar. The RABC algorithm has been tested and compared with other well-known algorithms on 41 single objective optimisation benchmark test functions taken from various literatures. The numerical results validated that the proposed algorithm demonstrated better performances in terms of robustness, convergence speed, efficiency and accuracy as compared to other algorithms.

In 2012, a new swarm intelligence algorithm, the krill herd (KH) algorithm was proposed by Gandomi and Alavi. The KH algorithm is based on the herding behaviour of krill individuals. The KH algorithm sets the minimum distances and highest density of krill herd from food as the objective function. Besides, KH algorithm also has taken movement induced by the presence of other individuals, foraging activity and random diffusion as three main factors to determine the time-dependent position of each krill. The KH algorithm has been compared with other eight algorithms to solve twenty single objective optimisation benchmark test functions. The result validated a better performance of the KH algorithm with the benchmark test functions as well as outperform other established algorithms.

A year later, Rizk-Allah has presented a hybrid algorithm of ant colony optimisation and firefly algorithm (ACO-FA) for solving single objective optimisation problems. The ACO-FA combined the advantages of both swarm intelligence algorithms where ant colony works as a global searcher and firefly colony works as a local searcher.The ACO-FA algorithm has been tested on a set of fifteen single objective optimisation benchmark test functions. The simulation results suggested that the ACO-FA algorithm demonstrated better performance for searching the global optimum solution as compared to other prominent algorithms.

Another variant of bee colony algorithm was proposed by Kumar in 2014. The algorithm was named directed bee colony (DBC) algorithm, and modelled a group decision-making process of nest site selection by a colony of honey bees. The ability of bees to perform tasks, the constant population of bees, environment of bees and information exchange process among bees are the main criteria in the DBC algorithm. The DBC algorithm was tested on nine single objective optimisation benchmark test functions of unimodal and multimodal types. The simulation results show better performance in terms of robustness and accuracy of DBC algorithm over other metaheuristic algorithms.

In the same year too, Askarzadeh has explored an algorithm inspired by bird mating strategy during mating season. The bird mating optimiser (BMO) algorithm is aimed to solve single objective optimisation problems. In BMO algorithm, the population is called *society* and in each *society* member is called a *bird* that represent a feasible solution. There are five groups of *bird*s in the society based on the real birds mating system. The groups are parthenogenetic, polyandrous, monogamous, polygynous and promiscuous. The BMO algorithm was tested on three categories of single objective optimisation benchmark test functions. The categories are unimodal functions, multimodal functions and low-dimensional multimodal functions. The simulation results showed a better performance of BMO algorithm to provide a good balance between global and local search effectively as compared to other algorithms.

Next, Campos has presented a bare bones particle swarm optimisation with scale matrix adaptation (SMA-BBPSO) aimed to solve single objective optimisation problems. This algorithm is an improved version to settle premature convergence problem suffered by the original bare bones particle swarm optimisation (BBPSO). In the SMA-BBPSO, each particle chooses new position in the search space using a multivariate t-distribution with a rule for adaptation of its scale matrix. The strategy induces accumulated learning in each particle and increases the ability of particles to escape from a local optimum.The performance of the SMA-BBPSO has been verified on fifteen single objective optimisation benchmark test functions. Statistical test results showed significant improvement of SMA-BBPSO to get good solutions for all test functions compared to other swarm algorithms.

Recently, Liang has proposed a social network-based swarm optimisation algorithm (SNSO) targeted for solving single objective optimisation problems. The SNSO algorithm adopted a social network evolution model of the swarm to improve the search performance of a swarm. The SNSO introduced a dynamical population topology, extended neighbourhood structure and divided the individuals into two groups based on their fitness. Results from computer simulation on twelve single objective optimisation benchmark test functions showed that SNSO achieved better performance as compared to seven others distinguished population-based algorithms.

Swarm intelligence algorithms based on bats have also appeared in the literature, among which significant one are bat algorithm (BA) by Yang in 2010 and bats sonar algorithm (BSA) by Tawfeeq in 2012. Both algorithms are inspired from echolocation of a colony of the bats. However, both algorithm will be detailed in the next chapter such that the BA and especially BSA will be a base for the development of a new set of bats echolocation-inspired algorithms.

2.3 Constrained optimisation problem

2.3.1 Background

A constrained optimisation comprises an objective function together with some equality and inequality constraints subject to lower bound and upper bound of variables as:

$$\begin{aligned}
&\text{Optimise } F(x),\ \ x=(x_1,x_2,\dots,x_N)\\
&\text{subject to}\\
&\qquad g_j(x)\geq 0,\ \ j=1,2,\dots,J\\
&\qquad h_k(x)=0,\ \ k=1,2,\dots,K\\
&\quad\text{where}\\
&\qquad x_i^{(L)}\leq x_i\leq x_i^{(U)},\ \ i=1,2,\dots,n
\end{aligned}\tag{2.3}$$

Here $g_j(x)$ represents inequality constraint functions with J inequality constraint. $h_k(x)$ represents equality constraints functions with K is equality constraints. $x_i^{(L)}$ represents the lower bounds and $x_i^{(U)}$ the upper bounds of variable x_i with n variables respectively.

A constrained optimisation problem deals with interferences between multi-variable and multi-constraint features. So it is difficult to solve the constrained optimisation problem as compared to unconstrained optimisation problem in a way that ensures efficient, optimum and constraint-satisfying convergence condition. If the constraints can be handled, it is easier to solve the constrained optimisation problem. The most popular way that researchers have adopted is to convert the constrained optimisation problem to be unconstrained optimisation problem by getting in all constraints into the objective function.

Another challenge in a constrained optimisation problem is how to balance the search for feasible individuals and infeasible individuals throughout the search process. Feasible individuals are the individuals that satisfy all of the equality and inequality constraints and variables bound at that point, while infeasible individuals are individuals that do not satisfy at least one of the constraints. The conventional way to solve this problem is by ignoring the existence of infeasible individuals but continue the process of discovering for optimum solution with the feasible individual only.

So, a better searching approach for the optimum solution in part with the correct constraint handling technique plays an important role to solve a constrained optimisation problem effectively.

2.3.2 Constraints handling technique for constrained optimisation problems: a penalty function method

Constraint handling techniques will be used to direct the search for the algorithm towards feasible solution in the search space. There are several major approaches reported in the literature for handling constraints. These include:

1. Approach based on the preference of feasible solutions over infeasible ones with some operators.
2. Approach based on penalty functions that use a penalty term to convert constrained optimisation problem into a non-constrained optimisation problem.
3. Approach based on multi objective optimisation concept such as Pareto ranking scheme.
4. Approach that makes a clear distinction between feasible solutions and infeasible solutions.
5. Approach based on a separate treatment of objective function and the constraint violation, for instance; stochastic ranking (SR).
6. Hybrid methods combining evolutionary computation techniques with deterministic procedures for numerical optimisation.

An approach based on penalty function as constraint handling mechanism in constrained optimisation problems has gained more attention of researchers. Besides of its simplicity and easy implementation, the key successful factor of this approach is that an individual in the infeasible region is penalised to move toward the feasible region and provide useful information to help others to move in. The penalty function also offers as leverage balance between objective function and constraint violation. Even though many variants of penalty functions exist such as dynamic penalty, adaptive penalty and death penalty, the static penalty function (which is the basic one) will be adopted here.

In the static penalty function, the original objective function $F(x)$ is replaced by a substituted function $C(x)$ which considers the original objective function $F(x)$ add a penalty function $P(x)$ that introduces a tendency term to penalise constraint violations produced by x. Therefore, considering the constrained optimisation problem defined previously, the substituted function is defined as follows:

$$\begin{gathered} C(x) = F(x) + P(x) \\ \text{where} \\ P(x) = \mu \cdot \sum_{j=1}^{J} {g_j}^2(x) + v \cdot \sum_{k=1}^{K} {h_k}^2(x) \end{gathered} \tag{2.4}$$

where μ and v represent the penalty coefficients which weigh the relative importance of each $g_j(x)$ (inequality constraint) and $h_k(x)$ (equality constraint) respectively. In this book, μ and v values are problem-dependant.

2.3.3 Approaches to solving constrained optimisation problems by previous researchers

Various research works have been reported over the past two decades on dealing with constrained optimisation problems. This section will highlight some by dividing the approaches into four main bases, namely swarm intelligence algorithms, other evolutionary algorithm strategies, hybridised approaches and multi objective optimisation methods.

There are several swarm intelligence algorithms that have been used to solve the constrained optimisation problem. The PSO was the most favourable technique among them.In 2005, Parsopoulos and Vrahatis have proposed a variant of PSO scheme, a unified particle swarm optimisation (UPSO) method with a penalty function approach. The proposed algorithm has abilities to explore and exploit the search process without needing extra requirements of function evaluations and also preserves feasibility of the encountered solutions.Then, in 2006, Yang has introduced a master-slave particle swarm optimisation (MSPSO) where master swarm and slave swarm particles were created to fly toward better feasible and infeasible particles, updating and sharing information between them. This approach brings better global exploration ability and keeps away from being trapped into the local optimum.

Next, He and Wang explored a two-swarm groups mechanism in a co-evolutionary particle swarm optimisation (CPSO) in 2007. Both groups were used to evolve decision solution and adapt penalty factors for solution evaluation. A year later, Cagnina has investigated simple constrained particle swarm optimiser (SiC-PSO) which was coupled with a constrained-handling technique. The algorithm was faster, more reliable and efficient after combining local best (*lbest*) and global best (*gbest*) models to update the velocity as well as adding *gbest* to the best position of the particles and in its neighbourhood. Next, in 2013, Afshar has designed a fully constrained particle swarm optimisation (FCPSO) and three versions of partially constrained PSO (PCPSO) to deal with the real world water resources management problems. The proposed algorithms eliminated the infeasible region in the search space before and during a search process. As compared to the original PSO, the methods are computationally effective and less sensitive to initial swarm and swarm size.

Another popular swarm intelligence algorithm used is the artificial bee colony (ABC) algorithm. For instance, in 2014, Garg has introduced a penalty function guided ABC algorithm to solve several structural engineering design problems. Before that,in 2007, Karaboga and Basturk have adopted Deb's rule for the selection of mechanism to deal with the constraints of the ABC algorithm to solve a set of constrained numerical optimisation problems. Akay and Karaboga have extended previous version by adding constraint handling technique into the selection steps to solve large scale and constrained engineering design problems in 2012. The rest of the techniques shall be categorised under swarm intelligence algorithms such as, bat algorithm (BA) by Yang and Hossien in 2012 which is based on the level and loudness of pulse emitted in bats echolocation, a bacterial gene recombination algorithm (BGRA) that was inspired from virus resistance process in real bacteria by Hsieh in 2014 and the social spider optimisation (SSO-C) algorithm which is based on the cooperative behaviour in a colony of social spiders by Cuevas and Cienfuegos in the same year.

Besides swarm intelligence algorithms, several researchers applied other evolutionary algorithms to solve the constrained optimisation problems.For instance, Koziel and Michalewicz in 1999 utilised an evolutionary algorithm with decoder (a technique that uses a chromosome as in genetic algorithm), incorporated with a homomorphous mapping between an n-dimensional cube and a feasible search space. According to the originators, the proposed algorithm was an alternative approach to nonlinear programming (NLP).

Differential evolution (DE) which is among the popular methods in evolutionary algorithms is also widely applied to constrained optimisation problems. Huang in 2008 modified the algorithm to an archived DE (ADE) algorithm where an archive of all the best solutions from previous evolution process will be utilised to estimate new solutions. The algorithm also collaborated with dynamic penalty functions and fitness calculation of individuals. Later, Li proposed an improved DE with a self-adaptive strategy to determine the control parameters paired with the dynamic constraint-handling mechanism in 2012. The approach was enhanced with a self-adaptive parameter from its original version, DE with dynamic constraint-handling (DCDE) to seek and improve for a feasible solution and objective function. In year 2014, Gong has studied an improved constrained DE variant; improve mutation dynamic DE (rank-iMDDE). The improved algorithm introduced a ranking-based mutation operator to accelerate the convergence rate of DE as well as improve the dynamic diversity mechanism to maintain feasible and infeasible solutions in the population under the multiple trail vectors generation technique.

Other research works on evolutionary algorithms to solve the constrained optimisation problems include the adaptive segregational constraint handling evolutionary algorithm (ASCHEA) by Hamida and Schoenauer in 2002 that targeted to preserve feasible and infeasible individuals and the improved α constrained simplex method by Takahama and Sakai in 2005 to control the convergence speed. To take advantage of the information hidden in infeasible individuals efficiently, a self-adaptive penalty function-genetic algorithm by Tessema amd Yen in 2006 was introduced, while an effective global harmony search (EGHS) based on natural music performance was applied to optimise pressure vessel design problems by Gao in 2010. Other techniques are teaching-learning-based optimisation (TLBO) algorithm inspired by the real influence of a teacher on learners by Rao in 2011, a novel selection evolutionary strategy (NSES) with a self-adaptive selection method by Jiao in 2013 and mine blast algorithm (MBA) inspired from bomb explosion to clear the mines field by Sadollah in 2013.

There has also been attempts on hybridisation of two or more methods to solve constrained optimisation problems. The combination techniques desire features of each so that the new breed algorithm is better than the individual components. For instance, Coello in 2000 and Amirjanov in 2006 respectively hybridised GA with another strategy to improve the capabilities of GA to solve constrained optimisation problem. Here, Coello embedded co-evolution concepts to adapt the self-adaptive penalty factors of fitness function into GA. The co-evolution was applied to create two populations that interact with each other and can also be used to determine the penalty factor and optimise the objective function. The technique is easy to implement and is suitable to use on parallelisation to improve overall performance. On the other hand, Amirjanov injected GA with the changing range of design variable feature using a stochastic ranking method with the shifting and shrinking mechanism (SSM). The algorithm would move and shrink the search space towards the feasible region resulting in speedy convergence to the global optimum within reasonable precision.

PSO also became the subject of hybridisation method for producing a new strong algorithm to solve constrained optimisation problems. In 2007, He and Wang introduced a hybrid PSO (HPSO) after combining feasibility based rule and simulated annealing (SA). SA acted as a means to shun premature convergence, while the feasibility-based rule was used as an alternative to penalty function approach for the constraint-handling mechanism. Then, in year 2009, Zahara and Kao integrated the Nelder-Mead simplex method and PSO (NM-PSO) which combine the advantage of efficient local search in Nelder-Mead method and better global search in PSO. PSO also has been paired with DE by Liu in 2010, known as PSO-DE to accelerate the convergence process. DE, which has a strong searching ability, was used to help PSO escape from stagnation condition. Deb's feasibility-based rule to compare the updated particle is used in this hybrid method.

Further, Runarsson in 2000 has introduced a μ and λ evolution strategy, (μ+λ)ES with stochastic ranking method. The algorithm plans to balance dominance of penalty function and objective function stochastically and it is achieved through a ranking procedure based on the stochastic bubble-sort algorithm. Then, in 2005 Runnarsson has improved the algorithm by exploring an improved stochastic ranking (ISR) method to show the importance of search bias in constrained optimisation. Meanwhile, in 2003, Ray and Liew have introduced a society and civilisation algorithm modelled from the intra and inter-society interactions within a formal society and the civilisation. The algorithm combined the features of GA, machine learning model and Pareto ranking scheme. Later, Becerra and Coello in 2006 investigated a cultural algorithm with a differential evolution population, where the cultural algorithm is an evolutionary computation technique that uses domain knowledge to improve process performance. Few more examples are the dynamic stochastic selection in multi-member differential evolution (DSS-MDE) by Zhang in 2008, a hybrid evolutionary algorithm and an adaptive constraint handling technique (HEA-ACT) by Wang in 2009 and a differential evolution with level comparison (DELC) by Wang and Li in 2010.

Another significant method by researchers to optimise the constrained optimisation problems is by implementing a non-constraint handling mechanism or a multi objective optimisation method. For example, Coello and Mezura-Montes in 2002 have introduced a niched-Pareto GA (NPGA) where the new constraint handling approach was introduced based on multi objective optimisation technique. This method adopts concepts from multi objective optimisation, where it does not require penalty function or niching approach to maintain diversity in the population instead of deriving a new constraint-handling technique. Next, Mezura-Montes and Coello in 2005 have researched the non-penalty function, self-adaptive mutation of a simple multi-membered evolution strategy. This strategy involved a simple diversity mechanism to keep infeasible solution in the population, a simple feasibility-based comparison mechanism to drive the process toward the feasible region, and a hybrid recombination operator was used for exploitation process in the algorithm.

In the same year, Mezura-Montes and Coello have proposed a $(\mu + \lambda)$ evolution strategy to solve engineering design problems without using penalty function. This strategy handles the objective function and constraints separately. The algorithm successfully guided the generation of solutions close to the boundaries of the feasible region to achieve a better solution, regardless of its location inside or in the boundaries of the feasible set. Then, Fei in 2010 proposed a GA that use a multi objective optimisation Pareto ranking to deal with the infeasible solutions violation constraints. Then, Wang and Cai have proposed a combined multi objective optimisation with differential evolution (CMODE) in 2012. The CMODE uses infeasible solution replacement mechanism based on multi objective optimisation that aims to drive the population toward better solutions in the feasible region concurrently. The comparison between individuals in CMODE is also run through multi objective optimisation method.

2.4 Multi objective optimisation problem

2.4.1 Background

The general multi objective optimisation problem with N objectives is formulated as:

$$\begin{aligned}
&\text{Optimise } F(x) = [F_1(x), F_2(x), \ldots, F_N(x)] \\
&\text{subject to} \\
&\qquad g_j(x) \geq 0, \;\; j = 0, 1, \ldots, J \\
&\qquad h_k(x) = 0, \;\; k = 0, 1, \ldots, K \\
&\quad \text{where} \\
&\qquad x_i^{(L)} \leq x_i \leq x_i^{(U)}, \;\; i = 1, 2, \ldots, n
\end{aligned} \tag{2.5}$$

Here $F_N(x)$ represents objective function N, $g_j(x)$ represents inequality constraint functions with J inequality constraints. $h_k(x)$ represents equality constraints functions with K equality constraints. $x_i^{(L)}$ represents the lower bounds and $x_i^{(U)}$ the upper bounds of variable x_i with n variables respectively.

The objectives in a multi objective optimisation are conflicting with one another. Therefore, a perfect multi objective solution that simultaneously optimises (minimise or maximise) each objective function is almost impossible. So, the aim is to find good compromise or trade-off solutions rather than a single solution as in single objective optimisation. A reasonable solution to a multi objective optimisation problem is a set of solutions each of which satisfies the objectives at an acceptable level without being dominated by any feasible solutions in the entire search space. The set is called Pareto optimum set and the corresponding values of the objectives form Pareto front.

This Pareto optimum concept was originally introduced by Francis Ysidro Edgeworth in 1881 and then generalised by Vilfredo Pareto in 1896. In this concept, the Pareto optimum, dominated and non-dominated points, and Pareto front are defined as:

Definition 1 Pareto optimum: Consider a point $\vec{x}$ in the feasible solution space, $X, \vec{x} \in X$. The point (a set of decision variables) is Pareto optimum if and only if there does not exist another point, $x \in X$, that satisfies $F(x) < F(\vec{x})$ and $F_i(x) < F_i(\vec{x})$ for at least one function.

Definition 2 Dominated and non-dominated points: A vector of objective functions, $F(\vec{x})$, is non-dominated if and only if there does not exist another vector, $F(x)$, that satisfies $F(x) < F(\vec{x})$ with at least one $F_i(x) < F_i(\vec{x})$. Otherwise, $F(\vec{x})$ is dominated.

Definition 3 Pareto front: The set $\vec{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$, which is composed of all the non-dominated Pareto optimum solutions that comprise the Pareto front of non-dominated solutions.

Conventionally, there are numerous methods used to solve multi objective optimisation problems. The methods can be categorised into two major groups, namely non-Pareto techniques and Pareto-based techniques. Further, the methods considered under the non-Pareto techniques are weighted sum approaches, vector evaluated genetic algorithm (VEGA), lexicographic ordering, the ε-constraint method and target vector approaches. The Pareto-based techniques consist of pure Pareto ranking, a multi objective genetic algorithm (MOGA), non-dominated sorting genetic algorithm (NSGA), niched-Pareto genetic algorithm (NPGA), Pareto archived evolution strategy (PAES) and strength Pareto evolutionary algorithm (SPEA).

The weighted sum approach is adopted in the algorithm developed in this book. Thus, other approaches and corresponding categorisations are not further discussed here.

2.4.2 Weighted sum approach

The weighted sum approach is considered under non-Pareto techniques of multi objective optimisation problems. The Pareto optimum concept is indirectly incorporated into the approach. The approach is a kind of aggregating function as it associates or aggregates all the objectives to a sole objective. This approach was inspired by the Kuhn-Tucker conditions for a non-dominated solution in the oldest mathematical programming methods for solving the optimisation problem. When comparing to other ranking approaches, the weighted sum approach is better in terms of efficiency, simple and easy to implement. Indeed, the approach is suitable to use in any traditional or modern optimisation method.

In the weighted sum approach, all objectives F_k are merged into a single objective as:

$$F = \sum_{k=1}^{K} w_k F_k$$

where (2.6)

$$\sum_{k=1}^{K} w_k = 1$$

The weights w_k are produced randomly from a uniform distribution. The weights represent the parameters and they could be varied or changed during the evolution process as sufficient diversity will enable approximating the Pareto front to an acceptable level; in reality the precision and accuracy are very hard to comply with. The weights help in finding possible solutions in Pareto optimum sets but do not give information about the importance of the objectives studied.

The weighted sums approach further divided into three types that are:

1. Conventional weighted aggregation (CWA): the weights are fixed when only one Pareto optimum point acquired per algorithm run.
2. Bang-bang weighted aggregation (BWA): the weights can be altered abruptly during the algorithm run but a Pareto optimum set obtained on single algorithm run.
3. Dynamic weighted aggregation (DWA): the weights can be steadily changed but able to produce a Pareto optimum set only on single algorithm run.

In this book, a systematically monotonic weighted sum approach which was DWA-like is adopted in the algorithm for solving multi objective optimisation problems.

2.4.3 Approaches for solving multi objective optimisation problems using particle swarm optimisation algorithm by previous researchers

Nowadays, the PSO algorithm is among the most extensively used algorithms in solving multi objective optimisation problems. An extensive review shows that over thirty different works of multi objective PSO (MOPSO) were published in the specialised literature.

In 1999, Moore and Chapman claimed that they were the first to modify the PSO algorithm for solving single objective optimisation problem version to be applied to the multi objective optimisation problem. In their work, the p-vector was altered to a list of solutions that enabled to keep track of all non-dominated solutions to comply with Pareto preference. The MPSO were tested on two multi objective optimisation problem models that were taken from specific literature and the best results acquired were highly competitive from the results presented in the source.

Then, Parsopaulos and Vrahatis tested the performance of PSO to identify the Pareto optimum set and produce an appropriate shape of Pareto front in 2002. They integrated the multi-swarm PSO with important characteristics of a vector-evaluated genetic algorithm (VEGA) and utilised the weighted sum approach. They tested the vector evaluated PSO (VEPSO) on established non-trivial multi objective optimisation benchmark test functions and showed promising results as the VEPSO was able to record a good set of Pareto optimum set.

In the same year, Coello and Lechuga presented MOPSO that used the concept of Pareto dominance. In this technique, the flight direction of a particle is defined by the Pareto dominance while the non-dominated vectors are archived and used later as guidance for other particles' flight. They reported that the performance of the MOPSO was outstanding in comparison to PAES and NSGA-II on several multi objective optimisation benchmark test functions.

In 2005, Sierra and Coello also utilised the Pareto dominance concept into the MOPSO. However, this algorithm included further three elements namely; a crowding factor, different mutation operators and ε-dominance concept. They used the crowding factor to form a second discrimination criterion, a mutation operator for dividing the swarm into three subdivision while ε-dominance concept was applied to set the size of the final solutions set. The proposed algorithm was reported to be able to approximate the Pareto front as compared to other five established algorithms.

Next, Karpat and Özel in 2007 have attempted to solve multiple objectives of turning process in a manufacturing environment using a PSO-based algorithm. First, they integrated PSO with neural network models to form a swarm intelligent neural network system (SINNS) for the purpose of defining the objective functions and setting up the parameters involved. Then, they introduced the dynamic neighbourhood PSO (DN-PSO) methodology to solve the multi objective problem of turning process.

Another significant research is by Nebro *et al.* in 2009 where they have included a velocity constriction formula in the PSO resulting in speed-constrained multi objective PSO (SMPSO) and have tested the algorithm on multi objective optimisation benchmark test functions. In that year too, Abido solved environmental/economic dispatch (EED) problem using global best and local best-redefined MOPSO. A year later, Castro-Gutierrez *et al.* solved the vehicle routing problem (VRP) used the MOPSO with improved dynamic lexicographic ordering.

Chapter 3

Bats echolocation and existing algorithms inspired from bats echolocation

3.1 The colony of bats in nature

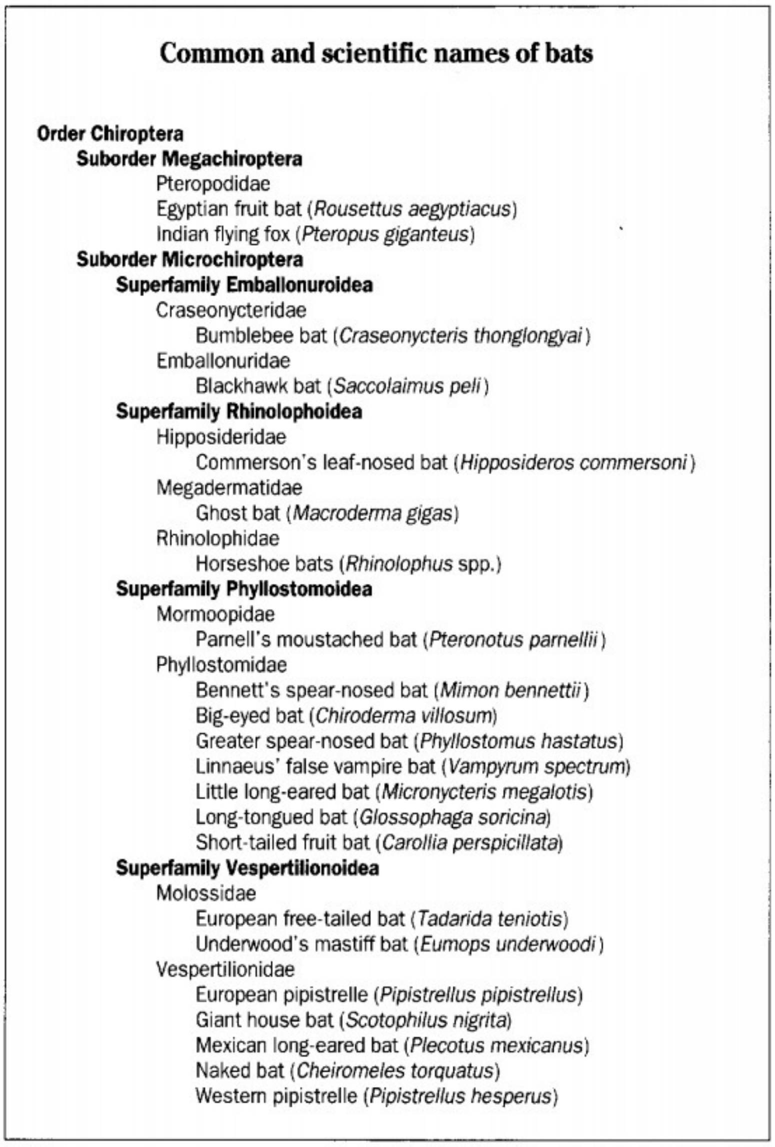
For ages, the livelihood of bats (Order *Chiroptera*) has attracted human interest. As one of the diverse and most extraordinary mammalian order, bats have more than 900 species distributed all around the world and make up almost a quarter of all mammal species. Every bat species have their unique qualities and own preference that make them special among all living creatures.

The species of bats are classified into two suborders (Figure 3.1) based on the size, namely Megachiroptera and Microchiroptera. The smallest size of microchiroptera (*e.g.* bumblebee bat) weighs only 1.5g and has wingspan of about 13cm while the biggest megachiroptera (*e.g.* indian flying fox) weighs over 2kg and has 1.7m wingspan. Figure 3.2 shows selected species under the Suborder Microchiptera.

Bats habitually live in a large colony approximately up to 700 or 1000 individuals under the sharing roost. Normally, a colony of bats will occupy a vertically roosting crevice (such as in caves or roof of abandoned buildings) that ends in a horizontal ceiling of the size of 0.75 to 1 inch wide and 16 to 24 inches deep. Figure 3.3 shows an example of a colony of bats roosting.

The bats usually fly out at dusk when the surrounding started to turn dark and they rely on spatial memory such that bats exiting from the roost in a colony concurrently. Most of the bats are insectivorous (eat insects), but there are also species of bats that have diversified their meals habit to fruits, nectar, small vertebrates (including fish) and also blood (vampire bats).

There are two categories of acoustic communication (or calls) used by a colony of bats. These are social calls for socialising or communicating between bats and echolocation calls for foraging and orientation purposes. A colony of bats are able to construct good communication and share information about roost site or forage area among one another.



Common and scientific names of bats

- **Order Chiroptera**
  - **Suborder Megachiroptera**
    - Pteropodidae
    - Egyptian fruit bat (*Rousettus aegyptiacus*)
    - Indian flying fox (*Pteropus giganteus*)
  - **Suborder Microchiroptera**
    - **Superfamily Emballonuroidea**
      - Craseonycteridae
        - Bumblebee bat (*Craseonycteris thonglongyai*)
      - Emballonuridae
        - Blackhawk bat (*Saccolaimus peli*)
    - **Superfamily Rhinolophoidea**
      - Hipposideridae
        - Commerson's leaf-nosed bat (*Hipposideros commersoni*)
      - Megadermatidae
        - Ghost bat (*Macroderma gigas*)
      - Rhinolophidae
        - Horseshoe bats (*Rhinolophus* spp.)
    - **Superfamily Phyllostomoidea**
      - Mormoopidae
        - Parnell's moustached bat (*Pteronotus parnellii*)
      - Phyllostomidae
        - Bennett's spear-nosed bat (*Mimon bennettii*)
        - Big-eyed bat (*Chiroderma villosum*)
        - Greater spear-nosed bat (*Phyllostomus hastatus*)
        - Linnaeus' false vampire bat (*Vampyrum spectrum*)
        - Little long-eared bat (*Micronycteris megalotis*)
        - Long-tongued bat (*Glossophaga soricina*)
        - Short-tailed fruit bat (*Carollia perspicillata*)
    - **Superfamily Vespertilionoidea**
      - Molossidae
        - European free-tailed bat (*Tadarida teniotis*)
        - Underwood's mastiff bat (*Eumops underwoodi*)
      - Vespertilionidae
        - European pipistrelle (*Pipistrellus pipistrellus*)
        - Giant house bat (*Scotophilus nigrita*)
        - Mexican long-eared bat (*Plecotus mexicanus*)
        - Naked bat (*Cheiromeles torquatus*)
        - Western pipistrelle (*Pipistrellus hesperus*)

Figure 3.1: Common and scientific names of bats

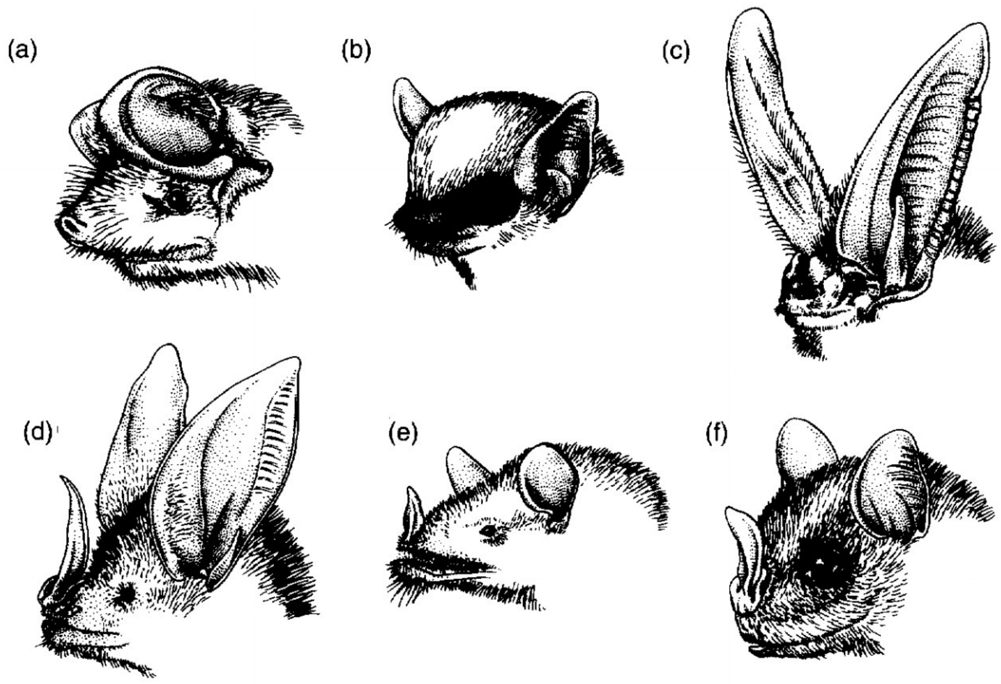


Figure 3.2: Portraits of selected Suborder Microchiptera. (a) Underwood's mastiff bat. (b) Western pipistrelle. (c) Mexican long-eared bat. (d) Bennett's spear-nosed bat. (e) Long-tongued bat. (f) Big-eyed bat

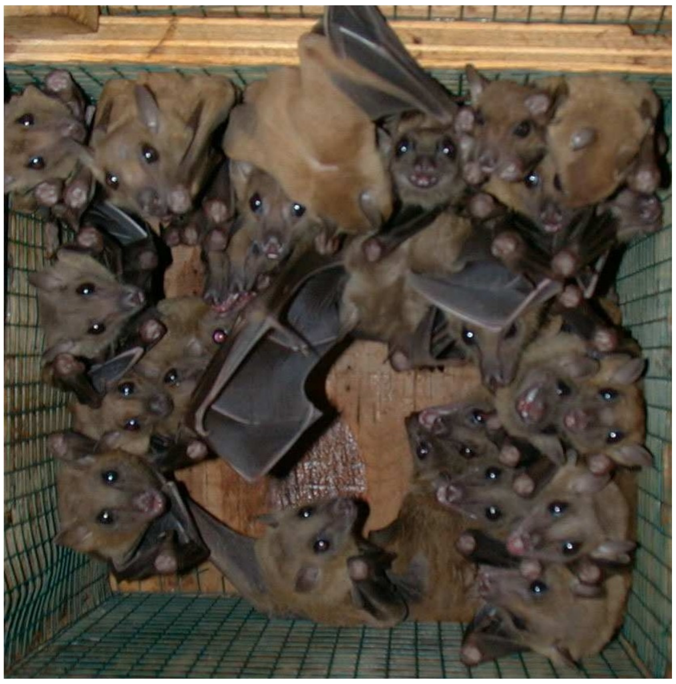


Figure 3.3: A colony of bats roosting where the picture is taken from below with the bats hanging upside down

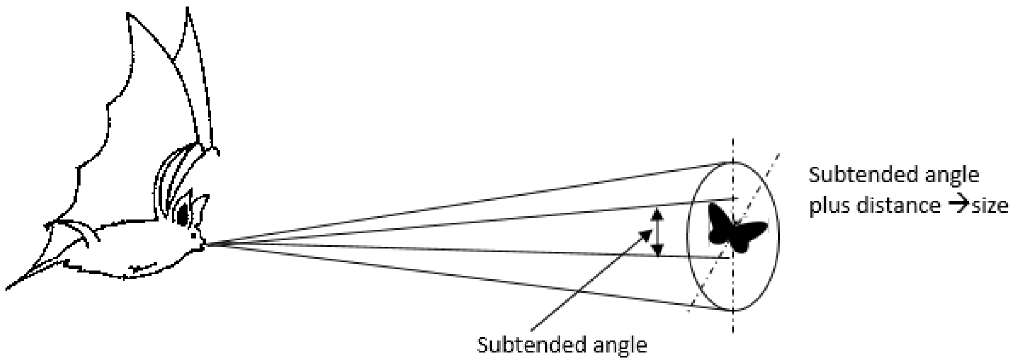


Figure 3.4: Sonar signal of a bat

There are four basic information transfer mechanisms in a colony of bats:

1. Intentional signalling: in the form of mating calls, territorial calls, alarm calls or food calls (advertisement of food and also to attract bats into foraging groups as they leave their cave roosts).
2. Local enhancement: involves unintentionally directing another bat to a specific part of the habitat.
3. Social facilitation: an increase in individual foraging success brought about by group foraging behaviour.
4. Imitative learning: bats can learn foraging techniques from other bats.

3.2 Real echolocation behaviour of bats

One of the great animal life ingenuities studied by many zoologists is the echolocation (or biological sonar) of bats. There are a few other animal groups that also possess echolocation capabilities such as birds (South American oilbird and south-east Asian swiftlets), whales, dolphins and small insectivores (shrews and tenrecs) but this is quite rare. The study of this behaviour of bats started by Lazzaro Spallanzani in 1794. Then the term 'echolocation' was introduced by Donald Griffin in 1944 to mark the ability of bat to produce sound with echoes beyond the frequency range of human hearing and use it for general orientation in the dark and to find prey at night.

With echolocation, a bat emits ultrasonic pulses either in frequency modulated (FM) or constant frequency (CF) and sometimes a combination of both. The tonal signals produced in the larynx (some bats use tongue) and emits in short bursts through mouth or nostrils as shown in Figure 3.4. The sound reflects back as echoes bump into objects in the bat's path. The reflected sounds were in compression condition or Doppler-shifted that made the echo received to be in higher frequency than the sound previously produced. The bat can identify the object and its distance by measuring the time of reflection of the modulated echoes.

The echolocation process of bats that leads to the catching of prey involves three phases; search phase, approach phase and terminal phase. When the bats start to hunt for prey in the search phase, they emit low rate pulse at around a frequency of 10Hz. During the approach phase, where the bats detect and get closer to the prey, the pulses have to get shorter to prevent overlap. The shorter pulses are cause by the decreasing of time between the pulse and echo. At this moment too, pulse emission rate gets gradually increased up to 200 per second as the bats keep updating the location of the prey . The pulse emission rate upsurges because the bats need to emit more signals to trail the prey precisely as the angular position of the prey changes more swiftly due to the closer distance between the bats and the object. In the last phase (terminal phase), the frequency of emitted pulses rises more than 200Hz and the pulse emission rate becomes faster at only fraction of milliseconds long just before the prey is captured.

In reality, a colony of bats has two exclusive approaches to avoid from colliding with one another during echolocation. The pulse characteristics emitted by each bat differ from one to another in terms of frequency range or time course of sweep or in sound type. Second, every bat marks its emitting pulse with a unique time structure so that they only retrieve echoes caused by their pulses. For generations, the echolocation was the great ability of bats that guided them to detect, localise and capture the prey simultaneously even the tiny insects at about the same distance in complex surroundings within splits seconds.

A colony of bats also embeds the concept of reciprocal altruism of food sharing during the echolocation process. This social behaviour of bats' group is based on animals returning favours for their mutual benefits. The example of this behaviour mostly applies to vampire bats species such as the regurgitation of blood-meals by successful bats to be fed to their futile member of the colony as a response to the finely balanced energy budget of each member of the colony.The reciprocal altruism behaviour grow in the survivor such that the fitness of the recipient is elevated relatively to a non-recipient. The reciprocal altruism also takes place during communal nursing or coalition formation in primates and support behaviour in cetaceans.

3.3 Bat algorithm

The bat algorithm (BA) by Yang in 2010 has been researched based on echolocation behaviour of bat species to find their prey. Bat form three-dimensional of surrounding by integrating the production of the sound pulse and echo recognition time difference, the variant intensity of the sound pulse and the time delay between ears of the bat. In a such way, the bat can identify the type, moving speed, distance and orientation of the prey.

To simplify, the algorithm was based on the ideal rules which are:

1. All bats use echolocation to detect distance and differentiate between food, prey and obstacles.
2. Bats fly randomly with velocity v_i at position x_i by fixed frequency f_{min} with varying wavelength λ and loudness A_0 to search for prey.
3. Bats can spontaneously adjust the wavelength or frequency and the rate of sound pulse emission $r \in [0,1]$ depending on the proximity of their target.
4. Loudness of emitted sound pulse assumed varies from a large positive A_0 to a minimum constant value A_{min}.
5. No ray tracing is used in estimating the time delay and the three dimensional topography.
6. Wavelength (λ) and frequency (f) of emitted sound pulse are related due to the fact λf is constant, so a range of $[f_{min}, f_{max}]$ is corresponds to a range of $[\lambda_{min}, \lambda_{max}]$.
7. Wavelength (or frequency) range can be adjusted and the largest wavelength (or frequency) should be selected to suit the size of the domain of the considered problem, and then toning down to smaller ranges.
8. Assume $f \in [0,1]$ even though higher frequencies have short wavelengths and travel a shorter distance.
9. The rate of sound pulse emission was in the range $[0,1]$ where 0 means no pulses at all and 1 means the maximum rate of pulse emission.

Algorithm 1 Bat algorithm

```
1: Objective function $F(x)$, $x = (x_1, \dots, x_d)^T$
2: Initialise: *bat population* $x_i$ and $v_i$ where $(i = 1, 2, \dots, d)$; *pulse frequency* $f_i$ at $x_i$; *pulse rate* $r_i$ and *loudness* $A_i$
3: **while** $t \leq$ *Maximum number of iterations* **do**
4:     Generate new solutions by adjusting frequency, and updating velocities and locations/solutions as **Equation 3.1**
5:     **if** $rand \geq r_i$ **then**
6:         Select a solution among the best solutions
7:         Generate a local solution around the selected best solution
8:     **end if**
9:     Generate a new solution by flying randomly
10:    **if** $rand \leq A_i$ & $F(x_i) \leq F(x_{i^*})$ **then**
11:        Accept new solutions
12:        Increase $r_i$ and reduce $A_i$
13:    **end if**
14:    Rank the bats and find the current best $x^*$
15: **end while**
16: Postprocess results and visualization
```

The bat algorithm is pictured in pseudo code as in Algorithm 1. In this algorithm, the velocity v_i and position x_i of bats' movement in a d-dimensional search space are updated as:

$$
\begin{aligned}
& f_i = f_{min} + (f_{max} - f_{min})\beta \\
& v_i^t = v_i^{t-1} + (x_i^t - x_*) f_i \\
& x_i^t = x_i^{t-1} + v_i^t \\
& \text{where} \\
& x_i^t \;\text{ is new solution of position at time step } t \\
& v_i^t \;\text{ is new solution of velocity at time step } t \\
& \beta \in [0,1] \;\text{ is random value} \\
& x_* \;\text{ is the recent global best solution which is derived} \\
& \quad \text{after examining every solutions among } n \text{ bats}
\end{aligned}
\tag{3.1}
$$

To update the velocity of the new solution, either f_i or λ_i could be used while fixing the other factor as velocity increment as a product of $\lambda_i f_i$. The value of f_i (or λ_i) is important to control the pace and range of the movement of the bats. On the other hand, values of f_{max} and f_{mim} have been fixed as $f_{min} = 0$ and $f_{max} = 100$ where each bat has its random frequency that is allocated uniformly around the fixed values above. However, the values have relied on the problem domain size.

A new position for every bat is produced using random walk after a solution is chosen among the current best positions as:

$$
\begin{aligned}
& x_{new} = x_{old} + \varepsilon A^t \\
& \text{where} \\
& \varepsilon \in [-1,1] \;\text{ is a random number} \\
& A^t = \langle A_i^t \rangle \;\text{ is the average loudness of all the bats at this time step}
\end{aligned}
\tag{3.2}
$$

Usually, when a bat approaches its prey, the loudness (A_i) will decrease but the rate of pulse emission r_i increases. Initially, every bat owns dissimilar random loudness values and pulse emission rate. So, as iteration proceeds and the new solutions are better, these two parameters have to be update.

For example, with the algorithm using $A_0 = 1$ and assuming $A_{min} = 0$ a bat moves to the prey and momentarily stops producing any sound. In contrast, with the algorithm using $r_0 = 0$ and assuming $r_{max} = 1$ a bat increases its pulse emission rate once approaching the prey. So the following equation is derived:

$$\begin{aligned} &A_i^{t+1} = \alpha A_i^t \\ &r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \\ &\text{where} \\ &\alpha = \gamma = 0.9 \end{aligned} \tag{3.3}$$

The BA method has been implemented on various test functions including Rosebrock's function, the egg crate function, De Jong's standard sphere function, Ackley's function and Michalewicz's test function. In all implementation, the numbers of bats (n) used were 25 to 50. The BA has been compared with standard GA and PSO algorithms in terms of the number of function evaluations for a fixed tolerance to show the better performance of BA. The fixed tolerance was set up at $\varepsilon \leq 10^{-5}$ and ran for 100 iterations. According to the results, the BA is more accurate and efficient compared to GA and PSO algorithms.

3.4 Variants of bat algorithm

After it was established five years ago, BA aroused intense interest in the optimisation field. There are numerous research works that have utilised the original BA in various engineering optimisation problems. In fact, many researchers tried to improve the original version of BA or pair it with other techniques to make the algorithm better and effective for solving certain problems.

3.4.1 Improved version

There are some research works that have been through improving the performances and wide spreading scopes of the solution of the original version of BA after it was introduced. For instances, in 2011, Yang has tried to use BA for solving specific nonlinear problems. The proposed method achieved better optimum solutions when compared with other existing algorithms.

In the same year, Tsai *et al.* introduced evolved bat algorithm (EBA) which modified the original framework of BA. The authors reanalysed and redefined corresponding operation behaviour of whole bat species. The method improves the accuracy of finding the best solution and reduces computational time when solving a numerical optimisation problem.

Meanwhile, Yang also extended his original technique to use in a multi objective optimisation problem in the same year. The multi objective bat algorithm (MOBA) works when it is applied to solve multi objective of welded beam design. Later, in 2012, Gandomi *et al.* solved constrained optimisation problems using BA. When compared with the various existing algorithms, the optimum solutions provided by BA are found to be better.

Lin *et al.* have incorporated chaotic sequence and chaotic Levy flight schemes to generate new solutions of original BA efficiently also in 2012. This work aims to enrich the searching behaviour and balance finely between intensification and diversification.The researchers demonstrated that the approach was reliable after adapting it for joint estimation of parameter vector in a reconstruction of a dynamical-biological system.

Furthermore, Lin *et al.* also tried to include Levy flight and chaotic dynamics mechanism for solving parameter estimation problem in nonlinear dynamic models of biological systems. Simulation results of the system have shown superiority of the approach.

3.4.2 Hybrid version

To improve the ability of any algorithm for solving many research areas, hybrid mechanism between algorithms become popular lately. BA is also not excluded from this cutting-edge phenomenon. Komarasamy and Wahi in 2012 for example, have combined K-means algorithm with BA for boosting efficiency in clustering large data sets of data analysis methods. The KMBA algorithm does not only achieves higher efficiency in clustering analysis but also contributes to the minimum computational resources and the time used for it. Besides that, Khan *et al.* have used the merits of BA to compensate the drawbacks of a fuzzy c-means (FCM) algorithm that are sensitive to starting configuration and lock into local optimum only.

The BA is also hybridised with differential evolution (DE) schemes by Fister *et al.* in 2013. This process significantly increases the ability of original BA as well as reveals encouraging results when testing on standard benchmark test functions. Meanwhile, Xie *et al.* use the same technique to establish the hybrid BA with mutation strategy (or called differential operator) which is a part of DE algorithm and Levy flight trajectory. This combination aims to increase the convergence rate and accuracy and the results displayed that the hybrid approach has better-quality of estimation capabilities, especially for advanced dimensional space.

Then, in the same year, Wang and Guo have established a robust hybrid metaheuristic optimisation approach by combining the step in harmony search (HS) algorithm into BA. To update the BA process, the researchers added one of HS attribute as an operator. By using pitch adjustment attribute, the hybrid technique showed very promising results of speeding up convergence rate to solve global numerical optimisation problems.

3.4.3 Direct application

Nowadays, BA has become the centre of attraction among the researchers' community to solve various engineering problems. Khan *et al.* in 2011 also used this popular swarm intelligence algorithm in their research. The authors have used BA with fuzzy modification to fast screening of company workplace with high ergonomic risk in short computational time. Another ergonomic research that adopted BA in the study is done by Akhtar *et al.*. In this work, each bat denotes a possible solution of a skeletal configuration of a human body to approximate the overall human body posture.

In the mechanical engineering field, BA is also utilised. For example, an industrial gas turbine has been modelled by Lemma and Hashim using BA method. The BA-based model created could be used to optimise and monitor the performance of thermal systems. Recently, Ramesh *et al.* estimated emissions produced by fossil-fuelled power plant also by using BA.

Other applications that embedded BA have included manufacturing areas such as warehouse data and record deduplication, multistage hybrid flow shop (HFS) scheduling problems and multi-stage multi-machine multi-product scheduling problems. In electrical and electronics sector, a brushless DC (BLDC) motor wheel optimisation and optimal capacitor placement (OCP) problems are also solved by BA approach.

Further research that is linked with the usage of BA consist of detection of phishing websites, training neural network of eLearning, classification of microarray data sets, feature selection technique, path planning for uninhabited combat air vehicle, shape or topology optimisation and image matching problem.

3.5 Bats sonar algorithm

The bats sonar algorithm (BSA) by Tawfeeq in 2012 is explored based on echolocation process of a colony of bats to find food or prey. During echolocation, bats can figure out the size, distance, velocity, azimuth and elevation of the target by using the sonar. The BSA models the principles of bat sonar used in echolocation to search the optimum solution for a specific problem. Each point (prey location detected) in the search space (specific confined area) represents one possible solution. A bat is labelled as one sonar unit.

The BSA is starts by setting the *solution range* or the minimum and maximum values of the search space. Then, the *beam length* (L) is initialised as:

$$L \leq Rand \times \frac{\text{Solution range}}{2} \tag{3.4}$$

At every iteration, Tawfeeq has selected random *starting angle* (θ_m) as well as used one of two *angle between beams*; either $Fixed_\theta$ which randomly select a small fixed value θ between any two successive beams or $Rand_\theta$ which randomly select a different angle θ_i between any two successive beams.

The sonar unit will transmit a number of sonar signals or *number of beams* (N) with L length from the designated starting point (pos_s) to several different directions. The pos_s also evaluates the value of *starting point fitness function* (F_s). Every beam's *end point position* (pos_i) is calculated as:

$$pos_i = pos_s + L\cos(\theta_m + (i-1))\theta \tag{3.5}$$

Then, the pos_i is evaluated for the value of *end point fitness function* (F_i). The values of F_i and F_s are compared with each other to determine the optimum one. If the optimum value belongs to one of the F_i, the sonar unit (the bat) will fly to its pos_i and set the point as a new pos_s. Then, the new number of N beams will be transmitted from this point to search for better optimum solution. Otherwise, the bat will stay at the original pos_s and retransmit the N beams to different direction. The process keeps on repeating and stops once the algorithm arrives at the maximum iteration (or finds the best fitness function). Algorithm 2 pictures the pseudo code of the bats sonar algorithm.

The algorithm is a parallel search type where several solutions are checked simultaneously. Over iterations, only the best fitness of each bat will survive and the best fitness among the best bats' fitness will become the global best fitness. Using this way, the proposed algorithm will converge to the optimum best fitness faster.

This algorithm started with the single sonar unit (SSU). Then, the investigation of the proposed algorithm was expanded to other two efficient search approaches. If only SSU approach was being used, the result is not guaranteed to obtain the global best fitness even it converges toward the minimum or maximum fitness especially in complex problems with wide state space. The two approaches mentioned were multi sonar search unit (MSU) and single sonar unit with a momentum (SSM).

Algorithm 2 Bats sonar algorithm

```
1: Objective function $F(x)$, $x = (x_1, \dots, x_d)^T$
2: Initialise *Solution range*, $L$ (**Equation 3.4**), $N$, random $pos_s$ and *angle between beams*
3: Evaluate $F_s$ for $pos_s$
4: while $t \leq$ Maximum number of iterations do
5:     Select random $\theta_m$
6:     Transmit $N$ beams from $pos_s$ with $\theta_m$ and *angle between beams*
7:     Determine the coordinates of the every beams' end point ($pos_i$) for each transmitted beam (**Equation 3.5**)
8:     Evaluate the $F_i$ for each $pos_i$
9:     if $F_i \leq F_s$ then
10:        Substitute the coordinates of $pos_s$ with the coordinates of $pos_i$
11:        Replace $F_s$ with the optimum $F_i$
12:    end if
13: end while
14: Declare the best $F_i$ as optimum fitness evaluated and its $pos_i$ as optimum value(s)
```

In multi sonar unit (MSU), a colony of bats will search for the optimum solution(s) at the same time where each bat (sonar unit) will be assigned with different starting point in the same search space. Meanwhile, a single sonar unit with a momentum (SSM) introduced a *momentum term* (μ) attached to the length of the transmitted beams so that new beam length becomes as:

$$L_{new} = L_{old}(1 \pm \mu)$$

where (3.6)

$$0 < \mu < 1$$

Nonetheless, both approaches still use SSU algorithm as the algorithm framework.

To demonstrate the performance of the algorithm, the BSA were tested and evaluated on different types of fitness functions that are:

1. A single variable-third order polynomial for maximum value.
2. A single variable-fifth order polynomial for maximum value.
3. A polynomial with two variables for maximum value.
4. A exponential with two variables for maximum value.
5. A trigonometric or a periodic function (repeated function values in regular intervals or periods) for maximum value.

The initial parameters set to be the same for all tests included $N = 5$, $Fixed_\theta = \pi/12$ and 100 maximum iterations.

The performances of BSA were measured by the degree on how much the obtained solution meets the goal where the goal is assumed to be equal or approximately equal to the optimum solution. Comparison of the algorithm with a genetic algorithm on the same fitness functions has been made. The comparison involves the value of obtained fitness functions and the execution time required to attain each function. The results concluded the bats sonar algorithm performed reasonable efficiency to achieve all the optimum values.

As a matter of fact, the algorithm is only tested on single objective optimisation problems. Till today, no extended version of the algorithm, neither the modification to the original algorithm, hybridisation with another technique nor application to any optimisation area has been reported.

3.6 Problems associated with bats sonar algorithm

There are some drawbacks associated with the BSA. There is no communication between bats in a colony to exchange information on current location or the best locations of individual bats during echolocation process. This makes the algorithm a parallel search technique. The number of bats used in the algorithm is too small and not portraying the normal population size of a colony of bats (normally in the order of hundreds) when searching for prey. The small population does not make the exploration and exploitation for the best fitness value optimum in the search space.

Furthermore, it is highly possible that the *N* beams will be transmitted in the same direction and location. This problem happens because the main transmit angle is fixed as well as roughly set up of random values of the angle between beams. These drawbacks will lead to premature convergence as the algorithm will diverge from the global best position but converge to local best location. Thus, the algorithm does not perform well to achieve the best accuracy while maintaining good precision and fast convergence to the optimum solution.

BSA also fail to capitalise on several good characteristics in the real behaviour of bats echolocation into the algorithm. This failure makes BSA unable to operate like the real process of echolocation of a colony of bats. BSA is not considered the issues such as there are three phases lead to catching the prey, mechanism to avoid collision between bats as well as the reciprocal altruism model of food sharing between a colony of bats.

3.7 Bats sonar algorithm as basis for a set of new bats echolocation inspired algorithms

The results from the literature review have shown that:

1. The BSA is easy to design and implement.
2. The BSA has a good combination of a set of rules and randomness as required by most evolutionary algorithms.
3. The BSA does not fully consider the real echolocation behaviour of a colony of bats.
4. There is no modified or a new version of BSA since it is still relatively newly explored swarm intelligence algorithm.

So, this book will investigate a set of new bats echolocation inspired algorithms based on the BSA. The new algorithms will refine and modify the BSA with new elements and as well as fully adopt the real echolocation behaviour of a colony of bats.

Then, the new set of algorithms that will be investigated is inspired to be the most promising in the swarm intelligence algorithms that can be applied for solving a wide range of single objective optimisation problems, constrained optimisation problems and multi objective optimisation problems.

Chapter 4

Development of adaptive bats sonar algorithm

4.1 Adaptive bats sonar algorithm

In bats sonar algorithm (BSA), some drawbacks have been detected. The BSA fail to imitate the real behaviour of a colony of bats during echolocation process to the maximum. These includes there is no proper communication between bats in a colony during echolocation process while the number of bats used is too small does not make the searching process efficiently. Besides, exists the possibility of redundancy location and direction of transmitted beam along the iteration.

An ABSA is proposed as an improved version of original BSA. ABSA try to fix the drawback of the BSA with the aim of improving accuracy, precision and convergence rate of the BSA. ABSA altering and incorporating new characteristics into the BSA. This includes modification of the number of bats, number of beams and their lengths, starting angle and introduction of new techniques comprising beam number increment, four level of best solution and reciprocal altruism behaviour of real bats. The purpose of ABSA is to solve single objective optimisation problems.

Overall, the ABSA has more steps than the original BSA introduced. However, the *number of iterations* (*MaxIter*) or generations used in ABSA is kept at 100, same number used in the original algorithm. One hundred generations are favourably enough for the bats to explore fully the d numbers of search space *dimension* (*Dim*) for the best prey or *global best fitness*, (F_{GB}). The chosen value is in line with maximum *MaxIter* which was used in the PSO algorithm when the algorithm was first introduced by Kennedy and Eberhart in 1995.

Inspired by a description of the number of bats in a colony by biologists, the *number of bats* (*Bats*) or population in ABSA was selected in the range 700-1000 bats. The new population was higher by only three bats than that was used in the BSA. By having a larger number of bats, a discovery of the F_{GB} value becomes more resourceful such that there will be a pool of solutions (prey) that can be evaluated to obtain the best ones.

In the original BSA, the *beam length* (L) is initialised as a random value but not more than half of the *solution range* (SS_{size}). The solution range is the value between the *upper search space* (SS_{Max}) limit and the *lower search space* (SS_{Min}) limit as:

$$SS_{size} = SS_{Max} - SS_{Min} \tag{4.1}$$

The value of L is constant throughout the iterations. This fixation pushes every bat to search in larger perimeter each time without the opportunity to diversify the search tactic during iterations and thus may miss the F_{GB} that may be near to them. To resolve such weaknesses, the ABSA sets the L in relation to SS_{size} as:

$$L \leq Rand \times (\frac{SS_{size}}{10\% \times Bats}) \tag{4.2}$$

The solution range is divided into micron scale, such as 10% of the overall population of bats in the search space. The percentage is marked as possible search space size of each bat to emit sound without colliding with one another. The value of L is different for every iteration. A *momentum term* (μ) is used in ABSA as:

$$\begin{aligned} & L_{new} = L_{old}(1 \pm \mu) \\ \text{where} & \\ & 0 < \mu < 1 \end{aligned} \tag{4.3}$$

The above has been introduced by Tawfeeq (2012) to control the risk of convergence to a local optimum.

In BSA, the *number of beams* (*NBeam*) emitted by each bat at each iteration has been fixed to five. This value is too small and obviously only a part of the bat's surrounding is covered by the pulses and thus the exploitation of *local best fitness* (F_{LB}) and exploration of F_{GB} do not occur. Such a small value also does not illustrate the real echolocation of bats. The pulse emission rate grows bit by bit up to 200 per second as the bat keeps updating the location of the object until it catches the prey. This phenomenon is incorporated into the ABSA approach as *beam number increment* (*BNI*).

The *BNI* is defined in terms of the *maximum number of beams* ($NBeam_{Max}$) and *minimum number of beams* ($NBeam_{Min}$) as:

$$\begin{aligned} & BNI = (\frac{NBeam_{Max} - NBeam_{Min}}{MaxIter}) \times iter \\ \text{where} & \\ & NBeam_{Max} = 200 \\ & NBeam_{Min} = 20 \end{aligned} \tag{4.4}$$

Thus, *NBeam* is defined as:

$$NBeam = NBeam_{Min} + BNI \tag{4.5}$$

The *BNI* method mimics the original pulse rate emitted by the bat as it increases gradually toward the end of the search. As a result, *BNI* will provide a balance between global exploration and local exploitation thus requiring less iteration on average to find a sufficiently optimum solution.

Each *NBeam* with L is emitted from the *starting position* (pos_{SP}) with specific angle location. In BSA, random *starting angle* (θ_m) at every iteration has been selected, see Figure 4.1. For the angle between beams, the algorithm's initiator uses one of the following:

1. $Fixed_\theta$: randomly select a small fixed value θ between any two successive beams.
2. $Rand_\theta$: randomly select a different angle θ_i between any two successive beams.

In this manner, the beam transmitted will sweep at random angles at each iteration. However, the bats fail to verify that the sounds have spread to every corner of their surroundings and it is possible that the beam will be transmitted to the same point(s) at different iterations. As a consequence, the algorithm will get trapped at F_{LB} and will be unable to find the F_{GB}. To resolve this problem, ABSA limits the first beam to have θ_m not more than 45° from horizontal axis and the *angle between beams* (θ_i) is set as follows:

$$\theta_i = \frac{(2\pi - \theta_m)}{NBeam}$$

where (4.6)

$$\theta_m = rand \leq 0.7854$$

By setting θ_i as such, the beams will sweep at random 360° around the bats through iterations in such a way that the searching process will neither be too aggressive (overlay a circle) nor too slow (underlay a circle).

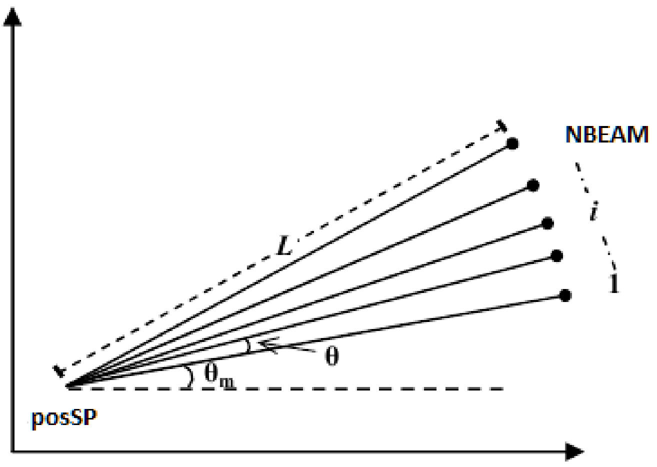


Figure 4.1: Single batch of beams transmitted by a bat

The *end point position* (pos_i) for each transmitted beam in ABSA is calculated the same way as in BSA as:

$$pos_i = pos_{SP} + L\cos[\theta_m + (i-1)\theta]$$

where (4.7)

$i = 1, \ldots, NBeam$; $NBeam$ is *number of beams*

The BSA declares a fitness at that position as the optimum fitness function once the algorithm has reached either the end of a fixed number of iterations or all solutions have converged to the same value. The one level declaration of best solution is consistent with the nature of the algorithm as a parallel search method where the algorithm checks for the solutions at once. Nonetheless, the level of best fitness solution found in the algorithm has been raised up to four stages in the ABSA. The duo are mentioned before; F_{LB} and F_{GB}, while another two levels are *starting position fitness* (F_{SP}) and *regional best fitness* (F_{RB}).

During the first iteration of ABSA, pos_{SP} of F_{SP} for each bat to transmit the *NBeam* is randomly selected within the designated search space. Next, the pos_i for each transmitted beam from pos_{SP} of each bat will be evaluate to produce *end point fitness* (F_i) where the best F_i is declare as F_{LB} and its position as *local best position* (pos_{LB}) of each bat. Later, the F_{SP} and F_{LB} of each bat is compared where the best will be F_{RB} and its position as *regional best position* (pos_{RB}). Finally, the best of the F_{RB} will be declared as F_{GB} and its position as *global best position* (pos_{GB}). There are three levels of best solution found by the algorithm in PSO. The levels are *personal best* (*pb*) which is the best solution for every particle, *local best* (*lb*) which is the neighbourhoods best solution and *global best* (*gb*) is the global best solution of among the *pb*.

These three levels are similar to F_{LB}, F_{RB} and F_{GB} of ABSA respectively.

In PSO, the *lb* improves the overall performance of algorithm where the individual *lb* influences the performance of immediate neighbours. Ultimately, the neighbourhoods preserve swarm diversity by hindering the flow of information through the network. This move prevents the particles from reaching the global best particle immediately or getting trap in a local optimum but allows them to explore larger search space. This beneficial element inspired the existence of F_{RB} which is functioning as neighbourhoods best solution-ABSA version. In addition, F_{RB} also forms the main link between F_{LB} and F_{GB} values. So F_{RB} acts as a leverage instrument to balance finely between exploration (diversification) and exploitation (intensification) processes of the algorithm and so to help the algorithm escape from premature convergence.

The initialisation of these levels will help the ABSA to refine the search for the best solution by a colony of bats in the search space in each step and leave out bad solutions immediately. As a result, the algorithm takes less time to converge to the optimum solution. In point of fact, many types of research show that communication between individuals within a group is important where the overall performance of the group is affected by the structure of the social network. Besides, the distribution of information via distant acquaintances is crucial, such that it possesses information that a colleague might not. In conjunction to that, the four levels of the best solution created in ABSA ideally match with the information transfer mechanisms practised by a colony of bats. These are intentional signalling match to F_{SP}, local enhancement match to F_{LB}, social facilitation match to F_{RB} and imitative learning match to F_{GB}.

The reciprocal altruism characteristic has further been incorporated into ABSA to strengthen the procedure of colony searching for the best solution. This reciprocal altruism behaviour widely runs through a colony of bats as reported by many researchers in bats ecology. By inserting this behaviour into the algorithm, a member of the colony will disseminate and share the location of the best fitness found so far to other bats. As a result, all bats will fly to the best prey ever found when the search process comes to an end. The adoption of this real prey hunting behaviour of the colony of bats into the algorithm is symbolised by two levels of arithmetic mean.

For every bat, the arithmetic mean evaluates the balancing point between pos_{SP}, pos_{LB} and pos_{RB} in current iteration (t) with pos_{GB} of the latest F_{GB} to be appoint as a new pos_{SP} for next iteration (t+1). The first level of arithmetic mean involves measuring of central tendency between pos_{SP}, pos_{LB} and pos_{RB} of each bat for current iteration only. Next, the second level of arithmetic mean finds the central tendency between the position value resulted from the first level of arithmetic mean and pos_{GB}. As a result, during new iteration, every bat will start to transmit a set of new beams from the pos_{SP} which has been specified after considering (or sharing) the balancing point of the positions of all four level of best fitness solutions; F_{SP}, F_{LB}, F_{RB} and F_{GB}. The two levels of arithmetic mean is expressed as follows:

$$pos_{SP}(t+1) = \left(\frac{pos_{SP}(t) + pos_{LB}(t) + pos_{RB}(t)}{3} + pos_{GB}\right) \Big/ 2 \tag{4.8}$$

Based on these modifications, the basic steps of the ABSA are represented as the pseudo code in Algorithm 3.

Algorithm 3 Adaptive bats sonar algorithm

```
1: Objective function $F(x), x = (x_1, \dots, x_d)^T$
2: Initialise: $Bats$, $MaxIter$, $Dim$, $SS_{Size}$, $NBeam_{MAX}$ and $NBeam_{MIN}$
3: for $n \leftarrow 1$ to $Bats$ do
4:     for $d \leftarrow 1$ to $Dim$ do
5:         Generate random $pos_{SP}$
6:         Evaluate $F_{SP}$ value for $F(pos_{SP})$
7:     end for
8: end for
9: Assign the most optimum value as $F_{GB}$ and its position as $pos_{GB}$
10: while $t \leq MaxIter$ do
11:     Define $NBeam$ to transmit by using $BNI$ (Equation 4.4 and Equation 4.5)
12:     Set $L$ and limit $\mu$ (Equation 4.2 and Equation 4.3)
13:     Generate random $\theta_m$ and $\theta$ (Equation 4.6)
14:     for $n \leftarrow 1$ to $Bats$ do
15:         Transmit $NBeam$ starting from $pos_{SP}$
16:         for $N \leftarrow 1$ to $NBeam$ do
17:             for $d \leftarrow 1$ to $Dim$ do
18:                 Determine $pos_i$ for each transmitted beam (Equation 4.7)
19:             end for
20:             Evaluate $F_i$ value for $F(pos_i)$
21:         end for
22:         Assign the optimum value of $F_i$ as $F_{LB}$ and its position as $pos_{LB}$
23:         if $F_{LB} \leq F_{SP}$ then
24:             Assign $F_{LB}$ as $F_{RB}$ and $pos_{LB}$ as $pos_{RB}$
25:         else
26:             Assign $F_{SP}$ as $F_{RB}$ and $pos_{SP}$ as $pos_{RB}$
27:         end if
28:     end for
29:     Select the optimum value among $F_{RB}$ as current $F_{GB}$ and its $pos_{RB}$ as current $pos_{GB}$
30:     if current $F_{GB} \leq$ previous $F_{GB}$ then
31:         Update current $F_{GB}$ as new $F_{GB}$ and current $pos_{GB}$ as new $pos_{GB}$
32:     else
33:         Retain previous $F_{GB}$ and $pos_{GB}$
34:     end if
35:     for $n \leftarrow 1$ to $Bats$ do
36:         Determine new $pos_{SP}$ using (Equation 4.8)
37:         Evaluate new $F_{SP}$ value for $F(pos_{SP})$
38:     end for
39: end while
40: Declare $F_{GB}$ as optimum fitness evaluated and $pos_{GB}$ as its optimum value(s)
```

4.2 Computer simulation and discussion

4.2.1 Effects of *number of bats* and *number of iterations* on performance of ABSA

Any swarm intelligence algorithm requires setting the values of several algorithm parameters correctly because these parameter values have a significant impact on the performance and efficiency of the algorithm. The size of population and number of iterations used are the main parameters in most of the swarm intelligence algorithms. In BSA and ABSA algorithms, the size of a population is referred to the *number of bats* (*Bats*). However, BSA applied three bats only while in ABSA the number of bats used are between 700 and 1000 bats, as motivated by the study reported by Rivers *et al.* and Voigt-Heucke *et al.*.

On the other hand, the *number of iterations* (*MaxIter*) used in both algorithms has been set to 100. This value is favourably enough for the bats to explore fully the search space for the best prey (best fitness value). The chosen value is twice the maximum of what *MaxIter* used in PSO when the algorithm was first introduced in 1995. The overall performance of ABSA is better than BSA not because of the large difference *Bats* used at various number of iterations only, but due to the improvement and modifications made to the original BSA. To demonstrate this, both BSA and ABSA are tested with two different benchmark functions as follows:

a. McCormick function

This function as in Figure 4.2a is unimodal test function and is defined as:

$$\begin{aligned} &F(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1 \\ &\text{where} \\ &x_1 \in [-1.5, 4.0] \\ &x_2 \in [-3.0, 4.0] \end{aligned} \tag{4.9}$$

The global minimum is $F(x^*) = -1.9132$ at $x^* = (-0.54719, -1.54719)$.

b. Rastrigin function

This function is a multimodal test function with several regularly distributed local minimum. This function as plot in Figure 4.2b is defined as:

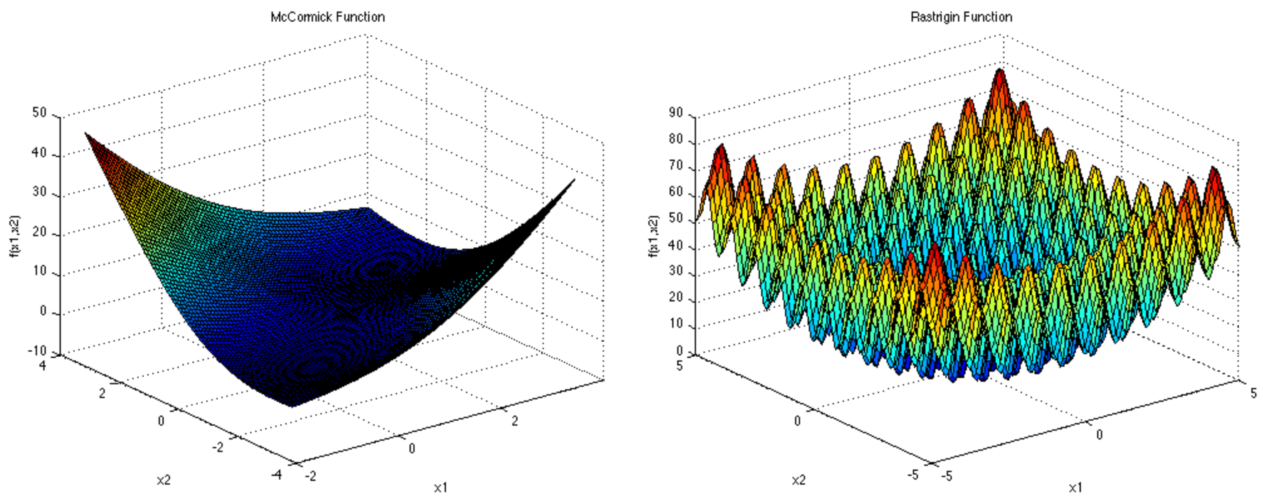
$$\begin{aligned} &F(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)] \\ &\text{where} \\ &x_i \in [-5.12, 5.12],\ i = 1, \ldots, d \end{aligned} \tag{4.10}$$

The global minimum at $F(x^*) = 0$ at $x^* = (0, \ldots, 0)$. The test of this function used $d = 3$.

In both cases, the number of *Bats* used were 3, 100 and 700 while the *MaxIter* is fixed to 25 and 100. So, number of function evaluations (NFEs) defined as:

$$NFE = Bats \times MaxIter \tag{4.11}$$

for each BSA and ABSA are 75, 300, 2500, 10000, 17500 and 70000.



(a) McCormick function (b) Rastrigin function

Figure 4.2: Functions used to evaluate the effects of *Bats* and *MaxIter* on the performances of BSA and ABSA

Table 4.1: Best global optimum value achieved by BSA and ABSA for McCormick function with different *Bats* over different *MaxIter*

Bats	*MaxIter*	Optimum value of $F(x)$	BSA	ABSA	*NFE*s
3	25	-1.9132	-1.8464	-1.9132	75
	100		-1.9130	-1.9127	300
100	25		-1.9130	-1.9132	2500
	100		-1.9123	-1.9132	10000
700	25		-1.9126	-1.9132	17500
	100		-1.9132	-1.9132	70000

Table 4.2: Best global optimum value achieved by BSA and ABSA for Rastrigin function with different *Bats* over different *MaxIter*

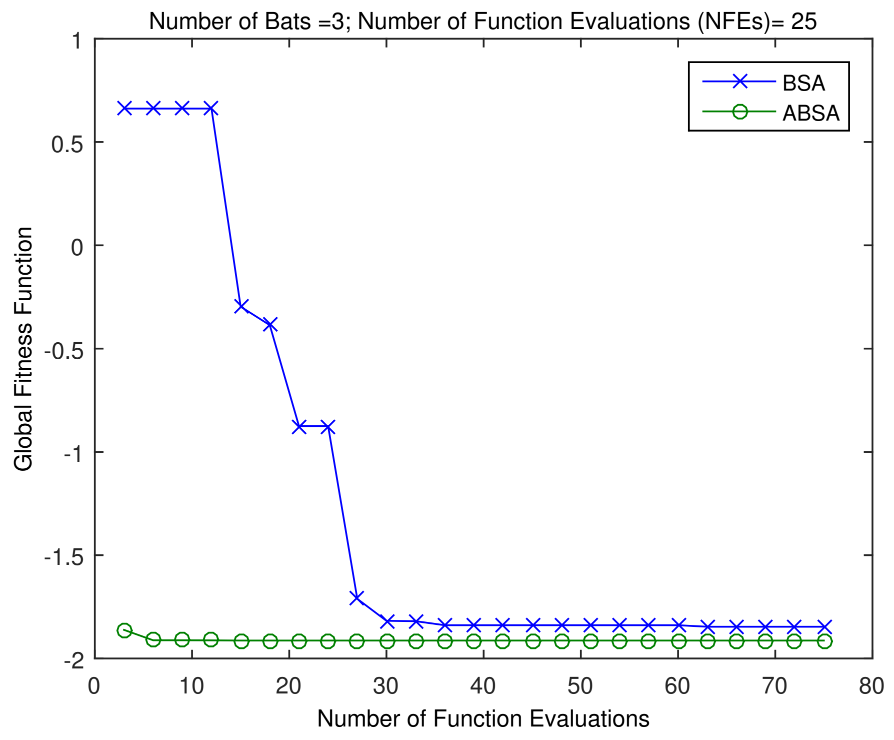
Bats	*MaxIter*	Optimum value of $F(x)$	BSA	ABSA	*NFE*s
3	25	0.0000	3.6481	0.7116	75
	100		1.2568	$1.2740e^{-1}$	300
100	25		0.9951	$3.8270e^{-6}$	2500
	100		$5.1865e^{-1}$	$5.8799e^{-7}$	10000
700	25		$2.1431e^{-1}$	$3.2585e^{-8}$	17500
	100		$7.0612e^{-2}$	$4.9231e^{-10}$	70000

Table 4.1 and Figure 4.3 depict the best results obtained by the BSA and ABSA in optimising the McCormick function. It is noted that the ABSA outperformed the original BSA at various *Bats* used with different *MaxIter* to accelerate the

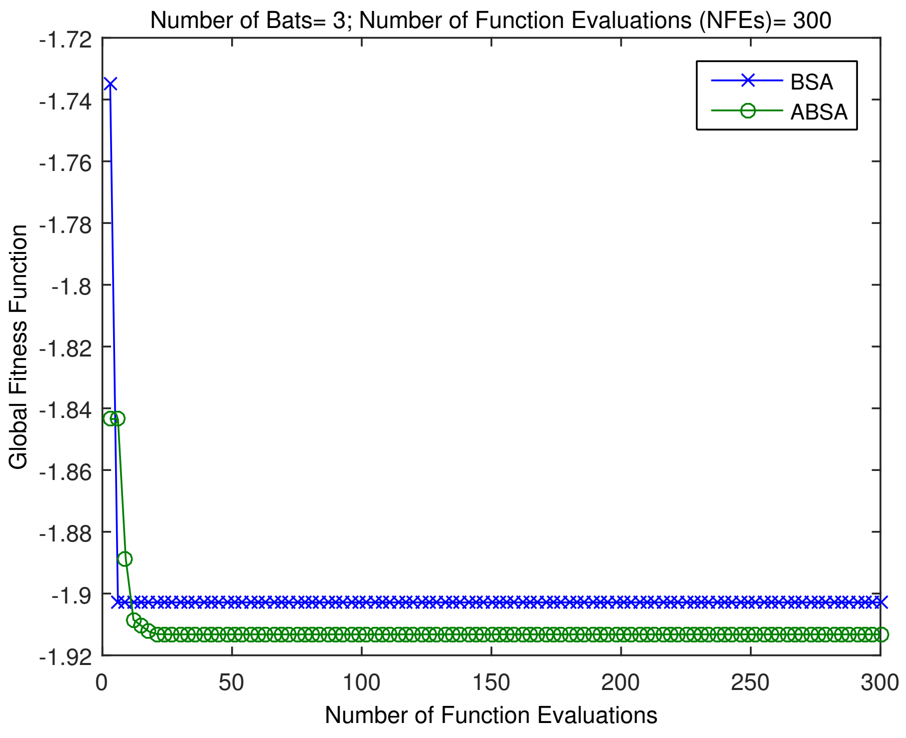
convergence rate to accurate known global optimum.

As evidenced in Table 4.2 and Figure 4.4, ABSA further showed promising results as compared to the original BSA method. The obtained results in optimising the Rastrigin function suggested that the ABSA succeeded to converge faster and near accurate to the best known global optimum at various numbers of bats used with different numbers of iterations as compared to original BSA.

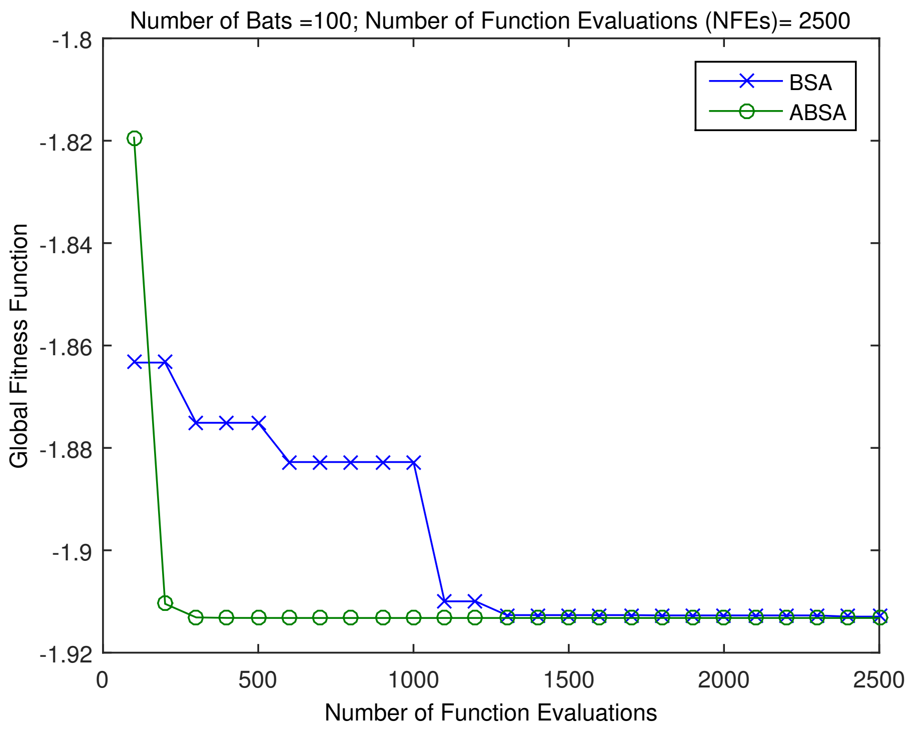
At this point, the preliminary conclusion drawn about the ABSA as compared to original BSA is that ABSA has successfully converged faster with better accuracy to the known global optimum when compared with BSA without it being affected by a large difference in the number of bats used at various numbers of iterations.



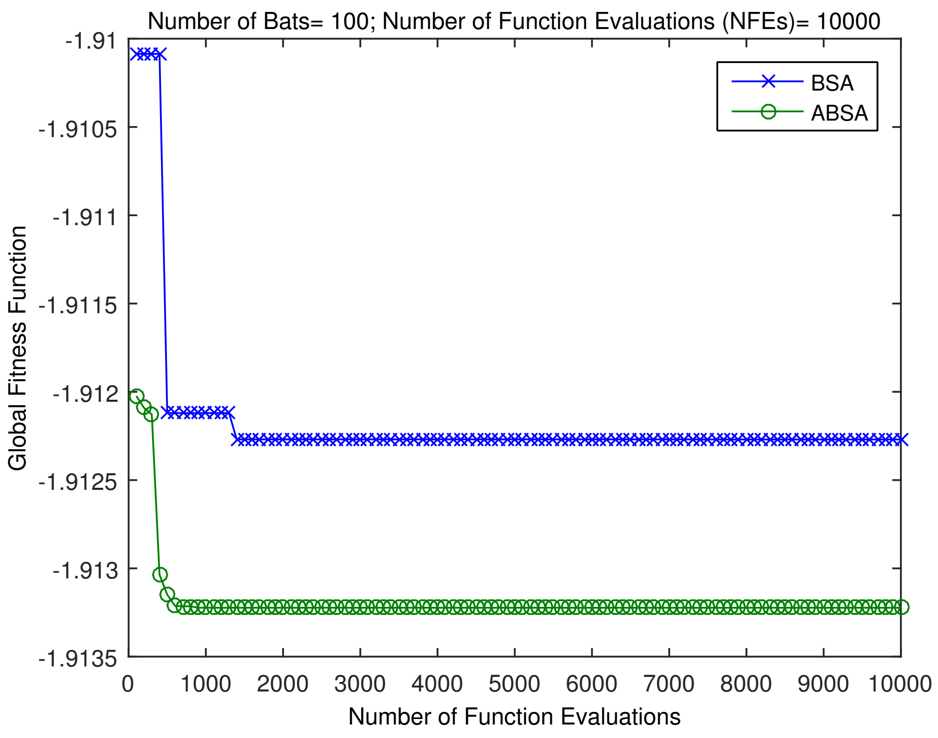
(a) 3 bats and 25 iterations



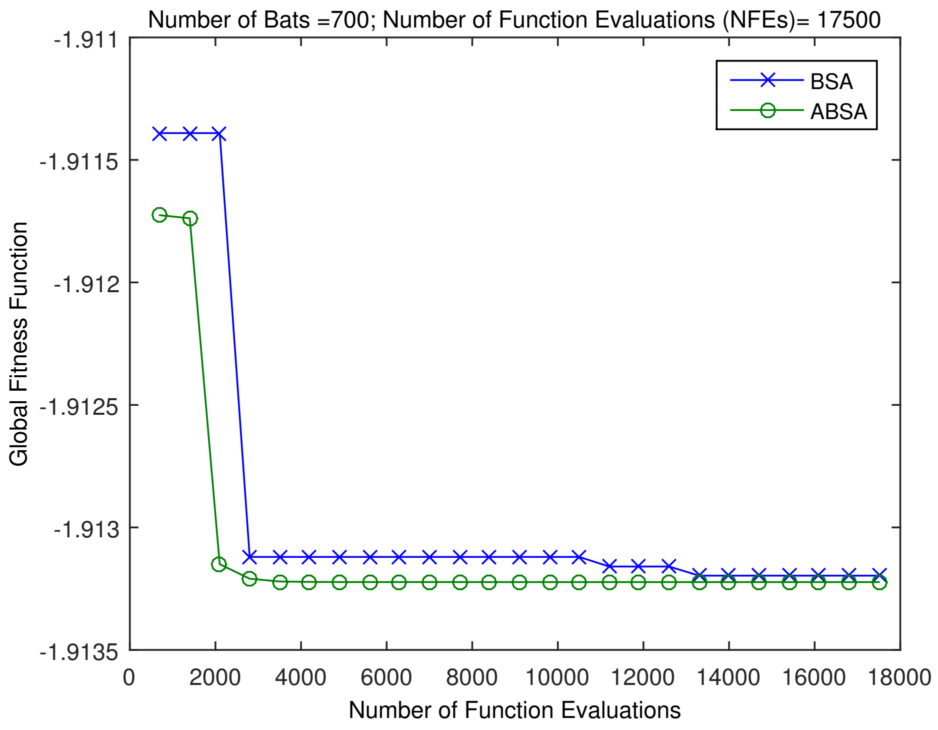
(b) 3 bats and 100 iterations



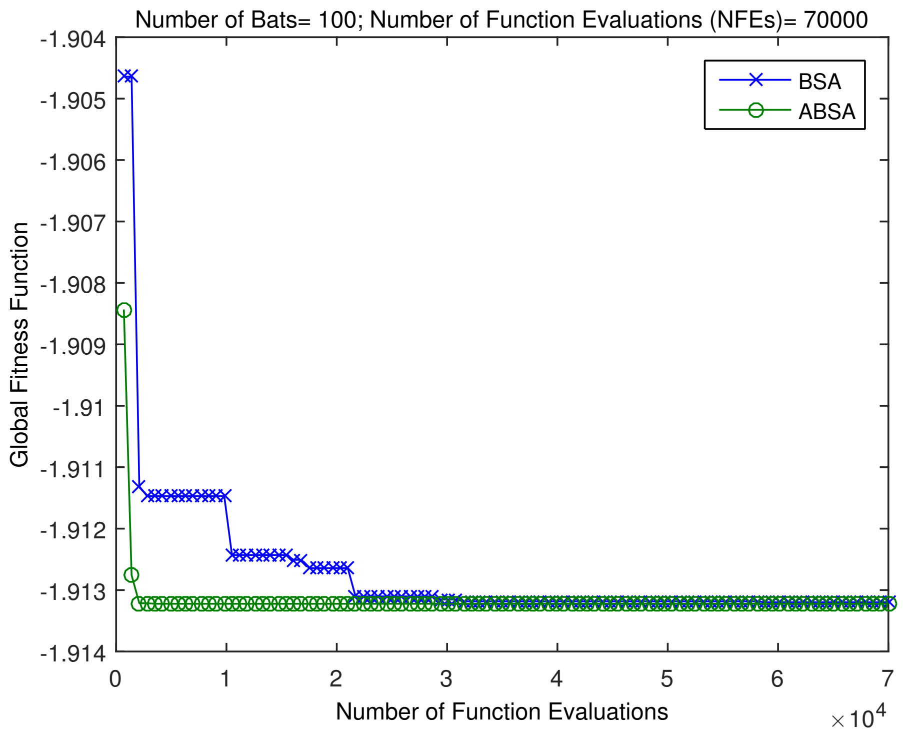
(c) 100 bats and 25 iterations



(d) 100 bats and 100 iterations

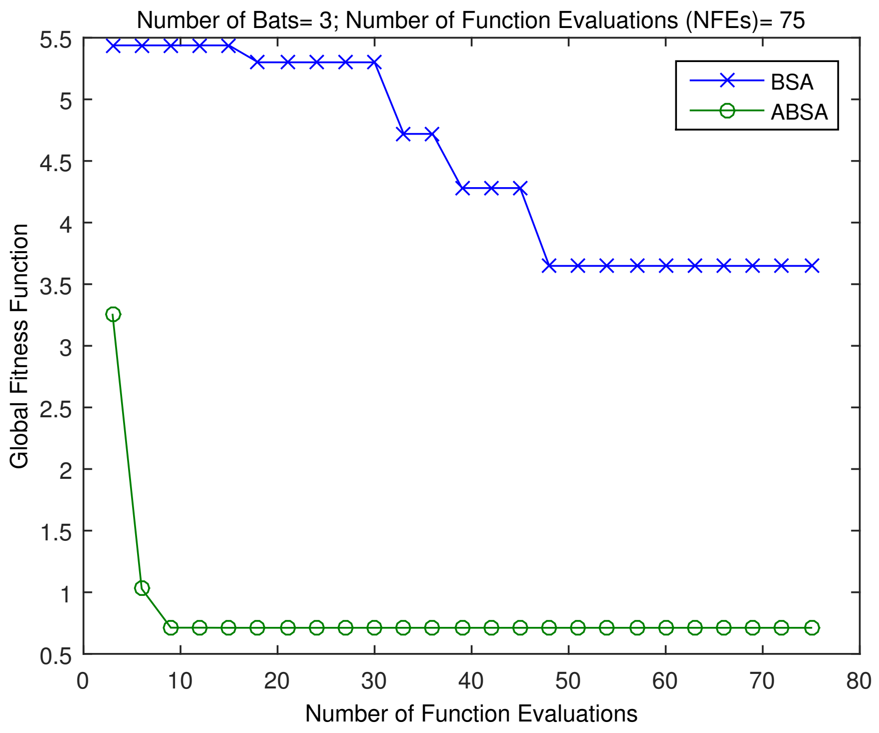


(e) 700 bats and 25 iterations

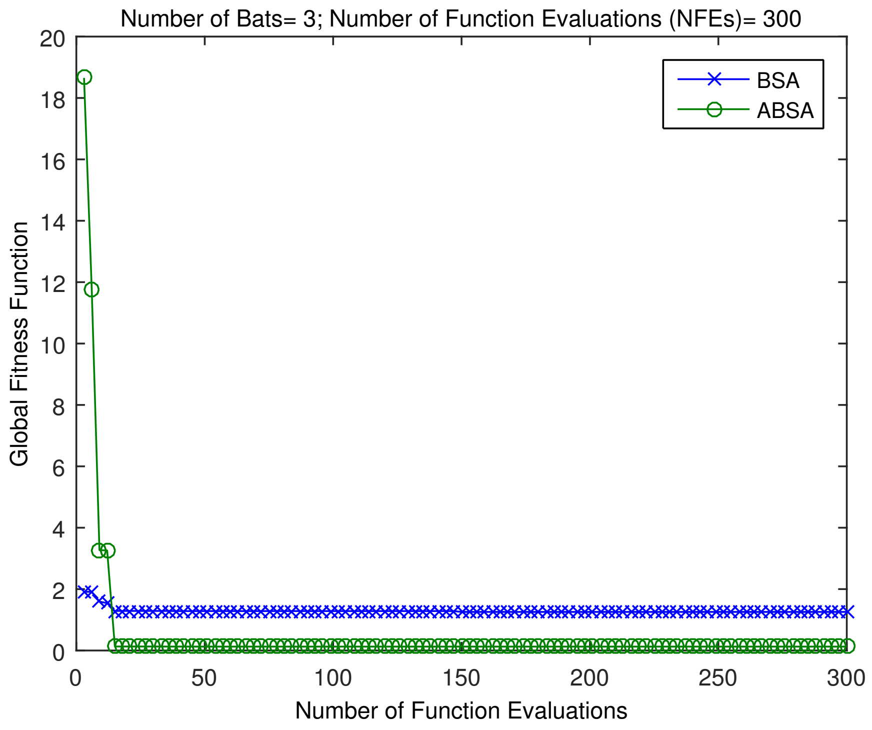


(f) 700 bats and 100 iterations

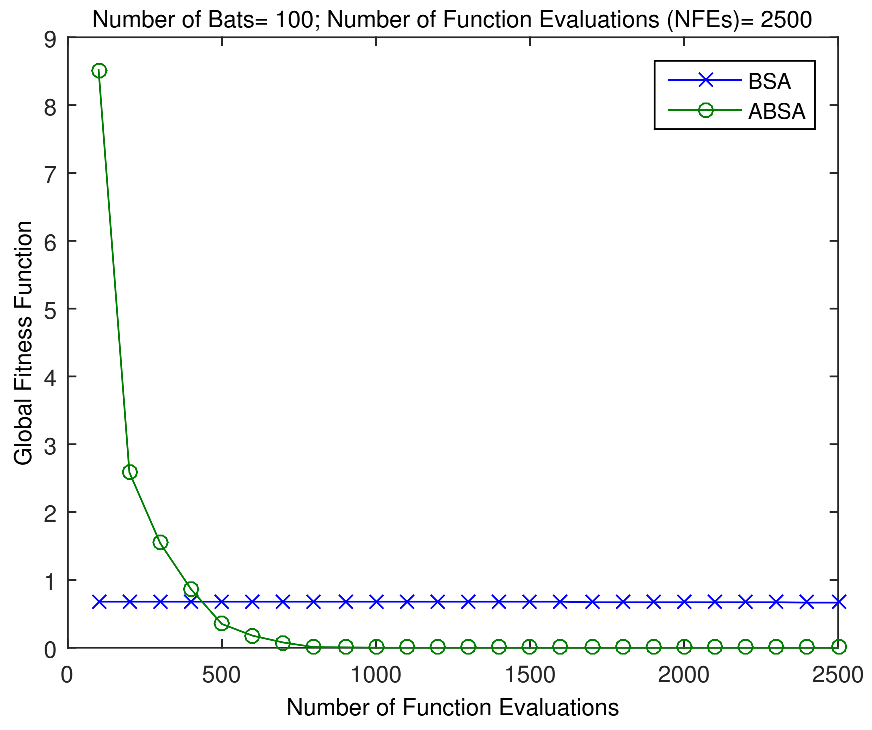
Figure 4.3: McCormick functions: comparison of performance of the original BSA and the ABSA



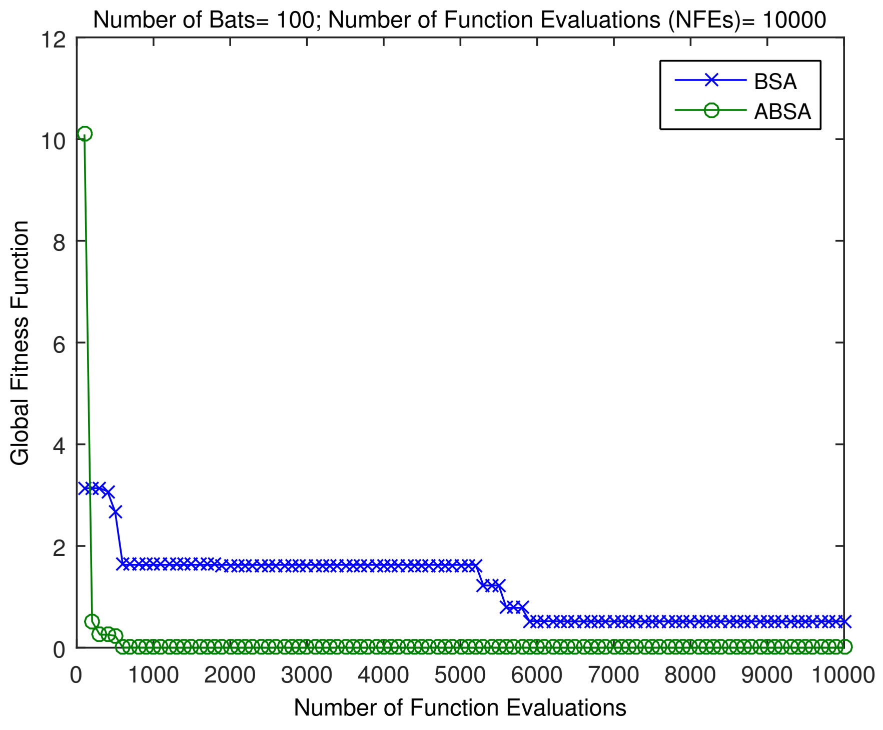
(a) 3 bats and 25 iterations



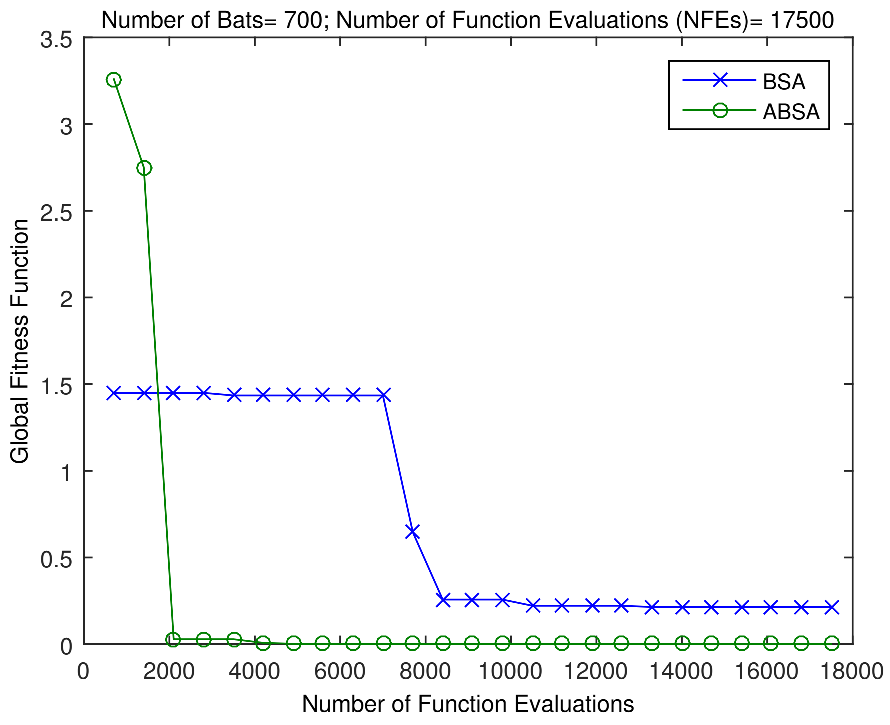
(b) 3 bats and 100 iterations



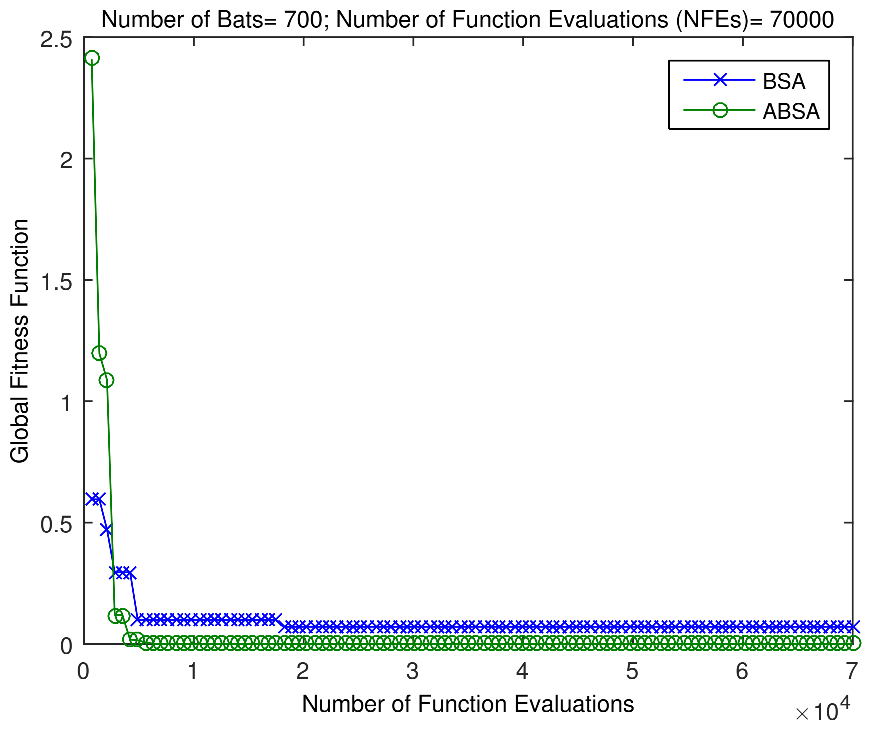
(c) 100 bats and 25 iterations



(d) 100 bats and 100 iterations



(e) 700 bats and 25 iterations



(f) 700 bats and 100 iterations

Figure 4.4: Rastrigin functions: comparison of performance of the original BSA and the ABSA

4.2.2 Performance of adaptive bats sonar algorithm on *black-box optimisation benchmarking* 2013 functions

This section deals with performance assessment of ABSA on the *black-box optimisation benchmarking* (BBOB) 2013 which is taken from Finck *et al.*. The authors established the test functions to evaluate the performance of the algorithm on the typical difficulties that occur in continuous domain search.

In conjunction to that, a generic algorithm for particle swarm optimisation (PSO)[1] also was tested on the same testbed. Here, the PSO is chosen based on few good points. PSO is a metaheuristic population-based search methods which established since 1995. The algorithm is based on the research of bird flock movement behaviour. PSO move from a set of points (population) to another set of points in a single iteration with likely improvement using a combination of deterministic and probabilistic rules. PSO search for the optimal solution by updating generations. Since the ABSA and PSO are supposed to find a solution to a given objective function but employ different strategies and computational effort, it is appropriate to compare their performance. The comparison is also made to show that ABSA is able to be at par with this well-known swarm intelligence algorithm type for solving any required single objective optimisation problems.

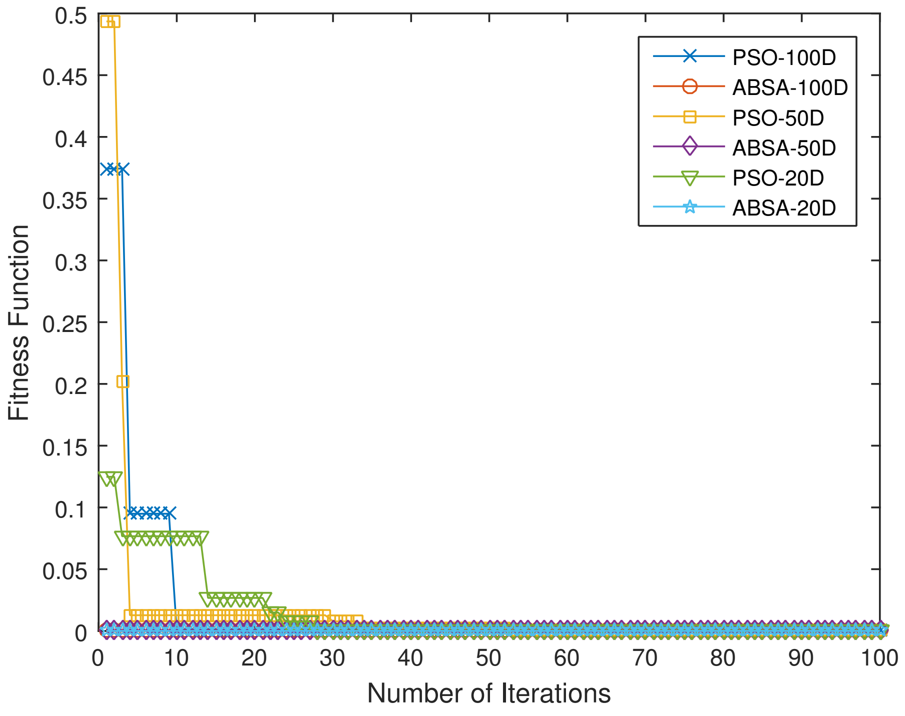
Five noiseless functions out of 24 noise-free real-parameter single objective optimisation benchmark test functions of BBOB 2013 were selected to be the test functions for ABSA and PSO. Each one of the nominated five functions has come from five different classes as shown in Table 4.3. The search space for all functions is defined as $[-5, 5]$ while the location of the majority of optimum x tabulated in $[-4, 4]$. Artificially, 0.0000 is chosen as the optimum function value of all functions.

For the purpose of this simulation, all considered algorithms single run for 20 dimensions, 50 dimensions and 100 dimensions. The overall simulation results recorded in Table 4.4 while Figure 4.5 shows the convergence of the algorithms towards global optimum function values.

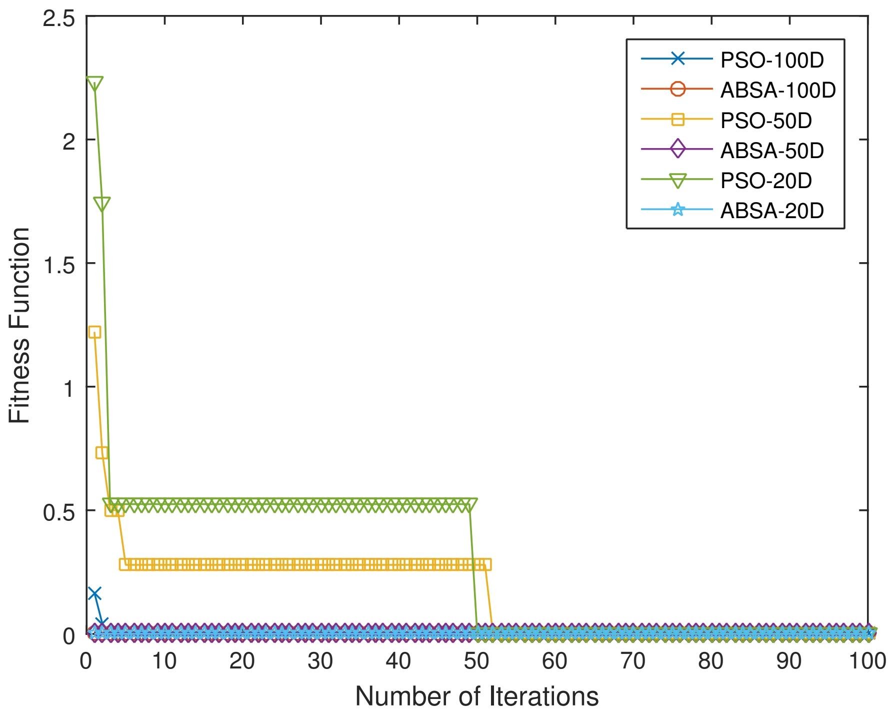
Table 4.3: Five test functions selected from BBOB 2013 functions

Function class	Function number	Function name	Description
Separable function	$f02$	Ellipsoidal	Unimodal; global quadratic and ill-conditioned function with smooth local irregularities
Function with low or moderate conditioning	$f07$	Step Ellipsoidal	Unimodal; non-separable; consists of many plateaus of different sizes
Function with high conditioning and unimodal	$f11$	Discus	Globally quadratic with local irregularities; a super-sensitive single direction in search space
Multi-modal functions with adequate global structure	$f16$	Weierstrass	Highly rugged and moderately repetitive landscape, where the global optimum is not unique
Multi-modal functions with weak global structure	$f23$	Katsuura	Highly rugged and highly repetitive function; focus on global search behaviour

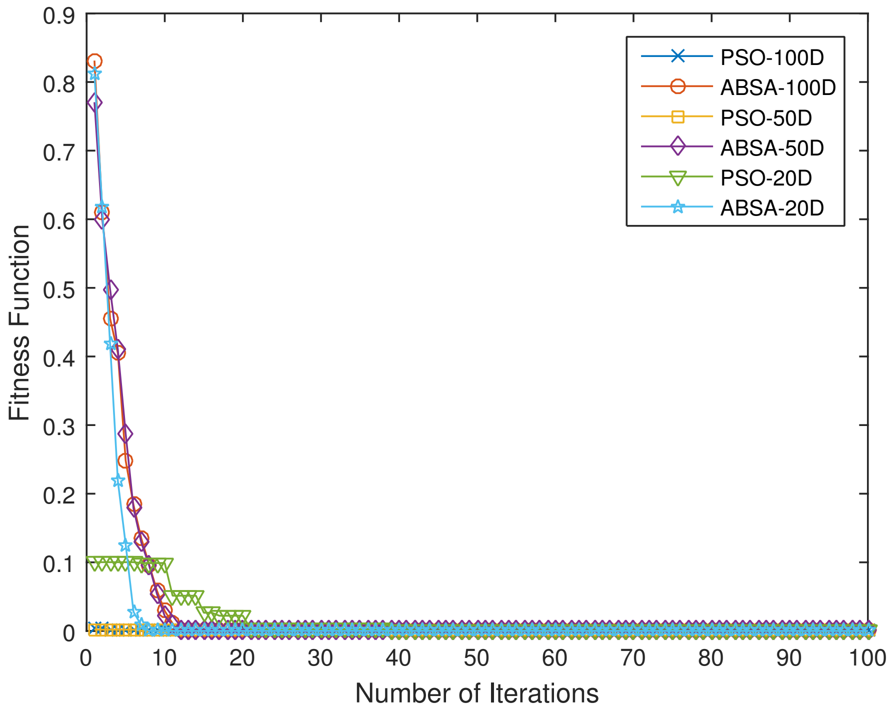
[1]The detail about PSO will be discussed in Chapter 6.



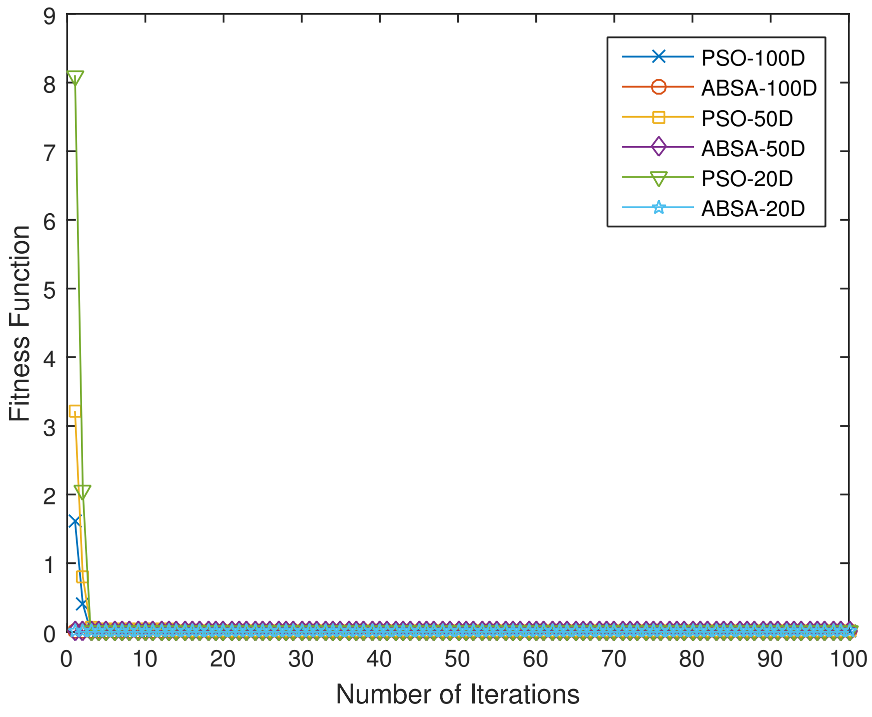
(a) Ellipsoidal function



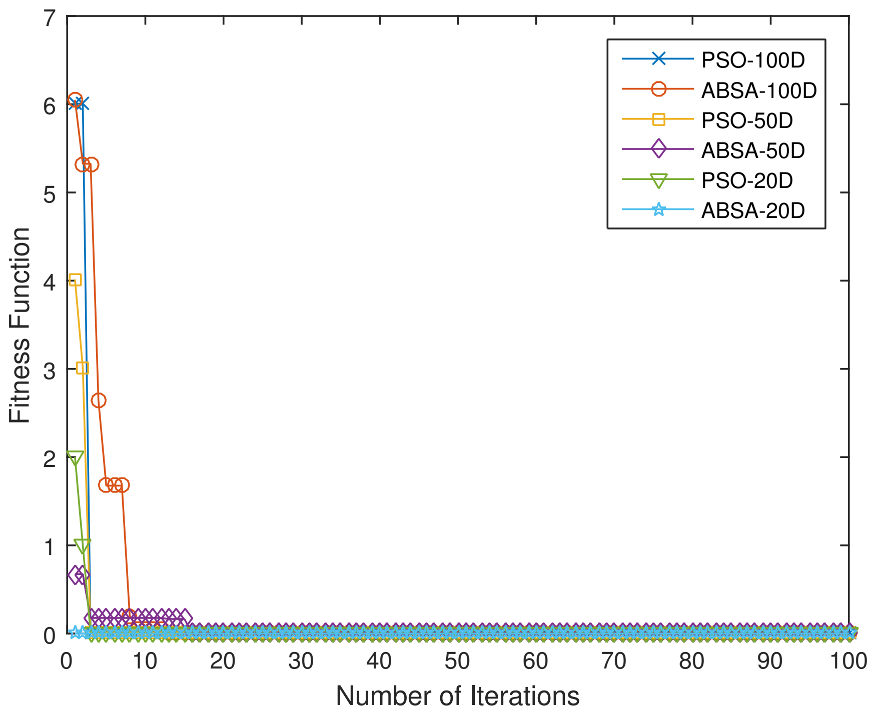
(b) Step ellipsoidal function



(c) Discus function



(d) Weierstrass function



(e) Katsuura function

Figure 4.5: Comparison of convergence performances toward optimum function value between ABSA and PSO

Table 4.4: Simulation results of considered algorithms on BBOB2013 functions

Function name	Number of dimension	Time to finish (*seconds*) PSO	Time to finish (*seconds*) ABSA	Max-Min value of x PSO	Max-Min value of x ABSA	Optimum value of $F(x)$ PSO	Optimum value of $F(x)$ ABSA
Ellipsoidal	20	0.2807	3.8531	[4.4272,-2.5453]	[3.0291,-1.5096]	$2.3345e^{-4}$	$1.2702e^{-10}$
	50	2.3545	9.0839	[4.5685,-2.7421]	[3.0935,-3.1312]	$7.2829e^{-4}$	$2.0649e^{-11}$
	100	0.4708	8.5256	[4.5632,-3.2764]	[3.3612,-3.2594]	$1.4007e^{-5}$	$2.5675e^{-7}$
Step Ellipsoidal	20	0.5033	4.9439	[4.6230,-1.5830]	[4.9998,-2.4871]	$2.1840e^{-4}$	$1.0000e^{-5}$
	50	1.2535	12.2634	[4.2198,-0.5049]	[4.0000,-2.8457]	$2.4778e^{-3}$	$1.0015e^{-5}$
	100	0.1489	12.5494	[4.0154,-4.9985]	[4.0088,-2.8772]	$4.3443e^{-3}$	$8.8924e^{-5}$
Discus	20	0.7512	2.5495	[2.9672,-1.6030]	[3.0654,-2.2418]	$1.6828e^{-6}$	$1.5444e^{-5}$
	50	1.1270	7.8474	[2.9895,-5.0000]	[3.7043,-2.9439]	$4.8821e^{-9}$	$4.5173e^{-5}$
	100	2.9454	9.5389	[3.3861,-5.0000]	[3.5868,-2.7154]	$7.0404e^{-7}$	$5.2313e^{-6}$
Weierstrass	20	0.0690	11.4833	[4.9997,-1.0053]	[4.9999,-2.7486]	$1.8412e^{-5}$	$-4.9629e^{-9}$
	50	0.3853	14.8405	[2.7328,-1.5061]	[4.9999,-2.4947]	$6.7763e^{-4}$	$-2.4373e^{-10}$
	100	2.8607	35.6994	[3.4229,-4.5920]	[2.7911,-4.0085]	$2.4310e^{-7}$	$2.0206e^{-2}$
Katsuura	20	0.4989	24.8136	[4.0000,-1.0086]	[2.7652,-5.0000]	$3.2532e^{-4}$	$1.5669e^{-11}$
	50	1.1724	26.7145	[4.6880,-2.9794]	[2.9451,-5.0000]	$4.6207e^{-6}$	$5.0208e^{-11}$
	100	2.4687	28.0494	[4.6306,-2.5353]	[2.8774,-5.0000]	$1.1605e^{-5}$	$3.4332e^{-12}$

According to the results shown in Table 4.4, the performance of the ABSA was at least at par as compared to PSO for all five considered test functions. Indeed, ABSA was able to achieve better global optimum function value for all cases compared to PSO. Even though the PSO was able to record the short duration of time to finish (in *seconds*) to global optimum as compared to ABSA in all dimensions of all test functions, this assessment can be waived out. This shows that the steps in ABSA algorithm were a little bit longer than in PSO that they make ABSA consumed much time to end the iteration.

Without a doubt, the good characteristics of bats behaviour embedded inside the ABSA make the algorithm able to start the searching process as close as possible to the best global optimum solution as compared to the standard PSO algorithm. These were shown from the convergence graphs as plotted in Figure 4.5a to Figure 4.5d where ABSA is able to find the global optimum solution less than 1.0000 within first 10 iterations before it starts to moves to the best global optimum solution later. These applied to all dimensions.

In contrast, PSO approximately starts to reach a reasonable optimum solution only after 10 iterations. However, for the Katsuura function (Figure 4.5e), the fact above does not apply as in the first 10 iterations, ABSA and PSO reached the global optimum solution theoretically far from the final global best solution. These are due to the characteristics of the test function itself.

4.2.3 Performance of adaptive bats sonar algorithm on established single objective optimisation benchmark test functions

There are many benchmark test functions that can be used for testing and validating the algorithm. Ten single objective optimisation benchmark test functions, as summarised in Table 4.5 are used to show the efficiency of ABSA. The first three test functions (FN01, FN02 and FN03) have previously been used to demonstrate the performance of the original BSA. All the three test functions have maximum values at their optimum. The remaining test functions have minimum values as their optimum . In this validation, the functions FN04, FN05, FN06 and FN07 were run in three different dimensions, namely three dimensions (FN0*a), five dimensions (FN0*b) and ten dimensions (FN0*c).

Two other algorithms are also tested on the same 10 test functions as in Table 4.5 to verify the performance of ABSA on a comparative basis. The algorithms are bats sonar algorithm (BSA) and bat algorithm (BA). The original algorithm parameters are used with BSA. These were three bats, five beams (N) in each transmitted signal and the angle between any two successive beams was fixed at $\pi \setminus 12$. Similarly, the standard algorithm parameters are used with BA. These were population size of 50, pulse rate (r) equal to 0.5, loudness (A) fixed at 0.25 and random number less than 1 for beta (β).

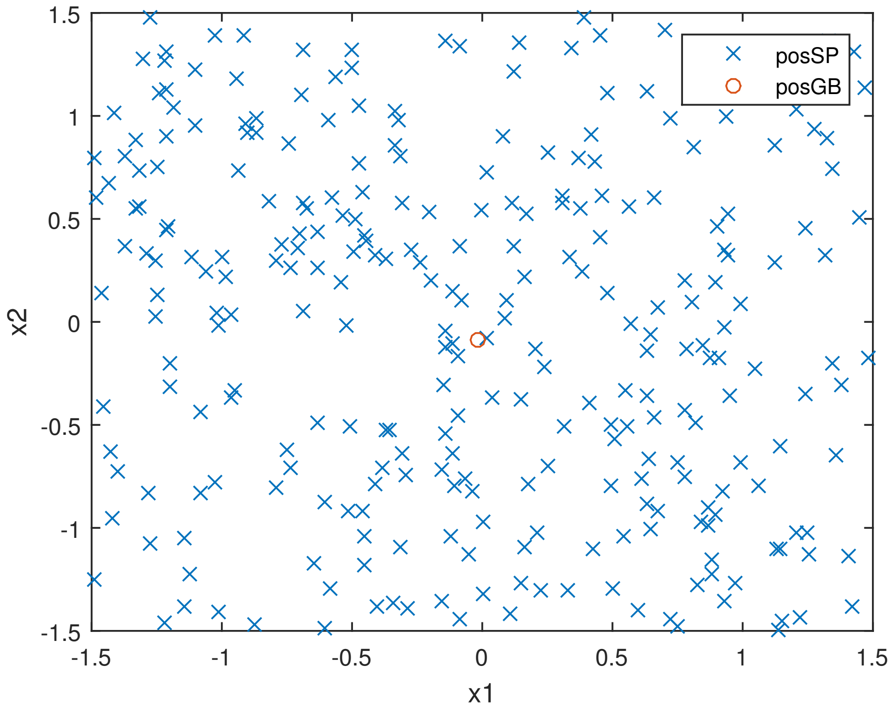
Each algorithm was run 30 times to allow it to carry out meaningful statistical analysis. The maximum number of iterations for each run was set to 100. All three algorithms on the ten function evaluations obtained the result of *best*, *mean*, *worst* and *standard deviation* values. To evaluate the statistical significance of the ABSA, one-way analysis of variance (ANOVA) with post-hoc test (Dunnett's test type) was applied, and the null hypothesis was rejected at the confidence level of 5%.

Figures 4.6a - 4.6d show the search patterns of 1000 bats positions using ABSA for 2 dimension De Jong function. Its global minimum $F(x) = 0$ was obtainable for $x_i = 0$, $i = 1, \ldots, N$. In iteration 1, 1000 bats scattered at various locations in the designated search space. Bats started to converge to the final value of x_i as the iteration increased. At iteration 50, all 1000 bats settled to the optimum values of $x_1 = 0$ and $x_2 = 0$.

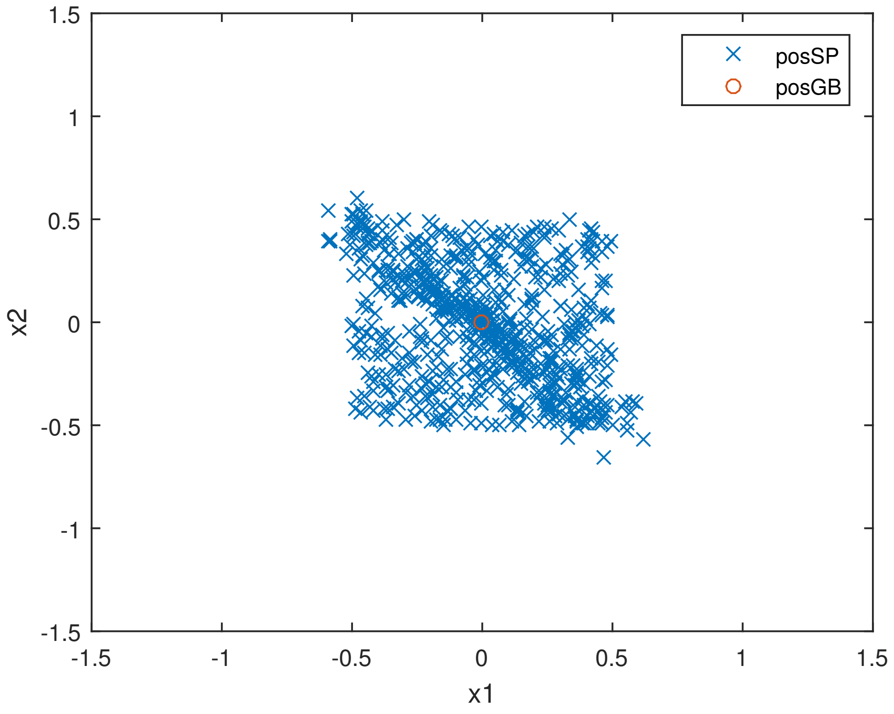
The results of the computer simulations for ABSA algorithm are given in Table 4.6. As noted, the algorithm achieved the global optimum value with zero or very small *standard deviation*. Comparative results of the *best*, *worst* and *mean* solutions with *standard deviation* values of the investigated algorithms are shown in Tables 4.7, 4.8, 4.9 and 4.10 respectively.

Table 4.5: Benchmark functions used to validate the performance of ABSA

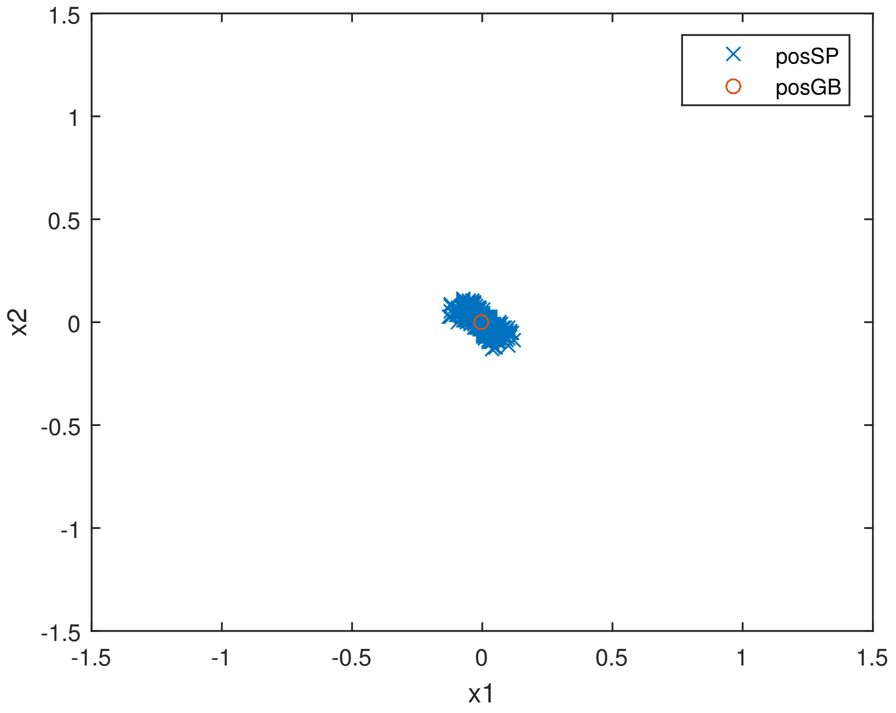
Label	Function name (type)	Function	Optimum value of $F(x)$	Range of solution space
FN01	Third-order polynomial with a single variable (Max)	$F(x) = x^3 - 5x^2 - 20x$	15.4564	$-65.12 \leq x \leq 65.12$
FN02	Polynomial with two variables (Max)	$F(x) = x_1^3 - 5x_1^2 - 2.04x_2^2 + 4x_2$	1.9608	$-3 \leq (x_1, x_2) \leq 3$
FN03	Exponential with two variables (Max)	$F(x) = x_1 \exp^{(-x_1^2 - x_2^2)}$	0.4289	$-2 \leq (x_1, x_2) \leq 2$
FN04	De Jong's (Min)	$F(x) = \sum_{i=1}^{n} x_i^2$	0.0000	$-5.12 \leq x_i \leq 5.12,\ i = 1, \ldots, N$
FN05	Weighted sphere model (Min)	$F(x) = \sum_{i=1}^{n} (i \cdot x_i^2)$	0.0000	$-5.12 \leq x_i \leq 5.12,\ i = 1, \ldots, N$
FN06	Shwefel's (Min)	$F(x) = \sum_{i=1}^{n} (i \cdot x_i^2)$	0.0000	$-65.536 \leq x_i \leq 65.536,\ i = 1, \ldots, N$
FN07	Rosenbrock's valley (Min)	$F(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2) + (1 - x_i)^2]$	0.0000	$-2.048 \leq x_i \leq 2.048,\ i = 1, \ldots, N$
FN08	Easom's (Min)	$F(x) = -\cos x_1 \cos x_2 \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	-1.0000	$-100 \leq (x_1, x_2) \leq 100$
FN09	Goldstein-Price's (Min)	$F(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	3.0000	$-2 \leq (x_1, x_2) \leq 2$
FN10	Booth's (Min)	$F(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	0.0000	$-10 \leq (x_1, x_2) \leq 10$



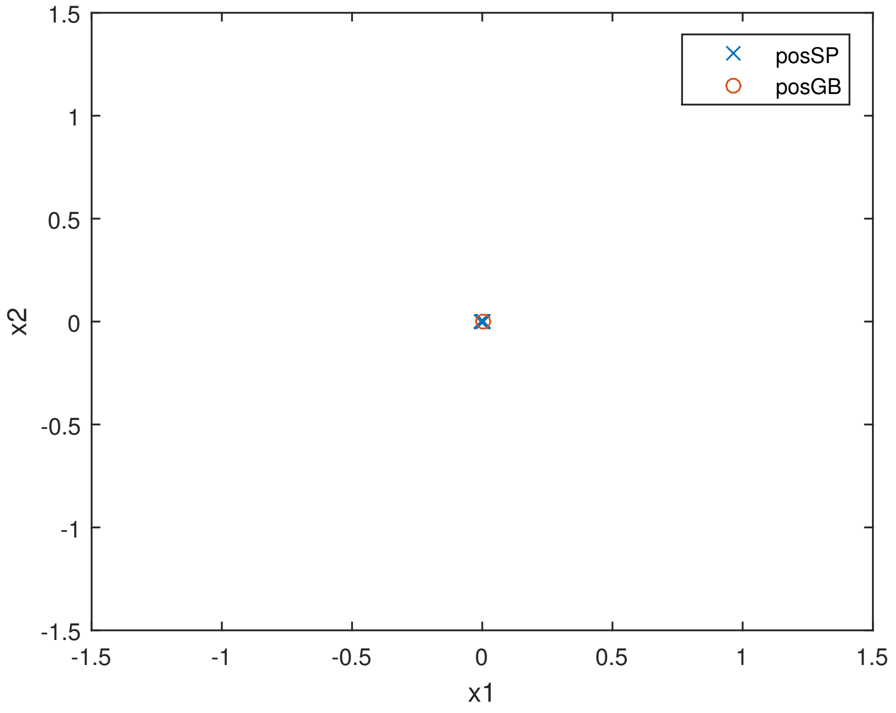
(a) Iteration 1



(b) Iteration 5



(c) Iteration 20



(d) Iteration 50

Figure 4.6: Locations of 1000 bats using ABSA for 2 dimensional De Jong function

As seen in Table 4.7, the ABSA approach found the exact or close global optimum value of thirteen out of the eighteen functions (FN02, FN04a-c, FN05a-c, FN06a-c and FN07a-c) through 30 runs. From one function (FN01), ABSA produced results similar to both BA and BSA. Moreover, ABSA achieved similar *best* value with BSA on FN03, with BA in three functions, namely FN08, FN09 and FN10. Overall, as noted, the ABSA *best* results were superior to those achieved with BSA and BA.

Table 4.6: Statistical results obtained for ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

Function number	Dim	Optimum $F(x)$	*Best*	*Mean*	*Worst*	*Standard deviation*
FN01	1	15.4564	15.4564	15.4564	15.4564	0.0000
FN02	2	1.9608	1.9608	1.9608	1.9608	0.0000
FN03	2	0.4289	0.4289	0.4289	0.4289	0.0000
FN04a	3	0.0000	$2.2810e^{-13}$	$1.2374e^{-9}$	$9.6814e^{-9}$	$2.4540e^{-9}$
FN04b	5	0.0000	$1.2726e^{-11}$	$2.1789e^{-8}$	$2.3951e^{-7}$	$5.2963e^{-8}$
FN04c	10	0.0000	$1.3720e^{-4}$	$5.4975e^{-2}$	$3.9510e^{-1}$	$1.0842e^{-1}$
FN05a	3	0.0000	$4.8111e^{-12}$	$4.0332e^{-10}$	$1.5621e^{-9}$	$4.5575e^{-10}$
FN05b	5	0.0000	$4.4514e^{-11}$	$1.1890e^{-8}$	$6.3666e^{-8}$	$1.5027e^{-8}$
FN05c	10	0.0000	$2.6957e^{-4}$	$2.5186e^{-2}$	$6.6100e^{-2}$	$1.7923e^{-2}$
FN06a	3	0.0000	$1.1643e^{-11}$	$2.0870e^{-9}$	$7.3697e^{-9}$	$2.1982e^{-9}$
FN06b	5	0.0000	$5.2555e^{-10}$	$5.4807e^{-8}$	$4.2394e^{-7}$	$1.0912e^{-7}$
FN06c	10	0.0000	$6.2212e^{-5}$	$5.6951e^{-3}$	$2.3500e^{-2}$	$7.7790e^{-3}$
FN07a	3	0.0000	$1.8990e^{-12}$	$2.9536e^{-9}$	$1.8916e^{-8}$	$4.3566e^{-9}$
FN07b	5	0.0000	$3.3335e^{-11}$	$1.6080e^{-7}$	$4.6234e^{-6}$	$8.4319e^{-7}$
FN07c	10	0.0000	$2.3001e^{-12}$	$3.9551e^{-9}$	$3.0717e^{-8}$	$7.6405e^{-9}$
FN08	2	-1.0000	-1.0000	-1.0000	-1.0000	0.0000
FN09	2	3.0000	3.0000	3.0000	3.0000	0.0000
FN10	2	0.0000	0.0000	0.0000	0.0000	0.0000

As noted in the *worst* solution results given in Table 4.8, ABSA outperformed BA and BSA in all eighteen functions tested. Even for the *worst* results, ABSA successfully achieved accurate or very near accurate results to global optimum points. Similarly, on the *mean* solutions as shown in Table 4.9, ABSA achieved accurate performance as compared to BA and BSA for seventeen out of the eighteen function evaluations. Even though for the FN04c the BA achieved better optimum solution compared to ABSA, the gap between them was small.

As far as *standard deviation* is concerned, the results in Table 4.10 show the best precision exhibited by ABSA. Less variation (some functions, no variation) of optimum solution from the *mean* values was produced by implementing ABSA on all test functions except FN04c. For FN04c, BA was able to achieve smaller *standard deviation* value compared to that achieved with ABSA but the difference was not significant.

Table 4.11 shows a comparison of the performance of ABSA with BA and BSA using one-way analysis of variance (ANOVA) on the *mean* value $\pm$ *standard deviation* of the global optimum. It is noted that at 95% confident interval, ABSA was statistically significant to achieve better global optimum solution ahead of BA and BSA. Overall, it can be concluded that ABSA outperforms BA and BSA for accuracy and precision to search for a global optimum solution either in maximisation or minimisation problems.

Table 4.7: The *best* solution obtained by BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

Function number	Dim	Optimum $F(x)$	BA	BSA	ABSA
FN01	1	15.4564	15.4564	15.4564	15.4564
FN02	2	1.9608	1.9832	1.9606	1.9608
FN03	2	0.4289	0.4280	0.4289	0.4289
FN04a	3	0.0000	$1.1985e^{-7}$	$1.8211e^{-5}$	$2.2810e^{-13}$
FN04b	5	0.0000	$1.0854e^{-6}$	$3.9700e^{-2}$	$1.2726e^{-11}$
FN04c	10	0.0000	$1.2000e^{-3}$	$8.0770e^{-1}$	$1.3720e^{-4}$
FN05a	3	0.0000	$2.5850e^{-7}$	$1.4324e^{-9}$	$4.8111e^{-12}$
FN05b	5	0.0000	$1.1000e^{-3}$	$5.7284e^{-5}$	$4.4514e^{-11}$
FN05c	10	0.0000	$4.6000e^{-3}$	$8.6000e^{-3}$	$2.6957e^{-4}$
FN06a	3	0.0000	$7.5661e^{-8}$	$1.7246e^{-9}$	$1.1643e^{-11}$
FN06b	5	0.0000	$1.0000e^{-3}$	$3.3504e^{-4}$	$5.2555e^{-10}$
FN06c	10	0.0000	$2.3800e^{-2}$	$4.5000e^{-3}$	$6.2212e^{-5}$
FN07a	3	0.0000	$3.4954e^{-9}$	$3.5720e^{-7}$	$1.8990e^{-12}$
FN07b	5	0.0000	$2.1000e^{-3}$	$1.3993e^{-4}$	$3.3335e^{-11}$
FN07c	10	0.0000	$8.6000e^{-3}$	$2.7000e^{-3}$	$2.3001e^{-12}$
FN08	2	-1.0000	-1.0000	-0.9999	-1.0000
FN09	2	3.0000	3.0000	3.0060	3.0000
FN10	2	0.0000	0.0000	0.0001	0.0000

Table 4.8: The *worst* solution obtained by BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

Function number	Dim	Optimum $F(x)$	BA	BSA	ABSA
FN01	1	15.4564	15.3302	15.4175	15.4564
FN02	2	1.9608	1.9006	1.9032	1.9608
FN03	2	0.4289	0.4024	0.4221	0.4289
FN04a	3	0.0000	$9.8722e^{-5}$	$8.5000e^{-3}$	$9.6814e^{-9}$
FN04b	5	0.0000	$6.7300e^{-2}$	$6.9350e^{-1}$	$2.3951e^{-7}$
FN04c	10	0.0000	$1.1070e^{-1}$	1.8506	$3.9510e^{-1}$
FN05a	3	0.0000	$8.6962e^{-4}$	$1.4619e^{-5}$	$1.5621e^{-9}$
FN05b	5	0.0000	$5.1300e^{-2}$	$9.5000e^{-3}$	$6.3666e^{-8}$
FN05c	10	0.0000	$8.8270e^{-1}$	$9.8190e^{-1}$	$6.6100e^{-2}$
FN06a	3	0.0000	$8.2515e^{-4}$	$3.9698e^{-5}$	$7.3697e^{-9}$
FN06b	5	0.0000	$8.9700e^{-2}$	$9.4000e^{-2}$	$4.2394e^{-7}$
FN06c	10	0.0000	$4.9420e^{-1}$	$9.0690e^{-1}$	$2.3500e^{-2}$
FN07a	3	0.0000	$9.4882e^{-4}$	$8.5589e^{-4}$	$1.8916e^{-8}$
FN07b	5	0.0000	$9.9000e^{-2}$	$1.4600e^{-2}$	$4.6234e^{-6}$
FN07c	10	0.0000	$8.7030e^{-1}$	$9.3110e^{-1}$	$3.0717e^{-8}$
FN08	2	-1.0000	-1.4070	-0.8110	-1.0000
FN09	2	3.0000	3.4618	3.8640	3.0000
FN10	2	0.0000	0.3314	0.1215	0.0000

Table 4.9: The *mean* solution obtained by BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

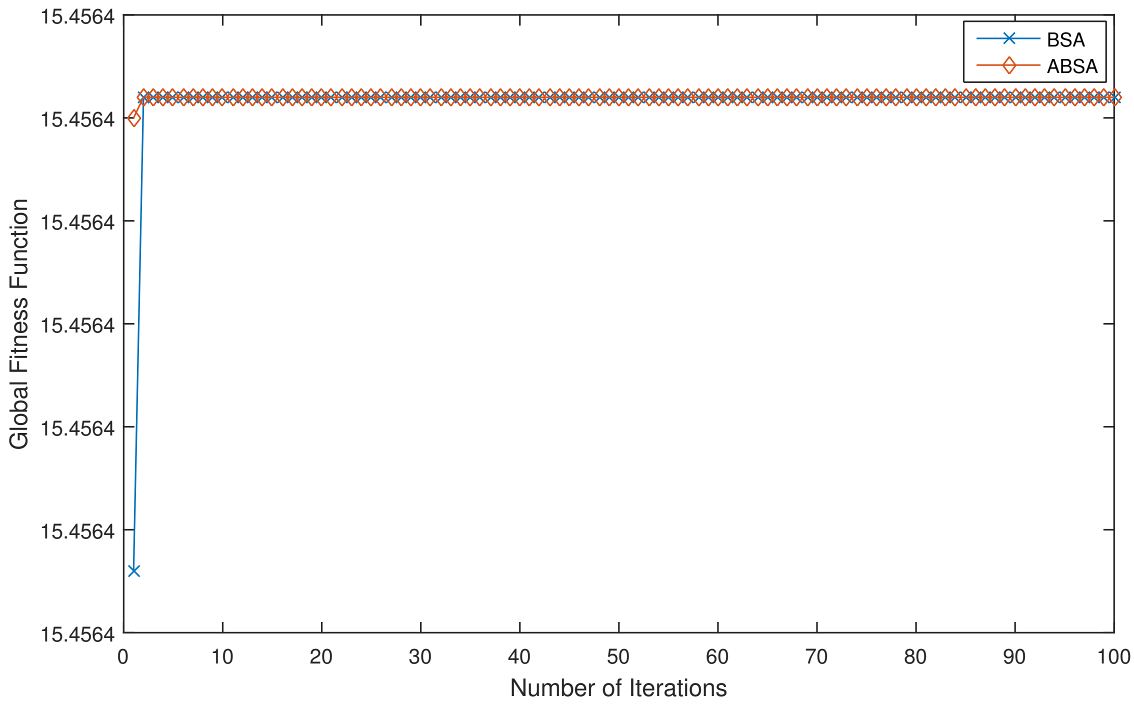
Function number	Dim	Optimum $F(x)$	BA	BSA	ABSA
FN01	1	15.4564	15.4458	15.4438	15.4564
FN02	2	1.9608	1.9308	1.9401	1.9608
FN03	2	0.4289	0.4177	0.4262	0.4289
FN04a	3	0.0000	$3.6929e^{-5}$	$2.6683e^{-3}$	$1.2374e^{-9}$
FN04b	5	0.0000	$5.1481e^{-3}$	$4.1950e^{-1}$	$2.1789e^{-8}$
FN04c	10	0.0000	$2.6150e^{-2}$	1.4665	$5.4975e^{-2}$
FN05a	3	0.0000	$8.0776e^{-5}$	$1.1634e^{-6}$	$4.0332e^{-10}$
FN05b	5	0.0000	$1.4917e^{-2}$	$3.6329e^{-3}$	$1.1890e^{-8}$
FN05c	10	0.0000	$3.4812e^{-1}$	$4.1136e^{-1}$	$2.5186e^{-2}$
FN06a	3	0.0000	$8.6964e^{-5}$	$3.2073e^{-6}$	$2.08470e^{-9}$
FN06b	5	0.0000	$2.4963e^{-2}$	$3.0683e^{-2}$	$5.4807e^{-8}$
FN06c	10	0.0000	$1.5900e^{-1}$	$3.4829e^{-1}$	$5.6951e^{-3}$
FN07a	3	0.0000	$5.9211e^{-4}$	$3.7671e^{-4}$	$2.9536e^{-9}$
FN07b	5	0.0000	$3.5097e^{-2}$	$4.5607e^{-3}$	$1.6080e^{-7}$
FN07c	10	0.0000	$3.9344e^{-1}$	$1.9216e^{-1}$	$3.9551e^{-9}$
FN08	2	-1.0000	-1.2144	-0.9554	-1.0000
FN09	2	3.0000	3.0938	3.3215	3.0000
FN10	2	0.0000	0.0869	0.0331	0.0000

Table 4.10: The *standard deviation* obtained by BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

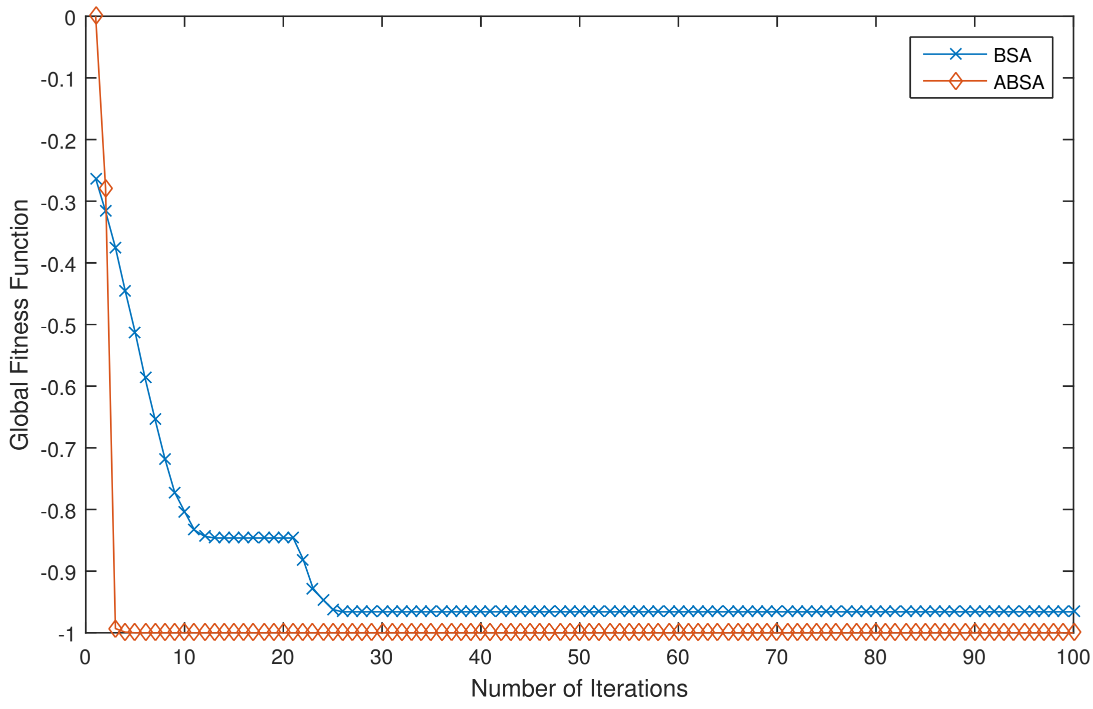
Function number	Dim	Optimum $F(x)$	BA	BSA	ABSA
FN01	1	15.4564	0.0278	0.0095	0.0000
FN02	2	1.9608	0.0188	0.0184	0.0000
FN03	2	0.4289	0.0081	0.0025	0.0000
FN04a	3	0.0000	$3.2411e^{-5}$	$2.3319e^{-3}$	$2.4540e^{-9}$
FN04b	5	0.0000	$1.2468e^{-2}$	$1.7864e^{-1}$	$5.2963e^{-8}$
FN04c	10	0.0000	$2.4978e^{-2}$	$3.3193e^{-1}$	$1.0842e^{-1}$
FN05a	3	0.0000	$1.9681e^{-4}$	$2.7481e^{-6}$	$4.5575e^{-10}$
FN05b	5	0.0000	$1.2349e^{-2}$	$3.0154e^{-3}$	$1.5027e^{-8}$
FN05c	10	0.0000	$2.5533e^{-1}$	$3.0597e^{-1}$	$1.7923e^{-2}$
FN06a	3	0.0000	$1.9133e^{-4}$	$8.3095e^{-6}$	$2.1982e^{-9}$
FN06b	5	0.0000	$1.8628e^{-2}$	$3.4283e^{-2}$	$1.0912e^{-7}$
FN06c	10	0.0000	$1.0826e^{-1}$	$2.5159e^{-1}$	$7.7790e^{-3}$
FN07a	3	0.0000	$2.5279e^{-4}$	$2.8526e^{-4}$	$4.3566e^{-9}$
FN07b	5	0.0000	$3.5821e^{-2}$	$4.2380e^{-3}$	$8.4319e^{-7}$
FN07c	10	0.0000	$2.7202e^{-1}$	$2.7346e^{-1}$	$7.6405e^{-9}$
FN08	2	-1.0000	0.1308	0.0438	0.0000
FN09	2	3.0000	0.2003	0.3021	0.0000
FN10	2	0.0000	0.0818	0.0356	0.0000

Table 4.11: Performance comparison using one-way analysis of variance (ANOVA) between BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs of 100 iterations each

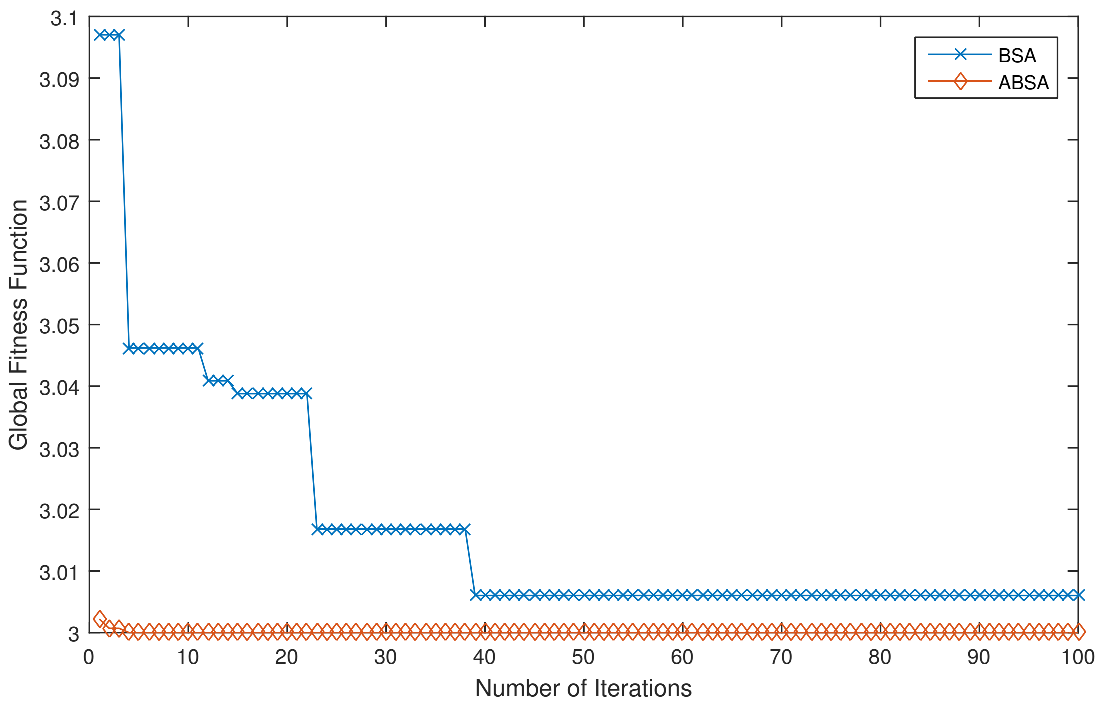
FN No.	BA	BSA	ABSA	Significantly
FN01	15.4564 ± 0.0278	15.4538 ± 0.0095	15.4564 ± 0.0000	Yes
FN02	1.9308 ± 0.0188	1.9401 ± 0.0184	1.9608 ± 0.0000	Yes
FN03	0.4177 ± 0.0081	0.4262 ± 0.0025	0.4289 ± 0.0000	Yes
FN04a	$3.6929e^{-5} \pm 3.2411e^{-5}$	$2.6683e^{-3} \pm 2.3319e^{-3}$	$1.2374e^{-9} \pm 2.4540e^{-9}$	Yes
FN04b	$5.1481e^{-3} \pm 1.2468e^{-2}$	$4.1950e^{-1} \pm 1.7864e^{-1}$	$2.1789e^{-8} \pm 5.2963e^{-8}$	Yes
FN04c	$2.6150e^{-2} \pm 2.4978e^{-2}$	$1.4665 \pm 3.3193e^{-1}$	$5.4975e^{-2} \pm 1.0842e^{-1}$	Yes
FN05a	$8.0776e^{-5} \pm 1.9681e^{-4}$	$1.1634e^{-6} \pm 2.7481e^{-6}$	$4.0332e^{-10} \pm 4.5575e^{-10}$	Yes
FN05b	$1.4917e^{-2} \pm 1.2349e^{-2}$	$3.6329e^{-3} \pm 3.0154e^{-3}$	$1.1890e^{-8} \pm 1.5027e^{-8}$	Yes
FN05c	$3.4812e^{-1} \pm 2.5533e^{-1}$	$4.1136e^{-1} \pm 3.0597e^{-1}$	$2.5186e^{-2} \pm 1.7923e^{-2}$	Yes
FN06a	$8.6964e^{-5} \pm 1.9133e^{-4}$	$3.2073e^{-6} \pm 8.3095e^{-6}$	$2.0870e^{-9} \pm 2.1982e^{-9}$	Yes
FN06b	$2.4963e^{-2} \pm 1.8628e^{-2}$	$3.0683e^{-2} \pm 3.4283e^{-2}$	$5.4807e^{-8} \pm 1.0912e^{-7}$	Yes
FN06c	$1.5900e^{-} \pm 1.0826e^{-1}$	$3.4829e^{-1} \pm 2.5159e^{-1}$	$5.6951e^{-3} \pm 7.7790e^{-3}$	Yes
FN07a	$5.9211e^{-4} \pm 2.5279e^{-4}$	$3.7671e^{-4} \pm 2.8526e^{-4}$	$2.9536e^{-9} \pm 4.3566e^{-9}$	*Yes*
FN07b	$3.5097e^{-2} \pm 3.5821e^{-2}$	$4.5607e^{-3} \pm 4.2380e^{-3}$	$1.6080e^{-7} \pm 8.4319e^{-7}$	Yes
FN07c	$3.9344e^{-1} \pm 2.7202e^{-1}$	$1.9216e^{-1} \pm 2.7346e^{-1}$	$3.9551e^{-9} \pm 7.6405e^{-9}$	Yes
FN08	-1.2144 ± 0.1308	-0.9554 ± 0.0438	-1.0000 ± 0.0000	Yes
FN09	3.0938 ± 0.2003	3.3215 ± 0.3021	3.0000 ± 0.0000	Yes
FN10	0.0869 ± 0.0818	0.0331 ± 0.0356	0.0000 ± 0.0000	Yes



(a) Third-order polynomial with single variable



(b) Easom’s function



(c) Goldstein-Price's function

Figure 4.7: Convergence to global best fitness function achieved by ABSA and BSA for selected test functions

Figure 4.7 shows convergence to global best fitness function value achieved by the ABSA as compared to BSA for selected benchmark test functions:

- Third-order polynomial with a single variable
- Easom's function
- Goldstein-Price's function

However, these do not account for differing computational costs, as in reality, ABSA has taken longer time than BSA to arrive at a maximum number of iteration. This is due to the new structure and additional steps incorporated into the original BSA to arrive at the ABSA. The graphical results show that ABSA was able to converge to global best fitness for each function in a smaller number of iterations compared to BSA. Moreover, with several random approaches introduced to locate the starting positions in ABSA, the algorithm is potentially able to start the search process at locations close to the optimum point and promptly move to the absolute global best point.

Table 4.12 presents the results of one-way analysis of variance (ANOVA) on the *mean* iteration value $\pm$ *standard deviation* of iteration number to arrive at a global optimum solution. The results show that at the 95% confident interval, ABSA significantly performed better than BA and BSA to converge to the global optimum solution faster. According to Figure 4.8, on average, in 100 iterations, the ABSA needed around 12% to 37% iterations to reach the global optimum solution. The algorithm outperformed BA and BSA, which took 24% to 49% and 35% to 58% iterations respectively. This implies that ABSA has faster convergence ability to a global optimum solution either for maximisation or minimisation problems as compared to BA and BSA.

Table 4.12: Performance comparison in terms of faster convergence to global optimum in 100 iterations using one-way analysis of variance (ANOVA) between BA, BSA and ABSA with 10 test functions of different dimensions over 30 independent runs

FN No.	BA	BSA	ABSA	Significantly
FN01	24.70 ± 15.12	52.13 ± 29.63	21.40 ± 8.79	Yes
FN02	47.77 ± 2.60	46.80 ± 29.51	28.67 ± 13.50	Yes
FN03	31.93 ± 12.60	51.23 ± 34.23	29.43 ± 13.88	Yes
FN04a	24.87 ± 16.87	55.37 ± 29.05	33.83 ± 11.11	Yes
FN04b	23.17 ± 13.98	48.17 ± 31.09	34.83 ± 11.11	Yes
FN04c	27.53 ± 14.49	42.77 ± 30.03	37.27 ± 8.79	Yes
FN05a	33.43 ± 10.25	56.83 ± 30.30	33.47 ± 11.75	Yes
FN05b	28.57 ± 15.93	49.03 ± 32.18	36.30 ± 9.55	Yes
FN05c	25.07 ± 12.65	58.53 ± 35.15	37.43 ± 9.26	Yes
FN06a	38.47 ± 9.78	54.30 ± 28.75	30.77 ± 12.14	Yes
FN06b	37.10 ± 7.44	44.70 ± 30.50	36.43 ± 10.81	Yes
FN06c	49.33 ± 7.37	35.67 ± 29.38	34.67 ± 11.56	Yes
FN07a	26.70 ± 15.62	51.63 ± 27.50	15.17 ± 10.02	Yes
FN07b	25.70 ± 11.76	56.47 ± 29.83	12.10 ± 5.84	Yes
FN07c	29.37 ± 11.94	55.87 ± 28.33	12.03 ± 3.37	Yes
FN08	28.20 ± 13.65	50.63 ± 29.89	24.57 ± 14.07	Yes
FN09	29.67 ± 16.58	51.00 ± 27.67	26.87 ± 14.21	Yes
FN10	25.23 ± 15.02	49.33 ± 26.75	21.90 ± 14.39	Yes

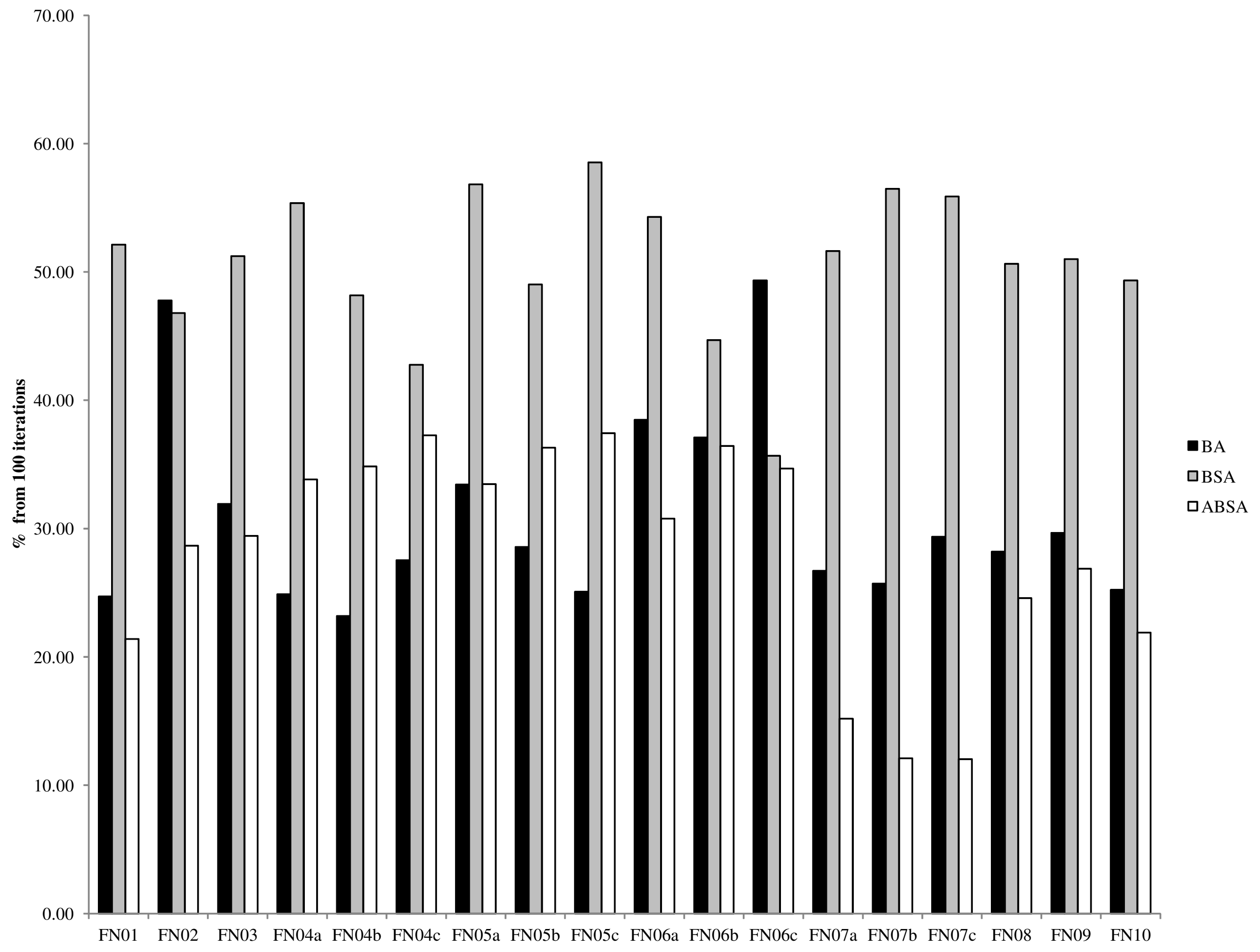


Figure 4.8: Comparison of average number of iterations to achieve global optimum solution

4.3 Application of adaptive bats sonar algorithm to solve single objective optimisation problems

4.3.1 Cost optimisation of shipping refined oil

The problem is about finding the minimum cost of refined oil (F) when shipped via the Malacca Straits to Japan in dollar per kiloliter ($\$/kL$). The optimum tanker size (x_1) in *dwt* and optimum refinery capacity (x_2) in *bbl/day* are variables of the problem. The problem has to include the crude oil cost, insurance cost, customs cost, freight cost for the oil, loading and unloading cost, sea berth cost, submarine pipe cost, storage cost, tank area cost, refining cost and freight cost of products in the linear sum as (note that 1 kL = 6.29bbl):

$$\begin{aligned}
\text{Minimise } F(x) = c_c + c_i + c_x &+ \frac{2.09e^4(x_1)^{-0.3017}}{360} + \frac{1.064e^6 a(x_1)^{0.4925}}{52.47(x_2)(360)} + \frac{0.1049(x_1)^{0.671}}{360} \\
&+ \frac{4.242e^4 a(x_1)^{0.7952} + 1.813ip(n(x_1)+1.2(x_2))^{0.861}}{52.47(x_2)(360)} + \frac{5.042e^3(x_2)^{-0.1899}}{360} \\
&+ \frac{4.25e^3 a(n(x_1)+1.2(x_2))}{52.47(x_2)(360)}
\end{aligned}$$

where

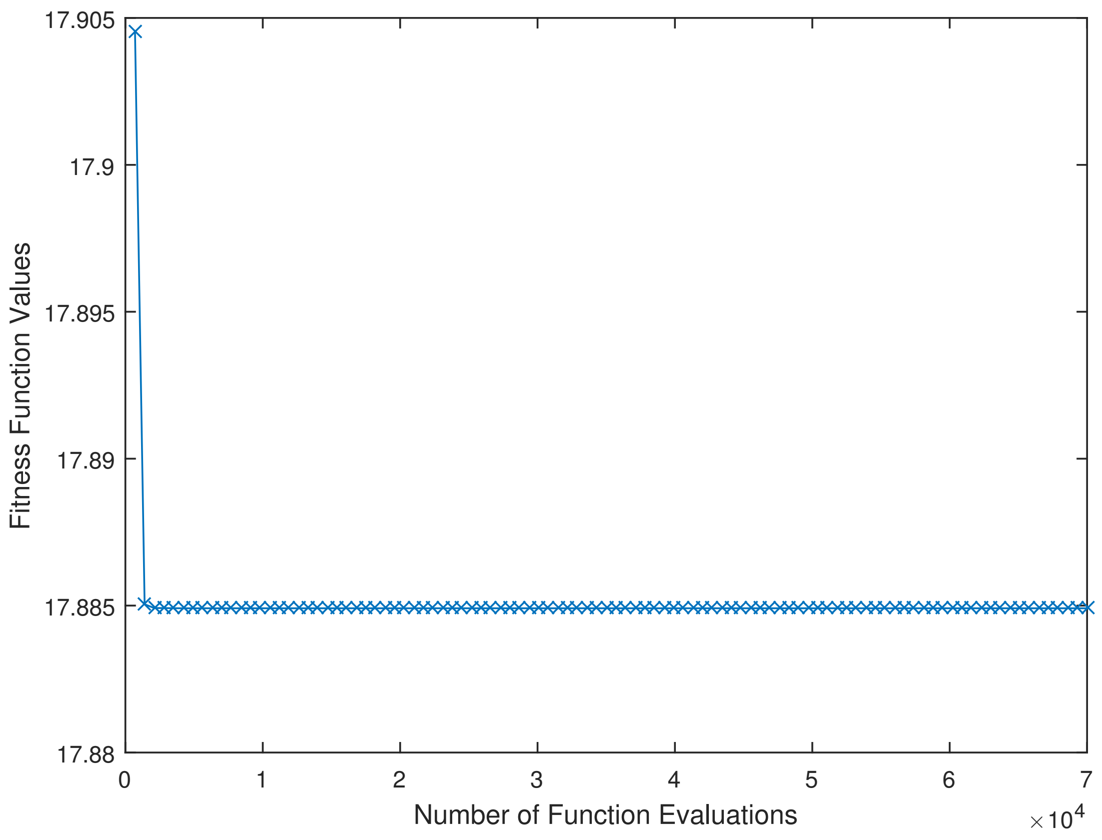
$$\begin{aligned}
&a = \text{annual fixed charges}, fraction\ (0.20) \\
&c_c = \text{crude oil price}, \$/kL\ (12.50) \\
&c_i = \text{insurance cost}, \$/kL\ (0.50) \\
&c_x = \text{customs cost}, \$/kL\ (0.90) \\
&i = \text{interest rate}\ (0.10) \\
&n = \text{number of ports}\ (2) \\
&p = \text{land price}, \$/m^2\ (700) \\
&x_1 \geq 0 \text{ and } x_2 \geq 0
\end{aligned} \tag{4.12}$$

The ABSA is applied to find the optimum cost for this problem. The ABSA is capable of finding the minimum cost of refined oil (F) in dollar per kiloliter ($\$/kL$). The results of 30 independent runs by the ABSA to solve this problem are shown in Table 4.13. According to the results, the minimum cost achieved by using ABSA is \$17.8849/kL. The value was similar for all 30 independent runs, so the *best*, *worst* or *mean* are equal as well as *standard deviation* is zero.

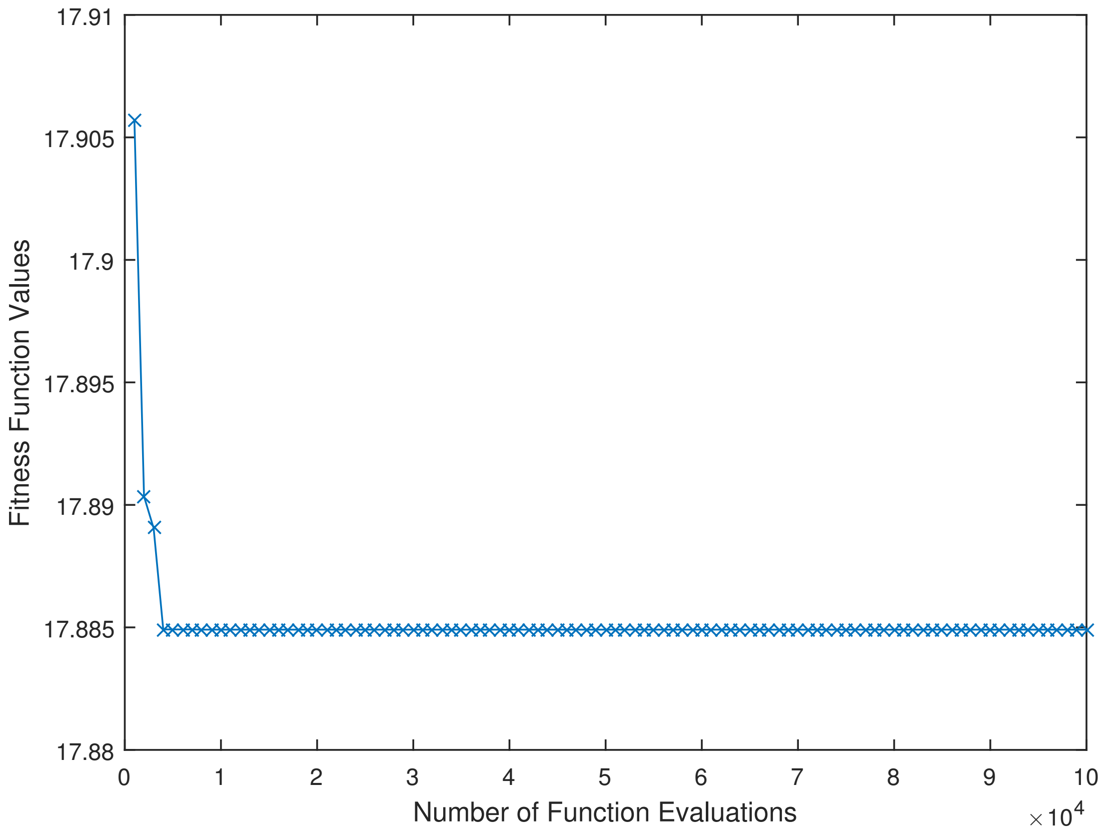
The results also recorded that 53.33% out of 30 ABSA independent runs successfully finished in less than 10 *seconds*. 23th run of the algorithm as shown in Figure 4.9a appeared as the fastest among runs that are 5.0343 *seconds* where the ABSA started to converge to optimum value during 19th iteration. Meanwhile, the 16th of the ABSA as shown in Figure 4.9b finished the slowest among runs; 99.9512 *seconds* where the convergence only occurred during the 100th iteration. Figure 4.9c shows the 8th runs of ABSA where the algorithm started to converge to the optimum value in the shortest iteration among the all 30 independent runs, which was during 18th iteration. Finally, Figure 4.10 shows the quality of the obtained variables where small ranges of variation for the tanker size and refinery capacity were achieved in all 30 independent runs of the ABSA.

Table 4.13: Result for 30 runs of ABSA to optimise the cost of shipping refined oil problem

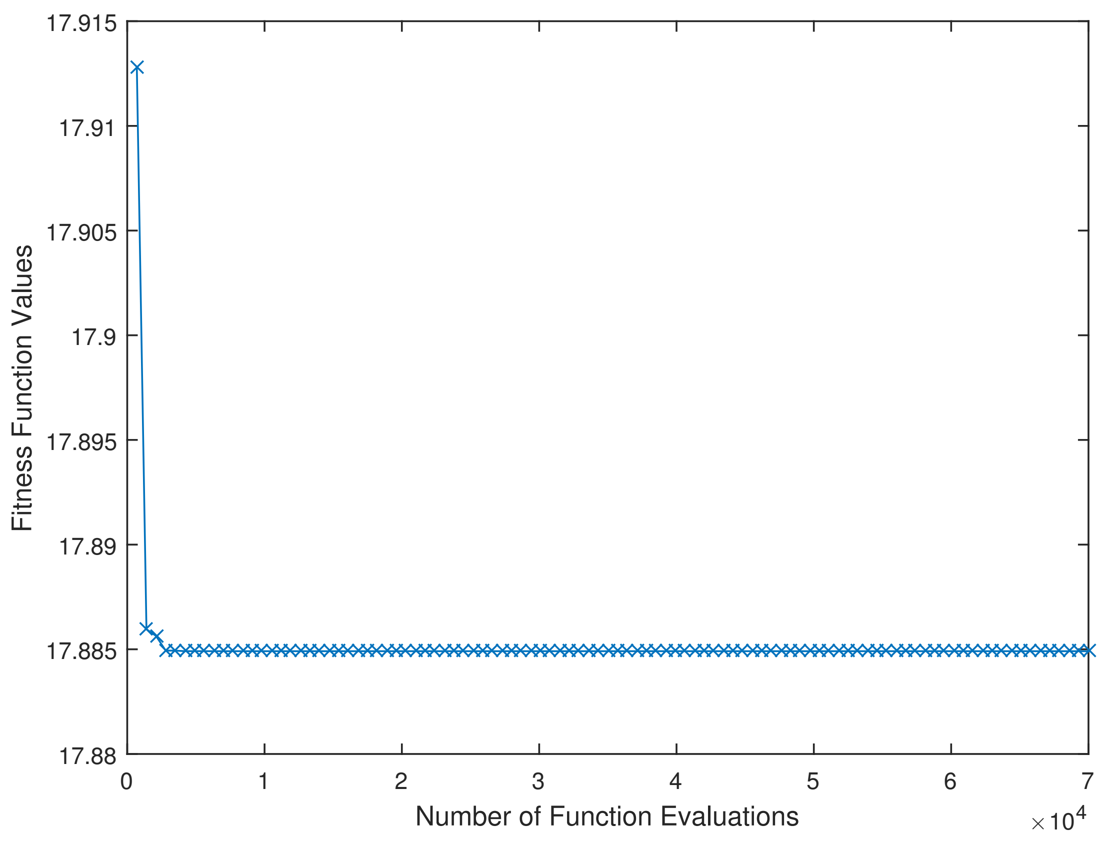
Run no.	Cost of shipping refined oil, F ($\$/kL$)	Variables: Tanker size, x_1 (dwt)	Variables: Refinery capacity, x_2 (bbl/day)	Time to finish ($seconds$)	Numbers of bats used	Iteration to converge	Number of function evaluation ($NFEs$)
1	17.8849	446967.4908	179845.3736	5.8591	700	21	70000
2	17.8849	446967.5156	179845.3803	5.4619	700	20	70000
3	17.8849	446967.5103	179845.3674	5.7946	700	21	70000
4	17.8849	446967.4991	179845.3667	13.5368	700	37	70000
5	17.8849	446967.5089	179845.3761	39.3762	1000	58	100000
6	17.8849	446967.5251	179845.3873	5.4673	700	20	70000
7	17.8849	446967.4977	179845.3759	11.4670	1000	26	100000
8	17.8849	446967.5080	179845.3874	17.8500	700	18	70000
9	17.8849	446967.5210	179845.3825	6.8512	856	20	85600
10	17.8849	446967.5104	179845.3894	5.4480	700	20	70000
11	17.8849	446967.5057	179845.3770	35.6761	983	55	98300
12	17.8849	446967.5036	179845.3764	38.2098	1000	57	100000
13	17.8849	446967.4864	179845.3696	7.1871	1000	19	100000
14	17.8849	446967.5182	179845.3793	11.3154	1000	26	100000
15	17.8849	446967.5110	179845.3752	28.1802	700	59	70000
16	17.8849	446967.5138	179845.3800	99.9512	1000	100	100000
17	17.8849	446967.5593	179845.3855	16.5342	876	36	87600
18	17.8849	446967.5286	179845.3780	27.1227	1000	46	100000
19	17.8849	446967.5190	179845.3755	8.2677	1000	21	100000
20	17.8849	446967.5027	179845.3721	5.4758	700	20	70000
21	17.8849	446967.4913	179845.3769	7.8470	1000	20	100000
22	17.8849	446967.5320	179845.3843	5.7775	700	21	70000
23	17.8849	446967.4972	179845.3779	5.0343	700	19	70000
24	17.8849	446967.4928	179845.3691	7.1951	1000	19	100000
25	17.8849	446967.5162	179845.3848	5.4591	700	20	70000
26	17.8849	446967.4817	179845.3711	5.4330	700	20	70000
27	17.8849	446967.5156	179845.3781	29.7657	1000	49	100000
28	17.8849	446967.5118	179845.3763	43.6045	898	65	89800
29	17.8849	446967.5176	179845.3795	16.4321	700	42	70000
30	17.8849	446967.5428	179845.3875	7.0305	884	20	88400



(a) 23th run of the ABSA



(b) 16th run of the ABSA



(c) 8th run of the ABSA

Figure 4.9: Convergence performances toward optimum fitness function of optimising the cost of shipping refined oil problem

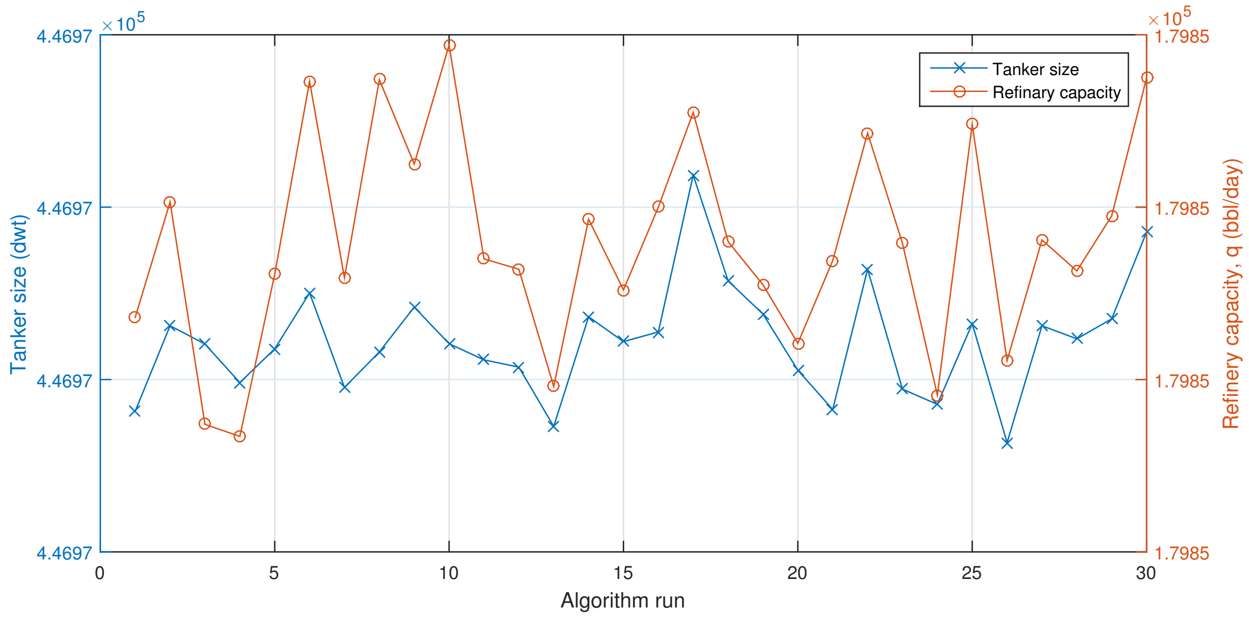


Figure 4.10: Tanker size and refinery capacity obtained in 30 independent runs of the ABSA to optimise the cost of shipping refined oil problem

4.3.2 Profit optimisation of selling television sets

The problem is to estimate the maximum yearly profit (F) in \$/year will be gained by the manufacturer of colour television (TV) sets when two types of TV sets are sold. There are two variables for this problem that are a number of 19" flat screen TV sets sell per year (x_1) and a number of 22" flat screen TV sets sell per year (x_2).

The problem has to consider the information such as:

- A manufacturer's suggested retail price (MSRP) of a 19" flat screen TV and a 21" flat screen TV are \$339 and \$399 respectively.
- A company cost to produce a 19" flat screen TV and a 21" flat screen TV are \$195 and \$225 respectively.
- A fixed cost of \$400000.
- An estimation that for each type of TV set, the average selling price drops by \$0.01 for each additional unit sold.
- An estimation that average selling price of the 19" flat screen TV will be reduced by an additional \$0.003 for each 21" flat screen TV and the price of the 21" flat screen TV will be reduced by an additional \$0.004 for each 19" flat screen TV sold.

The problem is formulated as:

$$\text{Maximise } F(x) = R(x) - C(x)$$

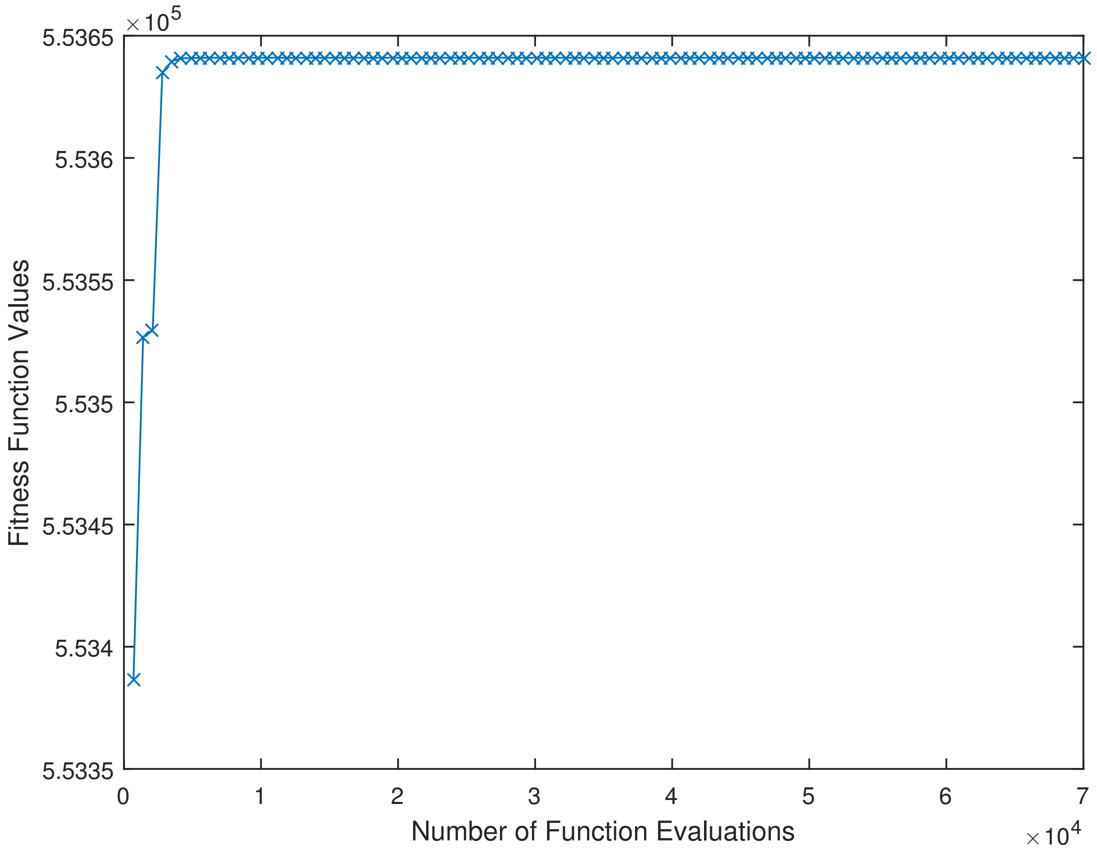
where

$$\begin{aligned}
&C(x) = 400000 + 195(x_1) + 225(x_2) \\
&R(x) = p(x)(x_1) + q(x)(x_2) \\
&p(x) = 339 - 0.01(x_1) - 0.003(x_2) \\
&q(x) = 399 - 0.004(x_1) - 0.01(x_2) \\
&p = \text{selling price for one 19" flat screen TV}, \$ \\
&q = \text{selling price for one 21" flat screen TV}, \$ \\
&C = \text{cost of manufacturing flat screen TV sets}, \$/year \\
&R = \text{revenue from sale of flat screen TV sets}, \$/year \\
&x_1 \geq 0 \text{ and } x_2 \geq 0
\end{aligned} \tag{4.13}$$

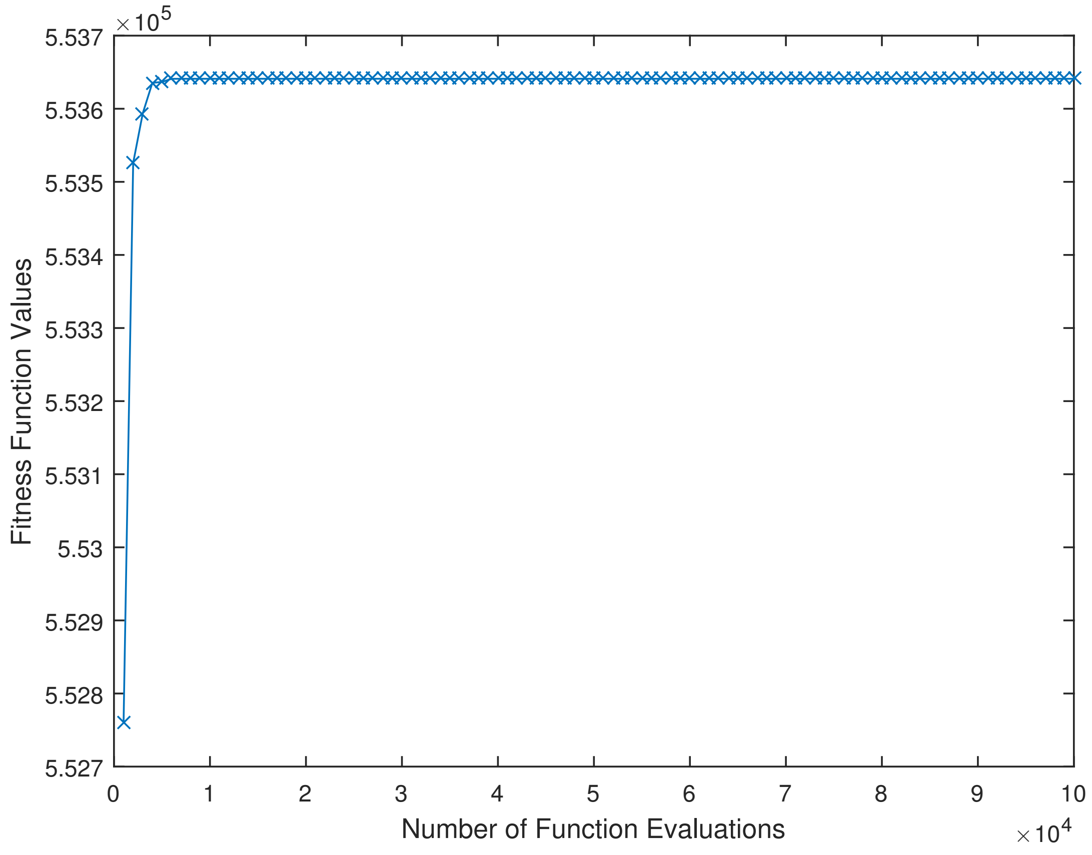
The ABSA is adopted to find the optimum profit for this problem. The ABSA is capable of estimating the maximum yearly profit (F) in \$/year will be gained by a manufacturer of colour TV sets. Table 4.14 shows the results of 30 independent runs by the ABSA to solve this problem. All 30 independent runs of ABSA achieved a similar maximum profit of \$553641.0256 by selling 4735 sets of 19" flat screen TV and 7043 sets of 21" flat screen. This mean that the *best*, *worst* or *mean* maximum profits are equal as well as *standard deviation* is zero.

Table 4.14: Result for 30 runs of ABSA to optimise the profit of selling television sets problem

Run no.	Best fitness, F ($\$/year$)	Variables 19" TV sets, x_1 ($unit\ sold\ /\ year$)	21" TV sets, x_2 ($unit\ sold\ /\ year$)	Time to finish ($seconds$)	Numbers of bats used	Iteration to converge	Number of function evaluation (NFEs)
1	553641.0256	4735	7043	5.8904	700	35	70000
2	553641.0256	4735	7043	3.8427	1000	20	100000
3	553641.0256	4735	7043	10.8029	854	44	85400
4	553641.0256	4735	7043	44.5163	1000	96	100000
5	553641.0256	4735	7043	15.5226	700	65	70000
6	553641.0256	4735	7043	6.7420	1000	30	100000
7	553641.0256	4735	7043	5.1212	1000	25	100000
8	553641.0256	4735	7043	4.5473	1000	23	100000
9	553641.0256	4735	7043	6.7754	1000	30	100000
10	553641.0256	4735	7043	3.8550	700	26	70000
11	553641.0256	4735	7043	4.3521	1000	22	100000
12	553641.0256	4735	7043	2.1589	700	17	70000
13	553641.0256	4735	7043	13.8235	1000	49	100000
14	553641.0256	4735	7043	5.3774	700	33	70000
15	553641.0256	4735	7043	22.2577	837	70	83700
16	553641.0256	4735	7043	20.6073	1000	63	100000
17	553641.0256	4735	7043	7.0210	1000	31	100000
18	553641.0256	4735	7043	6.3223	1000	29	100000
19	553641.0256	4735	7043	3.8573	700	26	70000
20	553641.0256	4735	7043	3.3606	762	21	76200
21	553641.0256	4735	7043	17.3687	1000	56	100000
22	553641.0256	4735	7043	28.7085	700	95	100000
23	553641.0256	4735	7043	4.0104	700	27	70000
24	553641.0256	4735	7043	3.478418	700	24	70000
25	553641.0256	4735	7043	39.5317	878	97	87800
26	553641.0256	4735	7043	12.7783	819	50	81900
27	553641.0256	4735	7043	6.1315	1000	28	100000
28	553641.0256	4735	7043	11.0377	1000	42	100000
29	553641.0256	4735	7043	7.7467	700	42	70000
30	553641.0256	4735	7043	29.7764	700	97	70000



(a) 12th run of the ABSA



(b) 4th run of the ABSA

Figure 4.11: Convergence performances toward optimum fitness function of optimising the profit of selling television sets problem

In term of time for the algorithm to finish, the *mean* time taken by all 30 independent runs of ABSA to solve this problem is 11.910748 *seconds*. From 30 independent runs, 12th run recorded the fastest time, 2.158887 *seconds* and 4th run recorded the slowest time, 44.516322 *seconds* where the results are shown in Figure 4.11a and Figure 4.11b respectively. In addition, the 12th also ran the fastest it started to converge to the optimum value during 17th iteration out of 100 total iterations. 25th and 30th runs recorded the slowest and they started to converge to the optimum value where both only began during 97th iteration respectively.

To solve this problem, the ABSA randomly used 70000 to 100000 number of function evaluations (NFEs). As shown in Figure 4.12, the considered range of NFEs did not much affect the time for the algorithm to finish for all 30 independent runs. Except for 4th, 22th, 25th and 30th runs, other independent runs of ABSA consistently recorded time below 25 *seconds*.

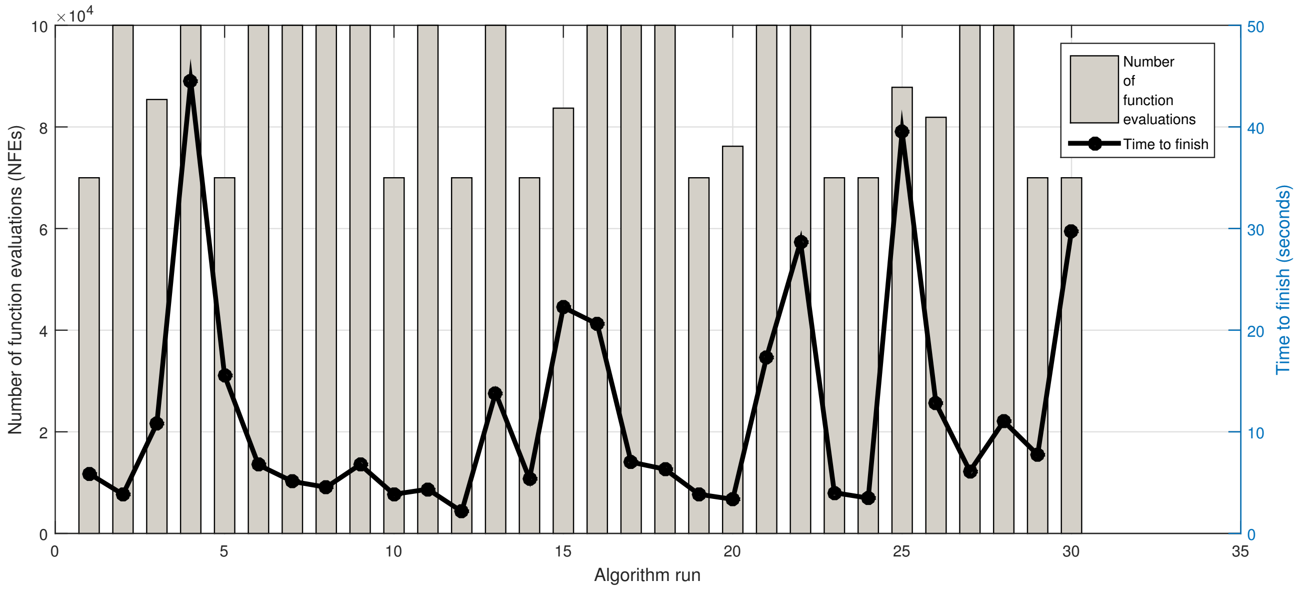


Figure 4.12: Number of function evaluations and time to finish recorded in 30 independent runs of the ABSA to optimise the profit of selling television sets problem

Chapter 5

Development of modified adaptive bats sonar algorithm

5.1 Modified adaptive bats sonar algorithm

ABSA was explored as an improved version of original BSA to solve unconstrained single objective optimisation problems. But, to deal with constrained single objective optimisation problems, a crucial problem on how to incorporate the inequality constraints as well as equality constraints with the objective function must be tackled appropriately. ABSA does not well function on this kind of problem such that an algorithm is a direct approach. A direct approach is often difficult to find the solution in the feasible regions enclosed by the constraints.

A new algorithm named; the modified adaptive bats sonar algorithm (MABSA) is researched here by redefining some elements in ABSA as well as reformulating a main component of BSA to compensate this problem. The MABSA will be able to generate a potential solution that satisfied all constraints. The purpose of MABSA is to solve constrained optimisation problems.

The MABSA is formulated after modifying three searching procedures of the original ABSA and adding a new component to it. The three procedures are the ways to setting up the *beam length* (*L*), determining *starting angle* (θ_m) and *angle between beams* (θ_i) and also calculating *end point position* (pos_i). On the other hand, the bounce back strategy is a new component that has been included in the MABSA, which was not considered in ABSA formerly. This section will elaborate solely of these three elements. The other components of MABSA will not be further discussed here as they are similar to the ABSA as presented in the earlier chapter.

In the MABSA, the new *L* is set up as:

$$L = Rand \times (\frac{SS_{size}}{10\% \times Bats}) \tag{5.1}$$

where the *solution range* (SS_{Size}) is the value between the *upper search space* (SS_{Max}) limit and the *lower search space* (SS_{Min}) limit. Every *dimension* (*Dim*) has its specific or known as *Dim* constraints. The solution range is divided into micron scale, such as 10% of the overall population of bats in the search space. The percentage is marked as possible search space size of each bat to emit sound without colliding with one another. The random value of *L* is offered to make real variation of beam lengths of each *number of beams* (*NBeam*) at every *Dim* (but stay within the *Dim* constraints) at every iteration. This fixation pushes every bat at each dimension to search for larger perimeter each time with the opportunity to diversify the search tactic during iterations and thus may find the global best solution that may be near to them.

Each *NBeam* with *L* is emitted from specific angle location. In the ABSA, the θ_m and θ_i determined random ones in every iteration. So all bats will emit the *NBeam* from a set of similar angle location in each iteration. To add another randomisation character inside MABSA, θ_m and θ_i will be determined in random and separately for every bat at every iteration. So at each iteration, every bat will emit the *NBeam* from a different set of angle location. Therefore, this randomisation will also add on to diversify the searching process in MABSA.

In the MABSA, the way to calculate the pos_i was redefined. The pos_i for each transmitted beam in MABSA is calculated as:

$$\begin{aligned} &pos_i = \alpha \times pos_{SP} + \beta \times L\left(\cos\left[\theta_m + (i-1)\,\theta\right]\right)^{\omega} \\ &\text{where} \\ &i = 1,\ldots,NBeam;\quad NBeam \text{ is } \textit{number of beams} \\ &pos_{SP} \text{ is beam's } \textit{starting position} \end{aligned} \tag{5.2}$$

In the above equation, there are two random variables and one constant. The first random variable is called *position adaptability factor* (α). The value for α is chosen randomly from the range between 0 and 1. This factor is included to make sure that every bat is able to adapt to the new pos_{SP} faster as derived from the previous pos_{SP}, pos_{LB}, pos_{RB} and pos_{GB}. This factor has the same characteristic as random walk method. The second random variable is *collision avoidance factor* (β). The value for β also is chosen randomly from the range between 0 and 1. The factor is essential to avoid the beams from overlapping or incidentally colliding with other bats' beam as every bat has produced a number of beams from new pos_{SP} simultaneously.

The only constant in this equation is *beam-tuning constant* (ω) which is equal to 2. This constant also can be considered as acceleration constant. The function of this constant is to strengthen β so that ω will divert the angle of transmitted beam to the new angle in the designated search space. The value 2 is selected because it will give a good balance. If a very high value is selected, it will destroy the influence of the beam angle such that the orientation of new bat position will be catastrophic. A smaller value, on the other hand, will not make any significant change to the angle of transmitted beam.

The MABSA is also equipped with bounce back strategy. This will confirm that every pos_i achieved by each bat during the iterations is worth considering as possible optimum pos_{GB} for the algorithm. When each beam is transmitted from every bat, it will be verified to ensure that the pos_i of the transmitted beam does not fall beyond SS_{Max} or below SS_{Min}. If the pos_i reaches outside SS_{Size}, the transmitted beam will be diverted automatically to new location inside the labelled SS_{Size} using one of the following equations:

$$pos_i = SS_{Max} - \tau, \quad i = 1,\ldots,N \tag{5.3a}$$

$$pos_i = SS_{Max} + \tau, \quad i = 1,\ldots,N \tag{5.3b}$$

These equations contain *bounce back repositioning factor* (τ) where the value is $0 < \tau < 1$. This factor is to help the

Algorithm 4 Modified adaptive bats sonar algorithm

```
Objective function $F(x)$, $x = (x_1, \ldots, x_d)^T$
Initialise: $Bats$, $MaxIter$, $Dim$, $SS_{Size}$, $NBeam_{MAX}$ and $NBeam_{MIN}$
for $n \leftarrow 1$ to $Bats$ do
  for $d \leftarrow 1$ to $Dim$ do
    Generate random $pos_{SP}$
    Evaluate $F_{SP}$ value for $F(pos_{SP})$
  end for
end for
Assign the most optimum value as $F_{GB}$ and its position as $pos_{GB}$
while $t \leq MaxIter$ do
  Define $NBeam$ to transmit by using $BNI$ (Equation 4.4 and Equation 4.5)
  for $n \leftarrow 1$ to $Bats$ do
    for $N \leftarrow 1$ to $NBeam$ do
      for $d \leftarrow 1$ to $Dim$ do
        Set $L$ and limit $\mu$ (Equation 5.1 and Equation 4.3)
      end for
    end for
    Generate random $\theta_m$ and $\theta$ (Equation 4.6)
    Transmit $NBeam$ starting from $pos_{SP}$
    for $N \leftarrow 1$ to $NBeam$ do
      for $d \leftarrow 1$ to $Dim$ do
        Determine $pos_i$ for each transmitted beam (Equation 5.2)
        Verify $pos_i$ for each transmitted beam within $SS_{Size}$
        if $pos_i \geq SS_{Max}$ then
          Update $pos_i$ (Equation 5.3a)
        end if
        if $pos_i \leq SS_{Min}$ then
          Update $pos_i$ (Equation 5.3b)
        end if
      end for
      Evaluate $F_i$ value for $F(pos_i)$
      Assign the optimum value of $F_i$ as $F_{LB}$ and its position as $pos_{LB}$
      if $F_{LB} \leq F_{SP}$ then
        Assign $F_{LB}$ as $F_{RB}$ and $pos_{LB}$ as $pos_{RB}$
      else
        Assign $F_{SP}$ as $F_{RB}$ and $pos_{SP}$ as $pos_{RB}$
      end if
    end for
  end for
  Select the optimum value among $F_{RB}$ as current $F_{GB}$ and its $pos_{RB}$ as current $pos_{GB}$
  if current $F_{GB} \leq$ previous $F_{GB}$ then
    Update current $F_{GB}$ as new $F_{GB}$ and current $pos_{GB}$ as new $pos_{GB}$
  else
    Retain previous $F_{GB}$ and $pos_{GB}$
  end if
  for $n \leftarrow 1$ to $Bats$ do
    Determine new $pos_{SP}$ using (Equation 4.8)
    Evaluate new $F_{SP}$ value for $F(pos_{SP})$
  end for
end while
Declare $F_{GB}$ as optimum fitness evaluated and $pos_{GB}$ as its optimum value(s)
```

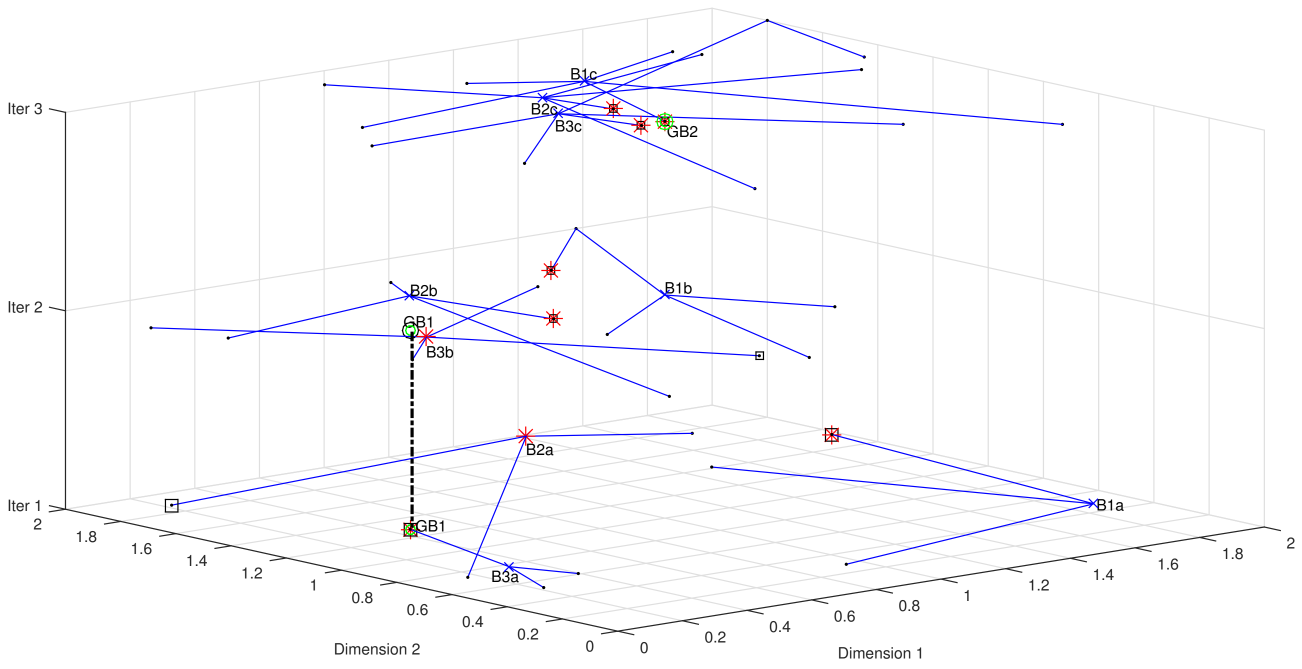
bats to relocate a beam transmission to a new beams' end point from the maximum or minimum search space. This factor will avoid overwriting other bats' beam end points. The *bounce back repositioning factor* is the fastest contingency action of bats to swing to newly transmitted beam's end point after hitting the designated search space boundaries. This strategy helps to reduce much time to spend to consider the previous factors (which are: *position adaptability factor*, *collision avoidance factor* or *beam-tuning constant*) as normal bats do. Algorithm 4 represented the pseudo code of MABSA. In the pseudo code, the new equations formulated from this chapter are referred as well as unchanged equations from the previous chapter remain.

In the meantime, Figure 5.1 shown the orthogonal and plan view of a sample on how the bats in MABSA move to search for the F_{GB}. This sample search is for 2-dimensional optimum points. The ranges of the solution search space are taken as $0 \leq Dim1, Dim2 \leq 2$.

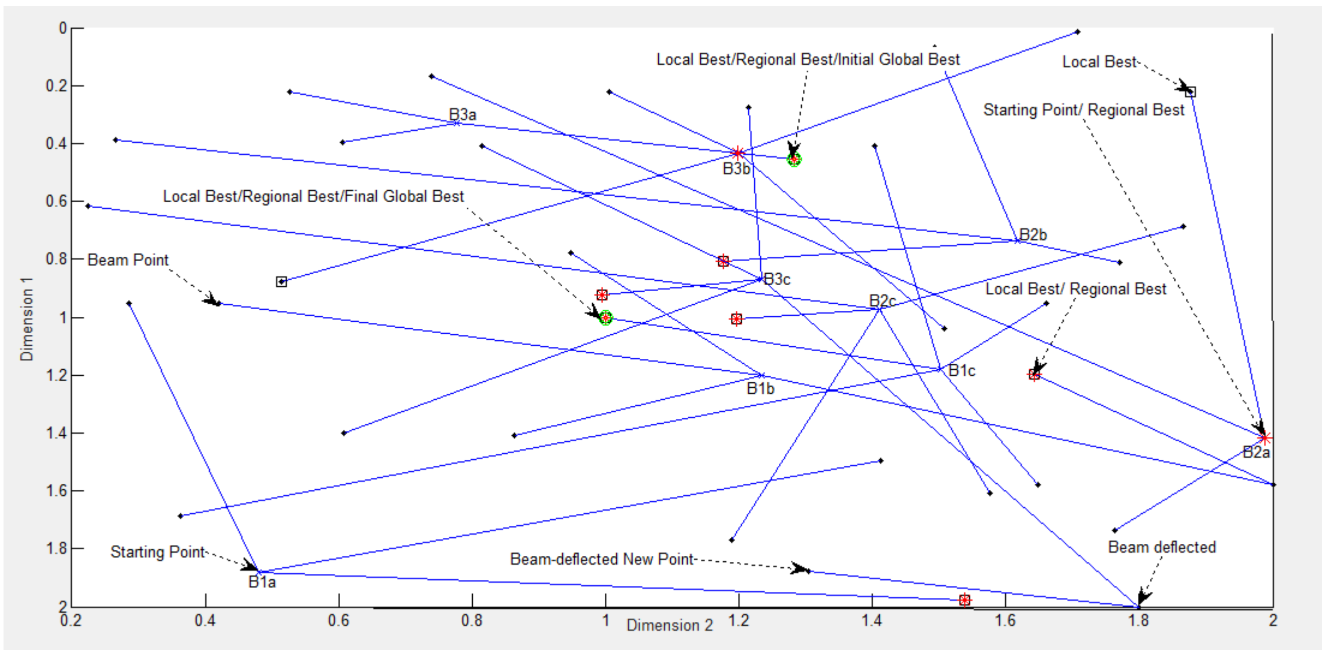
During the first iteration, three $Bats$ are introduced at random pos_{SP} (are evaluate to produce three F_{SP}) and are labelled as $B1a$, $B2a$ and $B3a$ respectively. Each bat transmits three $NBeam$ (Equation 4.4 and Equation 4.5) in different lengths (Equation 5.1 and Equation 4.3) to various directions (Equation 4.6). Then, every pos_i at each bat is evaluated (Equation 5.2). At each bat, the F_i from every pos_i are compared among them and the fittest ones are recognized as F_{LB}. Later, the F_{LB} will be compared with its F_{SP} and the best between the two will be F_{RB}. This means that, there are three F_{RB} all together and the best of them is declared as F_{GB}. After that, new pos_{SP} for the bats are identified (Equation 4.8) and tagged as $B1b$, $B2b$ and $B3b$ respectively.

The second iteration starts from $B1b$, $B2b$ and $B3b$ locations and similar processes are repeated as in the first iteration. In this iteration, the $NBeam$ is increased to four. If the transmitted beam goes beyond the search space, it will be deflected back to new direction within the solution range area (Equation 5.3a or Equation 5.3b). However, the F_{RB} in this iteration will be less than the F_{GB} value in the first iteration. Due to that, the F_{GB} value at this iteration will still be carried from the previous iteration.

In the last iteration, the processes are still continued the same as in the previous iterations but $NBeam$ is increased to five transmitted from $B1c$, $B2c$ and $B3c$ respectively. The final F_{GB} value was detected at the position pos_{GB}; $Dim1$=1 and $Dim2$=1 which were the source initially from $B1c$.



(a) Orthogonal view



(b) Plan view

Figure 5.1: Bats movement in MABSA approach

5.2 Computer simulation and discussion

5.2.1 Performance of modified adaptive bats sonar algorithm on constrained optimisation benchmark test functions

In order to show the superiority of the MABSA to solve constrained optimisation problems, four constrained benchmark test functions from **2006 IEEE Congress on Evolutionary Computation** *CEC 2006* were examined and tested. The results are compared against other established algorithms based on results recorded in the specific literature (no re-simulation exercises using the established algorithms were conducted).

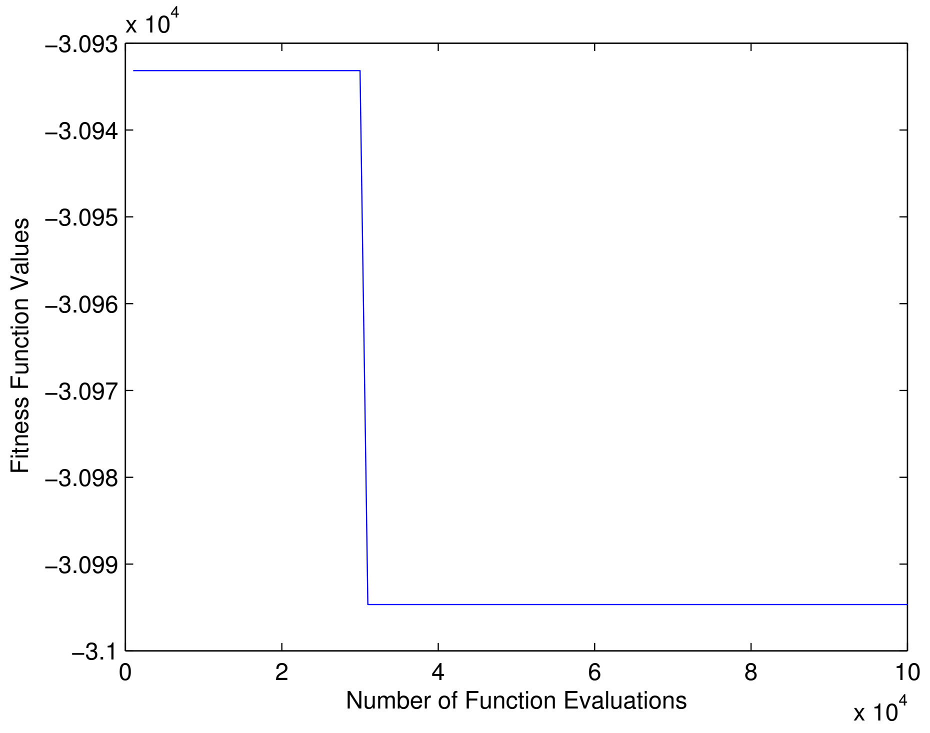
The algorithms are; changing range genetic algorithm (CRGA), self adaptive penalty function (SAPF), cultured differential evolution (CULDE), simple multimembered evolution strategy (SMES), adaptive segregational constraint handling evolutionary algorithm (ASCHEA), particle swarm optimisation with diferential evolution (PSO-DE), stochastic ranking (SR), differential evolution with level comparison (DELC), differential evolution with dynamic stochastic selection (DEDS), hybrid evolutionary algorithm and adaptive constraint handling technique (HEA-ACT), improved stochastic ranking (ISR), α constrained with nonlinear simplex method with mutation (α Simplex), Nelder-Mead simplex method and particle swarm optimisation (NM-PSO), artificial bee colony 2 (ABC2) and mine blast algorithm (MBA). All established algorithms from specific literature provided the results for all constrained optimisation benchmark test functions but NM-PSO algorithm which has the results for constrained test function 1 only.

The quality of obtained optimisation results are compared in terms of statistical results (better *best*, *mean*, *median* and *worst* solution found), the robustness of the MABSA (the *standard deviation* values) and the *number of function evaluations* (NFEs). However, there are few cases where the results for *median* and *worst* solutions found as well as the *standard deviation* values are not available in certain established algorithms from the specific literature. Here, the notation "n/a" means not available are given.

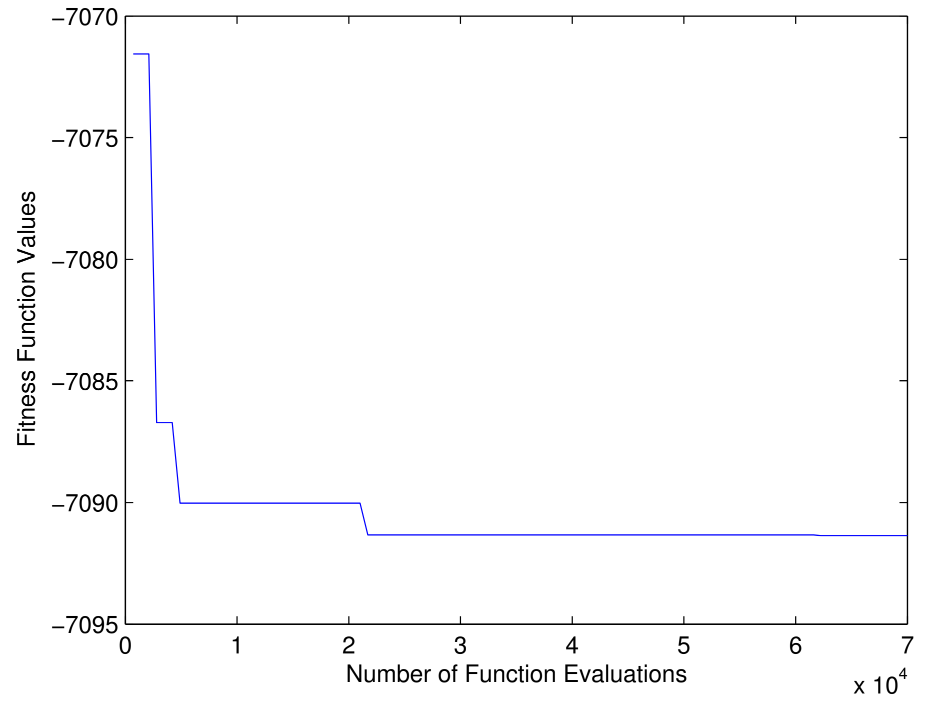
The results of the *best* solution obtained from MABSA for constrained optimisation benchmark test functions are summarised in Table 5.1. The MABSA is capable of finding the *best* solution (minimum value) which was better than the optimum value as suggested from *CEC 2006* for all constrained test functions. The time to converge to the *best* solution was recorded under 22 *seconds* for all four test functions shows that the algorithm is able to reach to the *best* solution faster than ordinary methods. So it is worth mentioning that MABSA is very effective and efficient to solve the constrained optimisation problems.

Table 5.1: Results of the *best* solution obtained from MABSA for constrained benchmark test functions

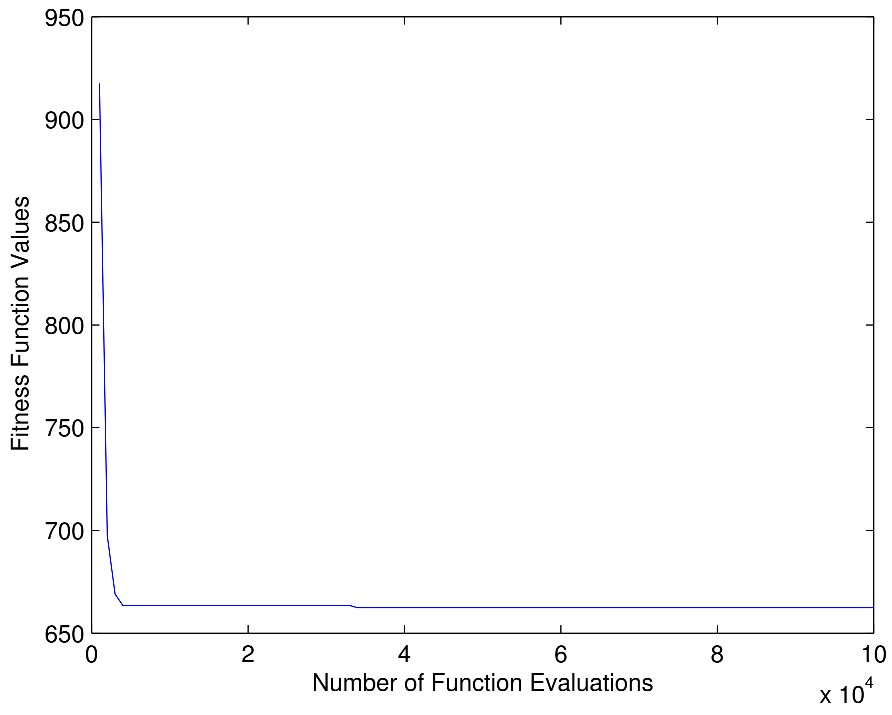
Items	Constrained test function 1	Constrained test function 2	Constrained test function 3	Constrained test function 4
Run No.	23	2	21	5
No. of $Bats$	1000	700	1000	700
NFEs	100000	70000	100000	70000
Time to converge ($seconds$)	9.7244	20.9769	14.2320	0.3656
Iteration to converge	31	89	34	3
$F(x)$	-30994.6595	-7091.3568	662.4557	0.7500
Optimum value of $F(x)$	-30665.5390	-6961.8139	680.6301	0.7500



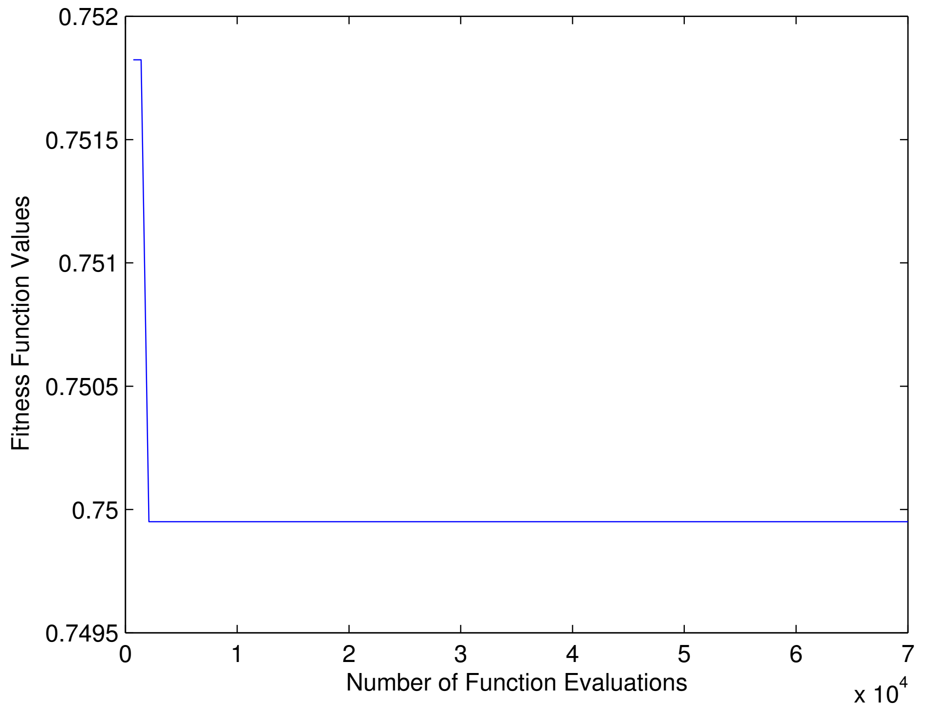
(a) Constrained problem 1



(b) Constrained problem 2



(c) Constrained problem 3



(d) Constrained problem 4

Figure 5.2: Convergence graphs of the best solution of MABSA for four constrained benchmark problems

MABSA also performed well to converge faster to the optimum solution. In all four constrained benchmark test functions, MABSA had reached the optimum solutions in less than 25 *seconds*. In term of NFEs, MABSA had shown good potential to be popular algorithm in future as it converges fast to the optimum solution. For instance, by considering the NFEs from the *best* solution obtained in all constrained benchmark test functions tested, MABSA started to settle down to the optimum solution after approximately 2000 to 4000 NFEs as shown in Figure 5.2.

Figure 5.3 compares the average NFEs used by all algorithms to solve four constrained benchmark test functions. When comparing the average NFEs used by MABSA on all constrained benchmark test functions with other established algorithms, the value between 70000 and 100000 is reasonable and more productive. The small value of NFEs will force the algorithm to settle down earlier as possible without a chance to explore more but may end up with the algorithm trapped in local optimum such as in CRGA, NM-PSO or DELC. On the other hand, if too many NFEs used such as in ASCHEA or even SAPF, the algorithm may waste the time to find the good solution but the solution which was already encountered earlier than the last set of NFEs is examined.

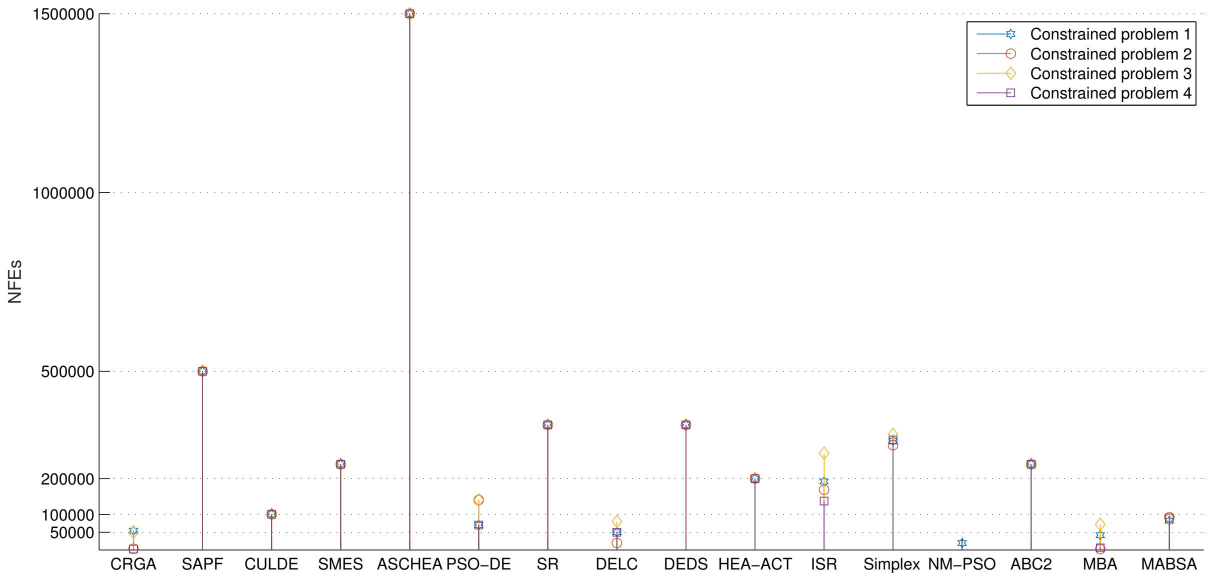


Figure 5.3: Comparison of NFEs used by considered algorithms for all constrained benchmark problems

Constrained optimisation benchmark test function 1

The constrained test function 1 is defined as:

$$\text{Minimise } F(x) = 5.3578547x_3^3 + 0.8356891x_1x_5 + 37.293239x_1 + 40729.141$$

subject to

$$\begin{aligned}
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0\\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 0.0022053x_3x_5 \leq 0\\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0\\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0\\
g_5(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0\\
g_6(x) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned} \tag{5.4}$$

where

$$78.0 \leq x_1 \leq 102.0$$

$$33.0 \leq x_2 \leq 45.0$$

$$27.0 \leq x_i \leq 45.0, \quad i = 3,4,5$$

Table 5.2: Comparison of statistical results obtained using different algorithms for constrained test function 1. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CRGA	-30660.3130	-30665.2520	-30664.3980	-30665.5200	1.6000	54400
SAPF	-30656.4710	-30663.9210	-30659.2210	-30665.4010	2.0430	500000
CULDE	-30665.5387	n/a	-30665.5387	-30665.5387	0.0000	100100
SMES	-30665.5390	-30665.5390	-30665.5390	-30665.5390	0.0000	240000
ASCHEA	n/a	-30665.5000	-30665.5000	-30665.5000	n/a	1500000
PSO-DE	-30665.5387	-30665.5387	-30665.5387	-30665.5387	$8.3000e^{-10}$	70100
SR	-30665.5390	-30665.5390	-30665.5390	-30665.5390	$2.0000e^{-05}$	350000
DELC	-30665.5390	-30665.5390	-30665.5390	-30665.5390	$1.0000e^{-11}$	50000
DEDS	-30665.5390	n/a	-30665.5390	-30665.5390	$2.7000e^{-11}$	350000
HEA-ACT	-30665.5390	-30665.5390	-30665.5390	-30665.5390	$7.4000e^{-12}$	200000
ISR	-30665.5390	-30665.5390	-30665.5390	-30665.5390	$1.1000e^{-11}$	192000
α Simplex	-30665.5387	-30665.5387	-30665.5387	-30665.5387	$4.2000e^{-11}$	305343
NM-PSO	-30665.5390	n/a	-30665.5390	-30665.5390	$1.4000e^{-05}$	19658
ABC2	-30665.5390	n/a	-30665.5390	-30665.5390	0.0000	240000
MBA	-30665.3300	n/a	-30665.5182	-30665.5386	$5.0800e^{-02}$	41750
MABSA	-30700.2654	-30793.4331	-30829.8768	-30994.6595	110.3421	82090

For constrained test function 1, there are 15 different algorithms from literature that have been chosen to compare with the MABSA. These included CRGA, SAPF, CULDE, SMES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α Simplex, NM-PSO, ABC2 and MBA. Table 5.2 shows the comparison between MABSA and other algorithms in term of statistical results obtained for solving constrained test function 1.

Overall, MABSA lead other algorithms to all criteria (*worst*, *median*, *mean* and *best* value) which demonstrate the quality of algorithm to achieve the optimum solution for constrained test function 1. This statement was strengthened by the bar plot pictured in Figure 5.4 where MABSA was significantly better to achieve the optimum solution as compared to optimum value compiled in *CEC 2006* or other algorithms. Indeed, the *worst* result from the MABSA; -30700.2654 is still a better result than the optimum value or the *best* result from other established algorithms. However, MABSA is less robust to solve the problem as shown by the higher value of *standard deviation* when compared to other listed algorithms.

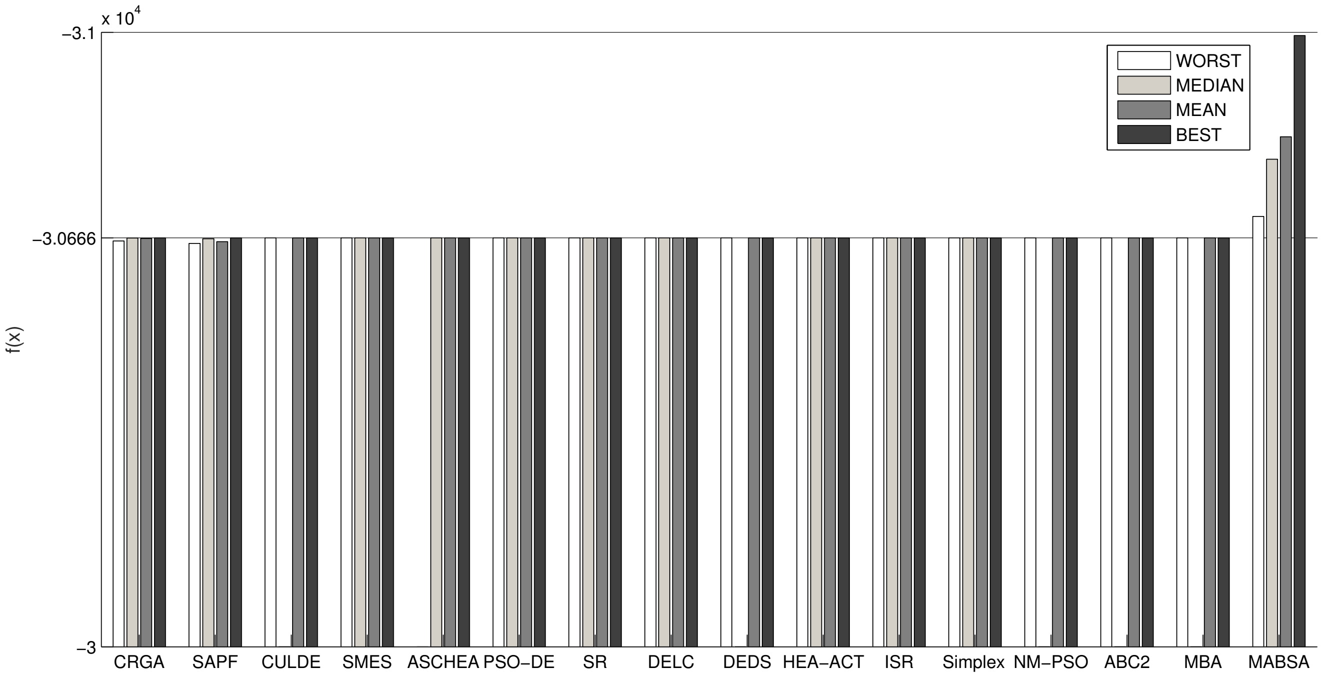


Figure 5.4: Bar plot of statistical results obtained using different algorithms for constrained test function 1

Contrained optimisation benchmark test function 2

The constrained test function 2 is defined as:

$$\begin{aligned}
&\text{Minimise } F(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \\
&\text{subject to} \\
&\qquad g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
&\qquad g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.85 \leq 0 \\
&\text{where} \\
&\qquad 13.0 \leq x_1 \leq 100.0 \\
&\qquad 0.0 \leq x_2 \leq 100.0, \quad i = 3,4,5
\end{aligned} \tag{5.5}$$

In constrained test function 2, the performance of MABSA was also compared with the 14 established algorithms. The algorithms are CRGA, SAPF, CULDE, SMES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α Simplex, ABC2 and MBA. The statistical results obtained by all algorithms including MABSA are shown in Table 5.3 while the *worst*, *median*, *mean* and *best* results for each considered algorithms plot are shown on bar plot as in Figure 5.5.

The outstanding performance of the MABSA to solve the constrained test function 2 can be seen in both table and bar plot. The fitness function value achieved by the MABSA for every statistical criterion was the optimum as compared to other 14 established algorithms as well as the optimum value from *CEC 2006*. In addition to that, the MABSA method was the only algorithm passing the -7000.0000 value in *median*, *mean* and *best* which was not able to be done by other algorithms. Nevertheless, the higher *standard deviation* value achieved by MABSA shows that the algorithm was less robust to solve the constrained test function 2 compared to other algorithms. However, the level of robustness for MABSA to solve this problem was better than the previous problem.

Table 5.3: Comparison of statistical results obtained using different algorithms for constrained test function 2. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CRGA	-6077.1230	-6867.4610	-6740.2880	-6956.2510	270.0000	3700
SAPF	-6943.3040	-6953.8230	-6953.0610	-6961.0460	5.8760	500000
CULDE	-6961.8139	n/a	-6961.8139	-6961.8139	0.0000	100100
SMES	-6952.4820	-6961.8140	-6961.2840	-6961.8140	1.8500	240000
ASCHEA	n/a	-6961.8100	-6961.8100	-6961.8100	n/a	1500000
PSO-DE	-6961.8139	-6961.8139	-6961.8139	-6961.8139	$2.3000e^{-09}$	140100
SR	-6350.2620	-6961.8140	-6875.9400	-6961.8140	160.0000	350000
DELC	-6961.8140	-6961.8140	-6961.8140	-6961.8140	$7.3000e^{-10}$	20000
DEDS	-6961.8140	n/a	-6961.8140	-6961.8140	0.0000	350000
HEA-ACT	-6961.8140	-6961.8140	-6961.8140	-6961.8140	$4.6000e^{-12}$	200000
ISR	-6961.8140	-6961.8140	-6961.8140	-6961.8140	$1.9000e^{-12}$	168800
α Simplex	-6961.8139	-6961.8139	-6961.8139	-6961.8139	$1.3000e^{-10}$	293367
ABC2	-6961.8050	n/a	-6961.8130	-6961.8140	$2.0000e^{-03}$	240000
MBA	-6961.8139	n/a	-6961.8139	-6961.8139	0.0000	2835
MABSA	-6973.2374	-7047.2779	-7043.7395	-7091.3568	34.227384	91530

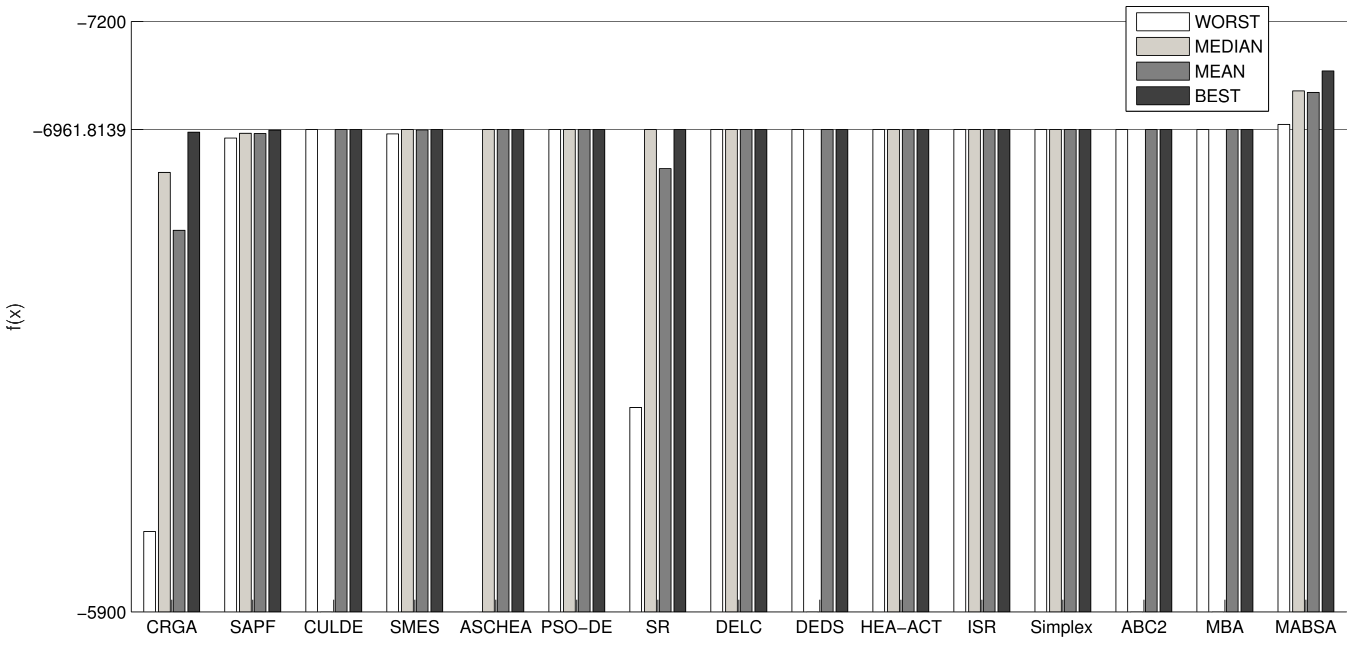


Figure 5.5: Bar plot of statistical results obtained using different algorithms for constrained test function 2

Contrained optimisation benchmark test function 3

The constrained test function 3 is defined as:

$$\text{Minimise } F(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$$

subject to

$$\begin{aligned}
g_1(x) &= 127-2x_1^2-3x_2^4-x_3-4x_4^2-5x_5 \geq 0\\
g_2(x) &= 282-7x_1-3x_2-10x_3^2-x_4+x_5 \geq 0\\
g_3(x) &= 196-23x_1-x_2^2-6x_6^2+8x_7 \geq 0\\
g_4(x) &= -4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7 \geq 0
\end{aligned} \tag{5.6}$$

where

$$-10.0 \leq x_i \leq 10.0, \quad i=1,2,3,4,5,6,7$$

In constrained test function 3, the statistical results between MABSA and 14 other algorithms that are taken from previous literature are compared. The algorithms are CRGA, SAPF, CULDE, MES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α Simplex, ABC2 and MBA. The comparison of statistical results obtained by all algorithms are provided in Table 5.4. Figure 5.6 visualized the bar plot of *worst*, *median*, *mean* and *best* solution of all algorithms with a benchmark of the optimum value from *CEC 2006*.

The performance of MABSA was exceptional when compared to other established algorithms to find the optimum fitness function value for constrained test function 3. The MABSA was the sole algorithm that recorded the minimum solution under 680.0000 for all statistical criterion with the *best* solution 662.4557 which was far better than the optimum value from *CEC 2006*. For this constrained test function 3, MABSA was well thought-out to be more robust when compared to the performances of the constrained test function 1 or constrained test function 2. Despite the fact that the *standard deviation* for MABSA was still larger than 1.0000, the value was acceptable to compromise with the range of *worst*, *median*, *mean* and *best* solution found which was better amongst considered algorithms.

Table 5.4: Comparison of statistical results obtained using different algorithms for constrained test function 3. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CRGA	682.9650	681.2040	681.3470	680.7260	$5.7000e^{-01}$	50000
SAPF	682.0810	681.2350	681.2460	680.7730	$3.2200e^{-01}$	500000
CULDE	680.6301	n/a	680.6301	680.6301	0.0000	100100
SMES	680.7190	680.6420	680.6430	680.6320	$1.5500e^{-02}$	240000
ASCHEA	n/a	680.6350	680.6410	680.6300	n/a	1500000
PSO-DE	680.6301	680.6301	680.6301	680.6301	$4.6000e^{-13}$	140100
SR	680.7630	680.6410	680.6560	680.6300	$3.4000e^{-02}$	350000
DELC	680.6300	680.6300	680.6300	680.6300	$3.2000e^{-12}$	80000
DEDS	680.6300	n/a	680.6300	680.6300	$2.5000e^{-13}$	350000
HEA-ACT	680.6300	680.6300	680.6300	680.6300	$5.8000e^{-13}$	200000
ISR	680.6300	680.6300	680.6300	680.6300	$3.2000e^{-13}$	271200
α Simplex	680.6301	680.6301	680.6301	680.6301	$2.9000e^{-10}$	323427
ABC2	680.6530	n/a	680.6400	680.6340	$4.0000e^{-03}$	240000
MBA	680.7882	n/a	680.6620	680.6322	$3.3000e^{-02}$	71750
MABSA	678.7398	672.6514	671.4536	662.4557	4.6726	88303

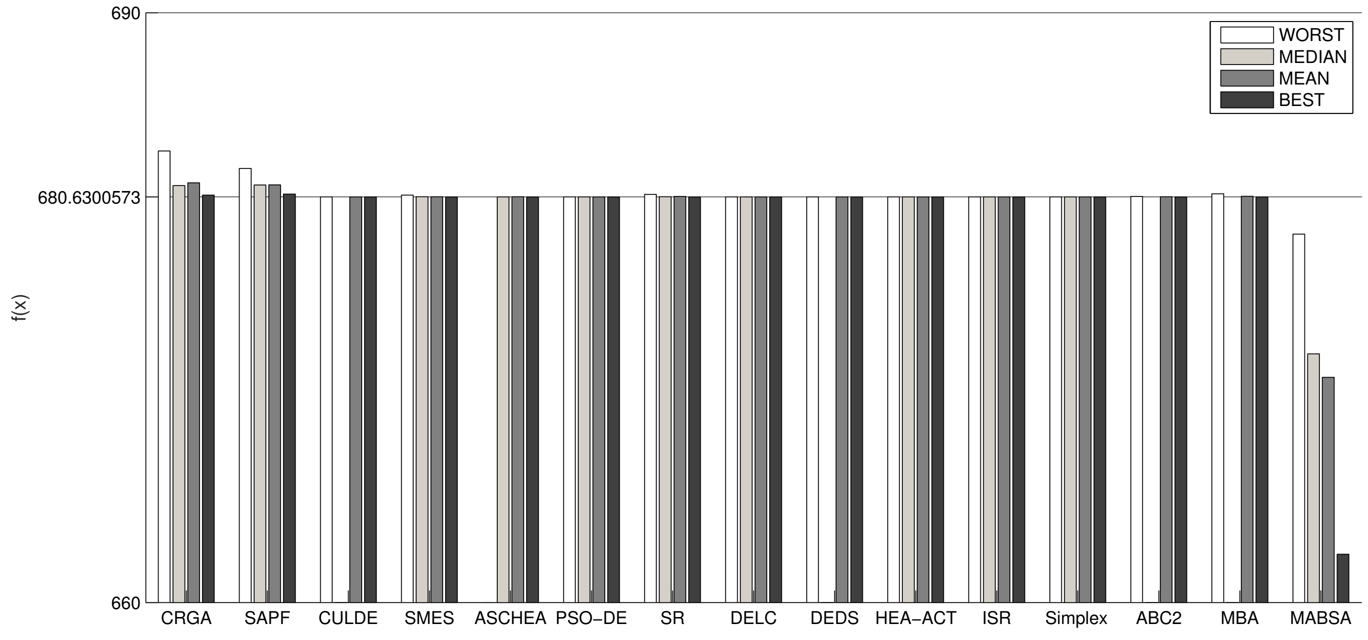


Figure 5.6: Bar plot of statistical results obtained using different algorithms for constrained test function 3

Contrained optimisation benchmark test function 4

The constrained test function 4 is defined as:

$$\text{Minimise } F(x) = x_1^2 + (x_2 - 1)^2$$

subject to

$$h(x) = x_2 - x_1^2 = 0 \tag{5.7}$$

where

$$-1.0 \leq x_i \leq 1.0, \quad i = 1,2$$

A set of 14 established algorithms is compared with MABSA in term of the statistical results obtained for constrained test function 4. These included CRGA, SAPF, CULDE, SMES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α Simplex, ABC2 and MBA. Table 5.5 listed the comparison results, while the bar plot of *worst*, *median*, *mean* and *best* solution acquired from all the algorithms with the optimum value from *CEC 2006* is shown in Figure 5.7.

Table 5.5: Comparison of statistical results obtained using different algorithms for constrained test function 4. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CRGA	0.7570	0.7510	0.7520	0.7500	$2.5000e^{-03}$	3000
SAPF	0.7570	0.7500	0.7510	0.7490	$2.0000e^{-03}$	500000
CULDE	0.7965	n/a	0.7580	0.7499	$1.7138e^{-02}$	100100
SMES	0.7500	0.7500	0.7500	0.7500	$1.5200e^{-04}$	240000
ASCHEA	n/a	0.7500	0.7500	0.7500	n/a	1500000
PSO-DE	0.7500	0.7499	0.7499	0.7499	$2.5000e^{-07}$	70100
SR	0.7500	0.7500	0.7500	0.7500	$8.0000e^{-05}$	350000
DELC	0.7500	0.7500	0.7500	0.7500	0.0000	50000
DEDS	0.7499	n/a	0.7499	0.7499	0.0000	350000
HEA-ACT	0.7500	0.7500	0.7500	0.7500	$3.4000e^{-16}$	200000
ISR	0.7500	0.7500	0.7500	0.7500	$1.1000e^{-16}$	137200
α Simplex	0.7499	0.7499	0.7499	0.7499	$4.9000e^{-16}$	308125
ABC2	0.7500	n/a	0.7500	0.7500	0.0000	240000
MBA	0.7500	n/a	0.7500	0.7500	$3.2900e^{-06}$	6405
MABSA	0.7500	0.7500	0.7500	0.7500	0.0000	89724

For constrained test function 4, MABSA successfully printed out results which have the same performance or better than other considered algorithms for all criteria. Indeed, the *median*, *mean* and *best* solution values achieved by MABSA method managed to achieve better than that *CEC 2006* benchmark value; 0.7500. The MABSA recorded 0.7500, 0.7500 and 0.7500 for *median*, *mean* and *best* criteria respectively. According to the results, MABSA is also considered to be more robust to solve the constrained test function 4 as its *standard deviation* value recorded was 0.000000. The robustness ability of MABSA to solve the problem was at par with other considered algorithms and better than CGRA, SAPF, CULDE and SMES.

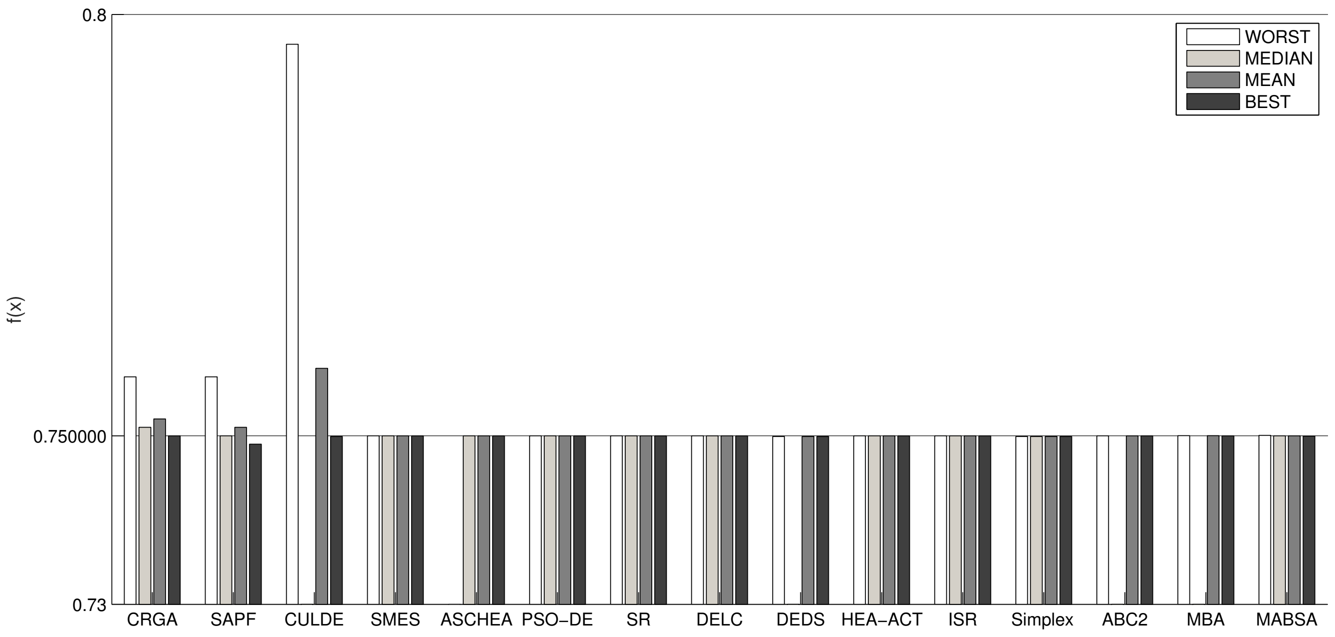


Figure 5.7: Bar plot of statistical results obtained using different algorithms for constrained test function 4

5.2.2 Performance of modified adaptive bats sonar algorithm in engineering design optimisation problems

The MABSA also was tested to solve six engineering design optimisation problems. The problems considered are pressure vessel design optimisation problem, three-truss bar design optimisation problem, gear train design optimisation problem, speed reducer design optimisation problem, welded beam design optimisation problem and tension/compression spring design optimisation problem. All six engineering design optimisation problems are established problems and broadly used in the literature.

The results produced by MABSA to solve all nominated engineering design optimisation problems have been compared against other established algorithms based on results recorded in the specific literature in a way to show the superior performance of the algorithm. Noteworthy to mention, no re-simulation exercises using the established algorithms were conducted. The considered algorithms are; co-evolutionary particle swarm optimisation (CPSO), hybrid particle swarm optimisation (HPSO), teaching-learning-based optimisation (TLBO), society and civilization algorithm (SC), particle swarm optimisation with diferential evolution (PSO-DE), differential evolution with level comparison (DELC), differential evolution with dynamic stochastic selection (DEDS), hybrid evolutionary algorithm and adaptive constraint handling technique (HEA-ACT), artificial bee colony 1 (ABC1), Nelder-Mead simplex method and particle swarm optimisation (NM-PSO), genetic algorithm 1 (GA1), genetic algorithm 2 (GA2), unified particle swarm optimisation (UPSO), μ and λ evolution strategy (($\mu + \lambda$)ES) and mine blast algorithm (MBA). However, not all established algorithms from specific literature provided the results for all nominated engineering design optimisation problems.

In overall, Figure 5.8 compared the average NFEs used by all considered algorithms to solve six engineering design optimisation problems. The NFEs used by the MABSA were in the acceptable range that was between 70000 and 100000. If the small number of NFEs which is less than 50000 is used like in TLBO, DELC or DEDS; the tendency of the premature convergence to the optimum solution to occur is higher. The premature convergence happened because the algorithm has to end the searching process earlier without having much time to explore every corner of the designated

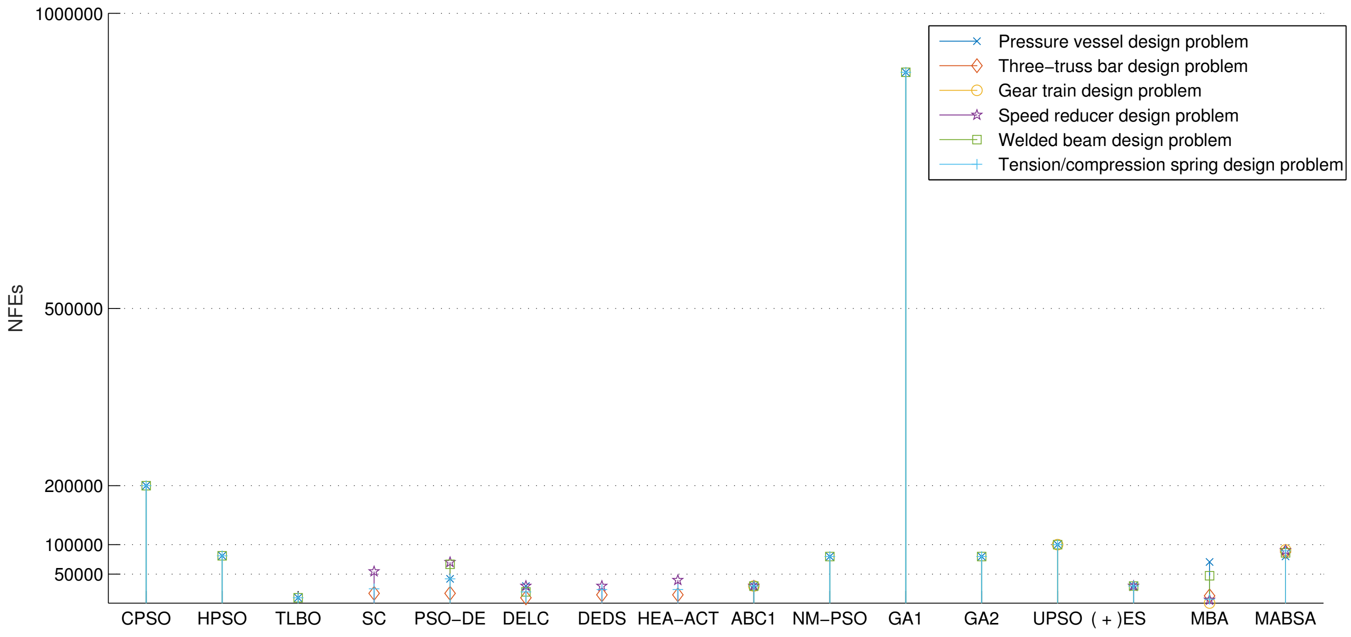


Figure 5.8: Comparison of NFEs used by considered algorithms for all engineering design optimisation problems

search space.

However, too large NFEs setup (200000 and above) as in GA1 or CPSO was an unproductive and computational burden as the algorithm will still search for the solution even though the optimum solution has already appeared in the early searching move. Too large NFEs also may contribute to an inconstant global best solution as the solution keeps changing until the searching process finished.

Pressure vessel design optimisation problem

The pressure vessel design optimisation problem is defined as:

$$\text{Minimise } F(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

subject to

$$\begin{aligned} g_1(x) &= -x_1 + 0.0193x_3 \leq 0 \\ g_2(x) &= -x_2 + 0.00954x_3 \leq 0 \\ g_3(x) &= -\pi x_3^2 x_4 - (4/3)\pi x_3^3 + 1296000 \leq 0 \\ g_4(x) &= x_4 - 240 \leq 0 \end{aligned} \tag{5.8}$$

where

$$0.0 \leq x_i \leq 100.0, \quad i = 1,2$$

$$10.0 \leq x_i \leq 200.0, \quad i = 3,4$$

The *best* solution acquired using MABSA for solving pressure vessel design optimisation problem is tabled in Table 5.6. The MABSA needed only 22 *seconds* to converge to the *best* solution which is 5167.3330. To illustrate the convergence rate of the MABSA, Figure 5.9 showed the convergence to the *best* solution in term of NFEs. The MABSA efficiently reached the *best* solution after 60000 NFEs out of 70000 NFEs.

Table 5.6: Results of the *best* solution obtained from MABSA for pressure vessel optimisation design problem

Items	Value
Run No.	14
No. of *Bats*	700
NFEs	70000
Time to converge (*seconds*)	22.0172
Iteration to converge	83
$F(x)$	5167.3330
Optimum value of $F(x)$	6059.7140

To further investigate the performance of MABSA to solve the pressure vessel design optimisation problem, the algorithm has been compared to 12 established techniques taken from literatures. The algorithms involved are CPSO, HPSO, TLBO, PSO-DE, DELC, ABC1, NM-PSO, GA1, GA2, UPSO, $(\mu + \lambda)$ES and MBA. The comparison was done on statistical results obtained by all algorithms discussed which is exhibited in Table 5.7 and plot on bar plot as in Figure 5.10.

According to the results, MABSA performed the best compared to other algorithms as the optimum solutions found by MABSA were under 6000.0000 for all statistical criteria except for the *worst* value. Indeed, the *worst* solution of MABSA was still better than the *best* solution achieved by GA1 or UPSO. Meanwhile, the MABSA was not so robust to solve the pressure vessel design optimisation problem as interpreted by the large value of *standard deviation* obtained by the algorithm. But, the level of robustness of MABSA is considered better as compared to UPSO alone.

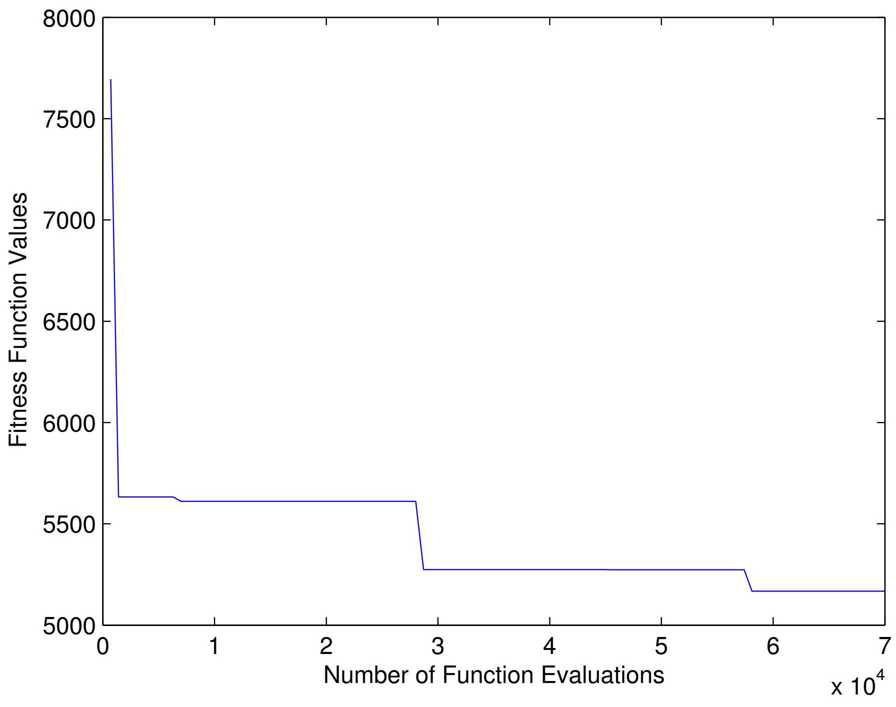


Figure 5.9: Convergence graph of the best solution of MABSA for pressure vessel design optimisation problem

Table 5.7: Comparison of statistical results obtained using different algorithms for pressure vessel design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CPSO	6363.8041	n/a	6147.1332	6061.0777	86.4545	200000
HPSO	6288.6770	n/a	6099.9323	6059.7143	86.2022	81000
TLBO	n/a	n/a	6059.7143	6059.7143	n/a	10000
PSO-DE	6059.7143	n/a	6059.7143	6059.7143	$1.0000e^{-10}$	42100
DELC	6059.7143	6059.7143	6059.7143	6059.7143	$2.1000e^{-11}$	30000
ABC1	n/a	n/a	6245.3081	6059.7147	205.0000	30000
NM-PSO	5960.0557	n/a	5946.7901	5930.3137	9.1614	80000
GA1	6308.1497	6290.0187	6293.8432	6288.7445	7.4133	900000
GA2	6469.3220	n/a	6177.2533	6059.9463	130.9297	80000
UPSO	11638.2000	n/a	9032.5500	6544.2700	995.5730	100000
(μ+λ)ES	6820.3975	n/a	6379.9380	6059.7016	210.0000	30000
MBA	6392.5062	n/a	6200.6477	5889.3216	160.3400	70650
MABSA	6092.8908	5618.6387	5607.7972	5167.3330	252.3335	80227

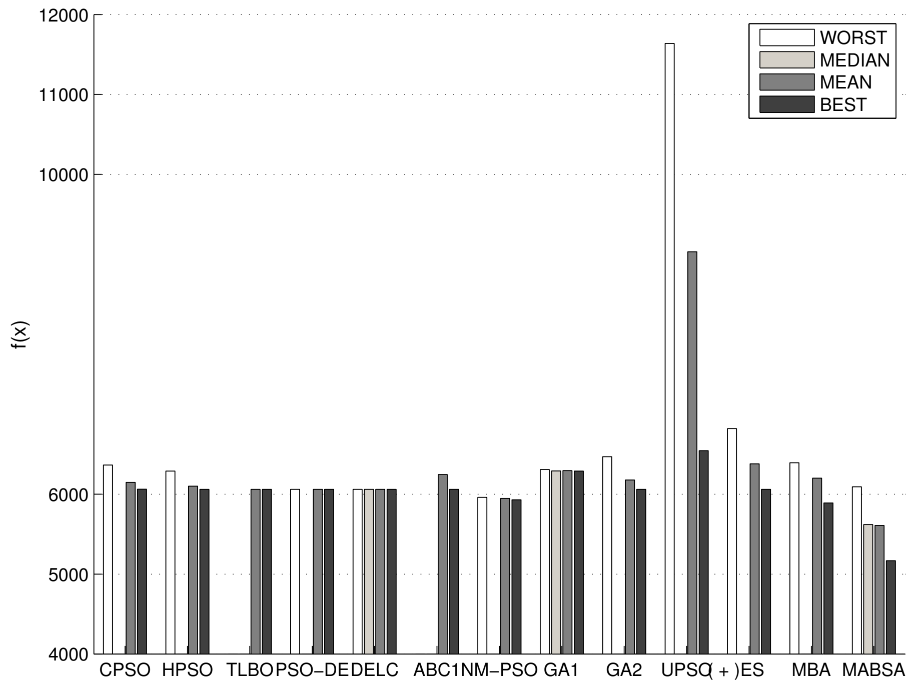


Figure 5.10: Bar plot of statistical results obtained using different algorithms for pressure vessel design optimisation problem

Three-truss bar design optimisation problem

The three-truss bar design optimisation problem is defined as:

$$\text{Minimise } F(x) = (2\sqrt{2x_1} + x_2) \times l$$

subject to

$$\begin{aligned} g_1(x) &= \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0 \\ g_2(x) &= \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0 \\ g_3(x) &= \frac{1}{\sqrt{2x_1} + x_1} P - \sigma \leq 0 \end{aligned} \tag{5.9}$$

where

$$0.0 \leq x_i \leq 1.0, \quad i = 1, 2$$

$$l = 100cm, \quad P = 2kN/cm^2, \quad \sigma = 2kN/cm^2$$

The *best* solution of MABSA for solving three-truss bar design optimisation problem is listed in the Table 5.8. By using only 700 bats, MABSA is able to reach the global optimum solution without trapping into a local optimum. In conjunction with that, as in Figure 5.11, MABSA starts to converge swiftly to the *best* solution just after 400 NFEs or within 7.8000 $seconds$.

The performance of MABSA has been compared with six established algorithms taken from literatures to solve this problem. These include SC, PSO-DE, DELC, DEDS, HEA-ACT and MBA. Definitely, the algorithm shows significant improvement of fitness function value obtained for the three-truss bar design optimisation problem.

As tabled in Table 5.9 and plot in bar plot as in Figure 5.12, MABSA has found the value that was better compared to other algorithms. For all statistical criteria considered, MABSA positively maintains its performance. Without a doubt, the smaller *standard deviation* existed after MABSA completing 30 runs demonstrated that the algorithm is more robust when solving the three-truss bar design optimisation problem. In this case, the MABSA is in third ranking of algorithm robustness behind DELC and PSO-DE from all algorithms evaluated.

Table 5.8: Results of the *best* solution obtained from MABSA for three-truss bar design optimisation problem

Items	Value
Run No.	18
No. of $Bats$	700
NFEs	70000
Time to converge ($seconds$)	7.7837
Iteration to converge	33
$F(x)$	263.8955
Optimum value of $F(x)$	263.9000

Table 5.9: Comparison of statistical results obtained using different algorithms for three-truss bar design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
SC	263.9698	263.8989	263.9033	263.8958	$1.2580e^{-02}$	17610
PSO-DE	263.8958	n/a	263.8958	263.8958	$1.2000e^{-10}$	17600
DELC	263.8958	263.8958	263.8958	263.8958	$4.3000e^{-14}$	10000
DEDS	263.8959	263.8958	263.8958	263.8958	$9.7200e^{-07}$	15000
HEA-ACT	263.8961	263.8959	263.8959	263.8958	$4.9000e^{-05}$	15000
MBA	263.9160	n/a	263.8980	263.8959	$3.9300e^{-03}$	13280
MABSA	263.8955	263.8955	263.8955	263.8955	$3.775720e^{-08}$	87650

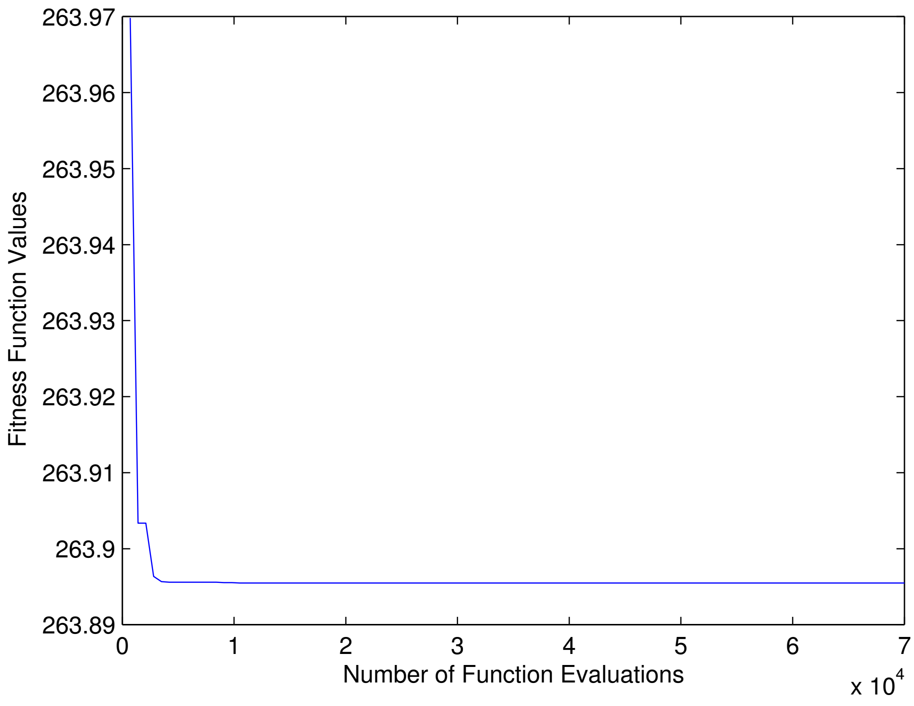


Figure 5.11: Convergence graph of the best solution of MABSA for three-truss bar design optimisation problems

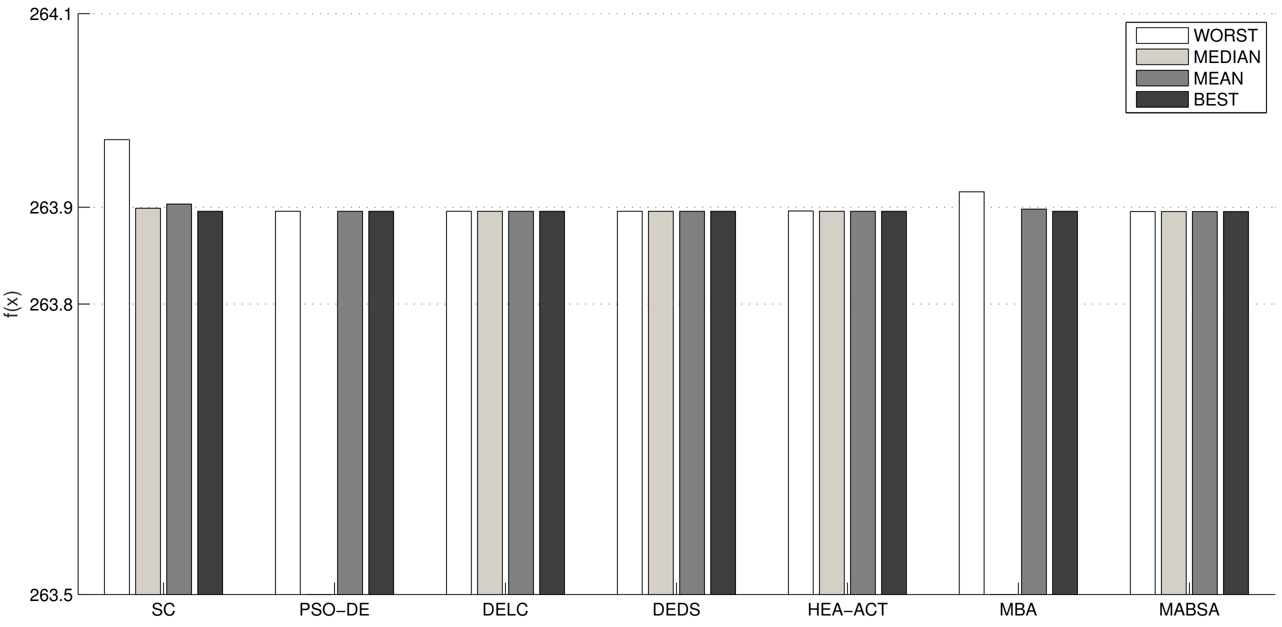


Figure 5.12: Bar plot of statistical results obtained using different algorithms for three-truss bar design optimisation problem

Gear train design optimisation problem

The gear train design optimisation problem is defined as:

$$\begin{aligned} &\text{Minimise } F(x) = ((1/1.6931) - (x_3x_2/x_1x_4))^2 \\ &\quad \text{where} \\ &\qquad 12 \le x_i \le 60, \quad i = 1,2,3,4 \end{aligned} \tag{5.10}$$

Table 5.10 depicts the information of the best solution achieved by MABSA for gear train design optimisation problem. The total *NFE*s used by MABSA to obtain the *best* solution were 89000 but it only needed approximately 1200 *NFE*s (as in Figure 5.13) or 18.0059 *seconds* to converge to the *best* fitness function value of $2.7473e^{-16}$.

The MABSA has been evaluated beside three other established algorithms found from literature which are ABC1, UPSO and MBA. The MABSA performed better than the three other algorithms evaluated for solving this task. As recorded in Table 5.11 and illustrated in Figure 5.14, MABSA was very excellent in finding the minimum fitness function for the problem considered compared to the ABC1, UPSO or MBA. In fact, the *worst* solution acquired by MABSA which is $1.8761e^{-12}$ was almost equal to the best solution of the other algorithms.

When discussing the algorithm robustness, the outstanding performance of the MABSA continues as compared to three established algorithms. The statement is present by the *standard deviation* value of $5.3938e^{-13}$ recorded by MABSA which was mathematically smaller than ABC1 ($5.5258e^{-10}$), UPSO ($1.0963e^{-07}$) or MBA ($3.9400e^{-09}$).

Table 5.10: Results of the *best* solution obtained from MABSA for gear train design optimisation problem problem

Items	Value
Run No.	13
No. of $Bats$	891
NFEs	89100
Time to converge ($seconds$)	18.0059
Iteration to converge	79
$F(x)$	$2.7473e^{-16}$
Optimum value of $F(x)$	$2.3500e^{-9}$

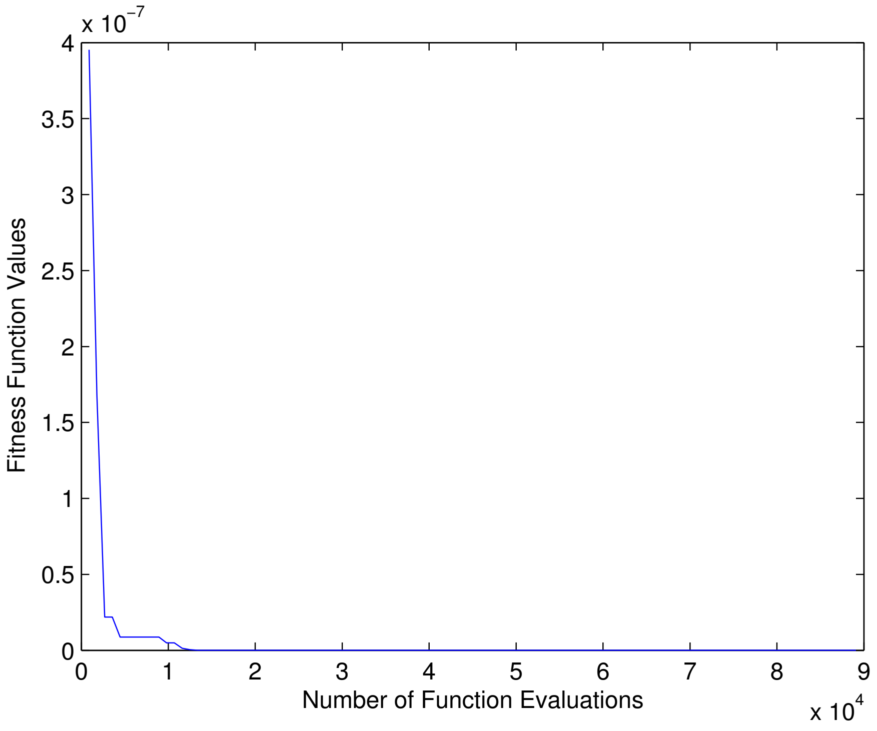


Figure 5.13: Convergence graph of the best solution of MABSA for gear train design optimisation problem

Table 5.11: Comparison of statistical results obtained using different algorithms for gear train design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	NFEs
ABC1	n/a	n/a	$3.6413e^{-10}$	$2.7009e^{-12}$	$5.5258e^{-10}$	30000
UPSO	$8.9490e^{-07}$	n/a	$3.8059e^{-08}$	$2.7085e^{-12}$	$1.0963e^{-07}$	100000
MBA	$2.0629e^{-08}$	n/a	$2.4716e^{-09}$	$2.7009e^{-12}$	$3.9400e^{-09}$	1120
MABSA	$1.8761e^{-12}$	$3.4364e^{-13}$	$4.7837e^{-13}$	$2.7473e^{-16}$	$5.3938e^{-13}$	91007

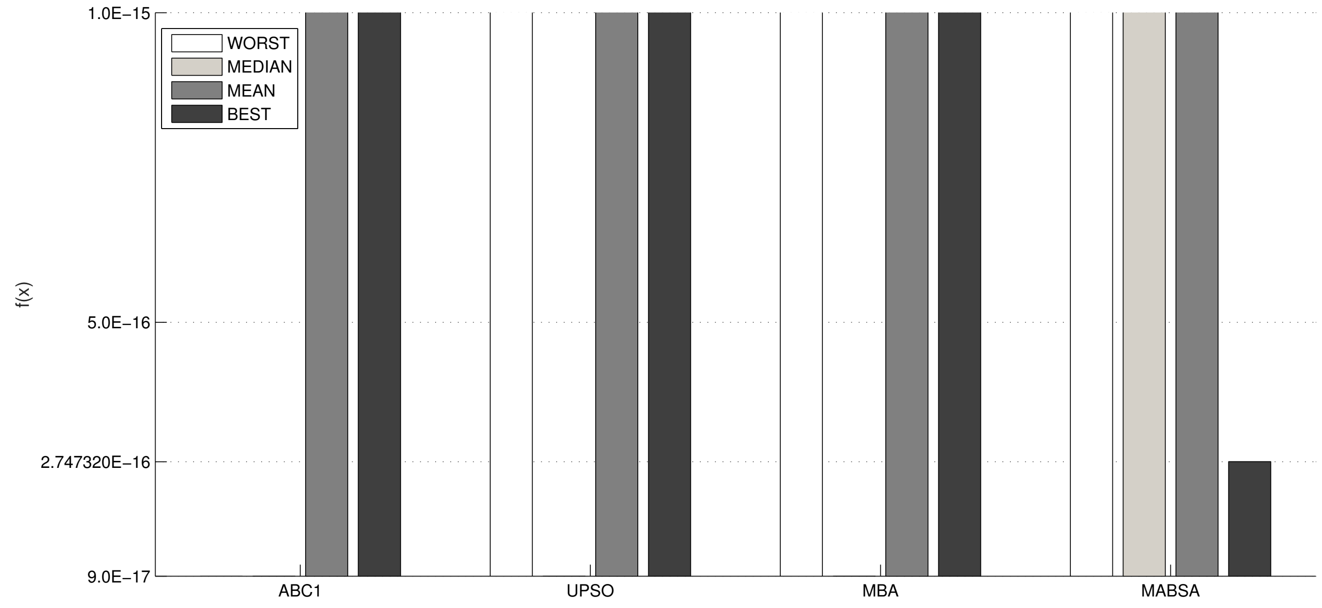


Figure 5.14: Bar plot of statistical results obtained using different algorithms for gear train design optimisation problem

Speed reducer design optimisation problem

The speed reducer design optimisation problem is defined as:

$$\text{Minimise } F(x) = 0.7854x_1x_2^2(3.3333x_3^2+14.9334x_3-43.0934)-1.508x_1(x_6^2+x_7^2)$$
$$+7.4777(x_6^3+x_7^3)+0.7854(x_4x6^2+x_5x_7^2)$$

subject to

$$\begin{aligned}
g_1(x) &= \frac{27}{x_1x_2^2x_3}-1\leq 0\\
g_2(x) &= \frac{397.5}{x_1x_2^2x_3^2}-1\leq 0\\
g_3(x) &= \frac{1.93x_4^3}{x_2x_6^4x_3}-1\leq 0\\
g_4(x) &= \frac{1.93x_5^3}{x_2x_7^4x_3}-1\leq 0\\
g_5(x) &= \frac{[(745(x_4/x_2x_3)^2+16.9\times10^6]^{1/2}}{110x_6^3}-1\leq 0\\
g_6(x) &= \frac{[(745(x_5/x_2x_3)^2+157.5\times10^6]^{1/2}}{85x_7^3}-1\leq 0\\
g_7(x) &= \frac{x_2x_3}{40}-1\leq 0\\
g_8(x) &= \frac{5x_2}{x_1}-1\leq 0\\
g_9(x) &= \frac{x_1}{12x_2}-1\leq 0\\
g_{10}(x) &= \frac{1.5x_6+1.9}{x_4}-1\leq 0\\
g_{11}(x) &= \frac{1.1x_7+1.9}{x_5}-1\leq 0
\end{aligned}\tag{5.11}$$

where

$$\begin{aligned}
&2.6\leq x_1\leq 3.6\\
&0.7\leq x_2\leq 0.8\\
&17.0\leq x_3\leq 28.0\\
&7.3\leq x_4,x_5\leq 8.3\\
&2.9\leq x_6\leq 3.9\\
&5.0\leq x_7\leq 5.5
\end{aligned}$$

The results of the *best* solution by MABSA solved speed reducer design optimisation problem are documented in Table 5.12. MABSA magnificently achieved the *best* fitness function for the problem, 2903.4328 in 1.9065 *seconds*. In term of NFEs, the MABSA started to converge to the *best* solution after approximately 400 NFEs (out of total 100000 NFEs analysed) as in Figure 5.15.

Table 5.12: Results of the *best* solution obtained from MABSA for speed reducer design optimisation problem

Items	Value
Run No.	12
No. of *Bats*	1000
*NFE*s	100000
Time to converge (*seconds*)	1.9065
Iteration to converge	5
$F(x)$	2903.4328
Optimum value of $F(x)$	2996.3480

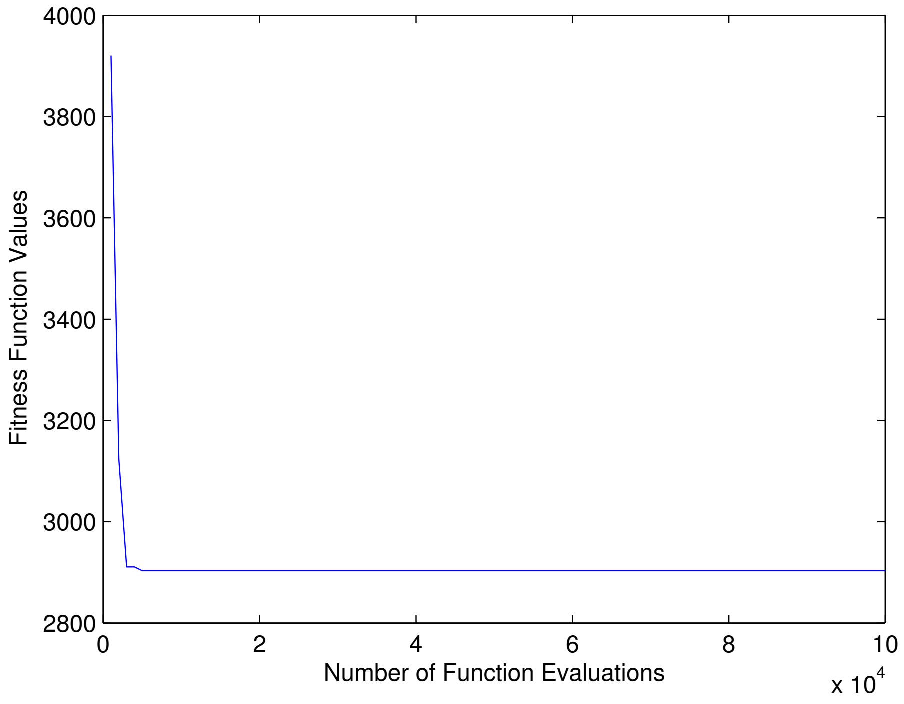


Figure 5.15: Convergence graph of the best solution of MABSA for speed reducer design optimisation problem

Besides, MABSA is evaluated alongside other eight methods taken from established literature to solve the speed reducer design optimisation problem. There are SC, PSO-DE, DELC, DEDS, HEA-ACT, ABC1, (μ + λ)ES and MBA.

When the comparison between statistical results obtained by all algorithms as in Table 5.13 and plotted on bar plot in Figure 5.16 is made, MABSA had shown more shining results. The statistical results by MABSA are better for all the criteria evaluated which are *worst*, *median*, *mean* and *best*. For instances, the *mean* value; 2939.3242 and *best* value; 2903.4328 recorded in MABSA were the most optimum solution found on each respective criteria to solve the discussed problem.

Unfortunately, the robustness of MABSA to solve the problem was the worst compared to other established algorithms. The *standard deviation* acquired from 30 runs of MABSA only noted 29.2630. For the record, the DEDS and DELC are top two robust algorithms to solve the speed reducer design problem as each algorithm logged the *standard deviation* values of $3.5800e^{-12}$ and $1.9000e^{-12}$ respectively.

Table 5.13: Comparison of statistical results obtained using different algorithms for speed reducer design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
SC	3009.9647	3001.7583	3001.7583	2994.7442	4.0091	54456
PSO-DE	2996.3482	n/a	2996.3482	2996.3482	$1.0000e^{-07}$	70100
DELC	2994.4711	2994.4711	2994.4711	2994.4711	$1.9000e^{-12}$	30000
DEDS	2994.4711	2994.4711	2994.4711	2994.4711	$3.5800e^{-12}$	30000
HEA-ACT	2994.7523	2994.5998	2994.6134	2994.4991	$7.0000e^{-02}$	40000
ABC1	n/a	n/a	2997.0584	2997.0584	0.0000	30000
(μ+λ)ES	2996.3481	n/a	2996.3481	2996.3481	0.0000	30000
MBA	2999.6524	n/a	2996.7690	2994.4825	1.5600	6300
MABSA	2992.6411	2932.6487	2939.3242	2903.4328	29.2630	90433

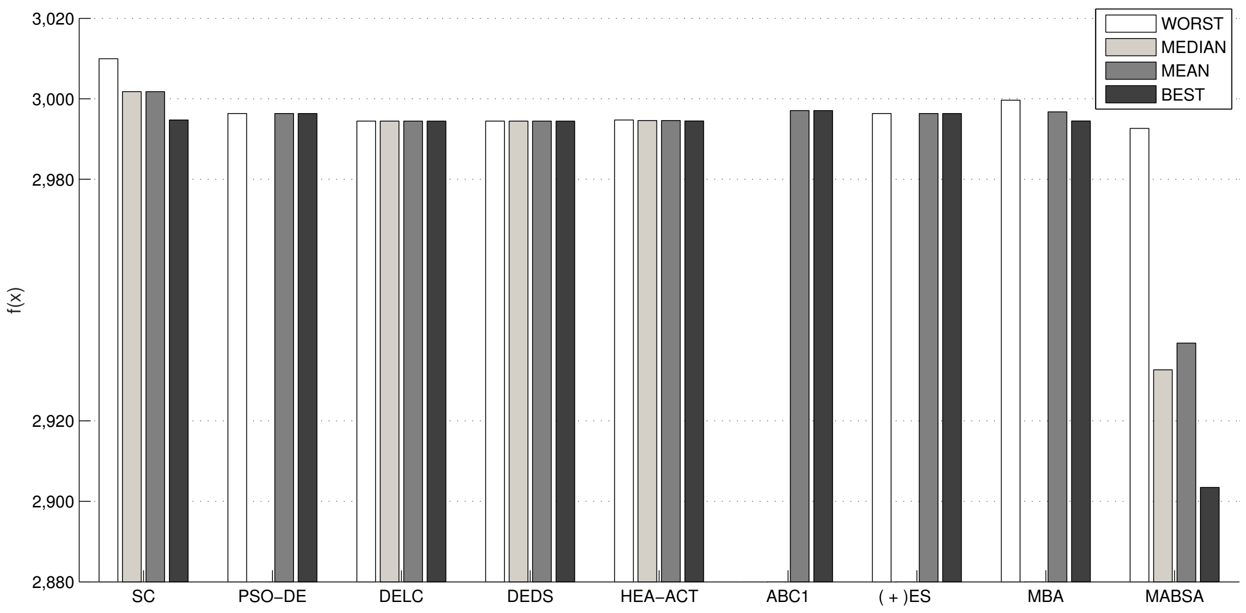


Figure 5.16: Bar plot of statistical results obtained using different algorithms for speed reducer design optimisation problem

Welded beam design optimisation problem

The welded beam design optimisation problem is defined as:

$$
\begin{aligned}
&\text{Minimise } F(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14+x_2)\\
&\text{subject to}\\
&\quad g_1(x) = \tau(x) - \tau_{max} \leq 0\\
&\quad g_2(x) = \sigma(x) - \sigma_{max} \leq 0\\
&\quad g_3(x) = x_1 - x_4 \leq 0\\
&\quad g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14+x_2) - 5 \leq 0\\
&\quad g_5(x) = 0.125 - x_1 \leq 0\\
&\quad g_6(x) = \delta(x) - \delta_{max} \leq 0\\
&\quad g_7(x) = P - P_c \leq 0\\
&\text{where}\\
&\quad 0.1 \leq x_i \leq 2.0, \quad i = 1,4\\
&\quad 0.1 \leq x_i \leq 10.0, \quad i = 2,3\\
&\quad \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}\\
&\quad \tau^{'} = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau^{''} = \frac{MR}{J}, \quad M = P(L + \frac{x_2}{2})\\
&\quad R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}\\
&\quad J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}\\
&\quad \sigma(x) = \frac{6PL}{x_4x_3^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4},\\
&\quad P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2} \times \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&\quad P = 6000lb, \; E = 30\times10^6 psi,\\
&\quad L = 4in, \; G = 12\times10^6 psi,\\
&\quad \tau_{max} = 13600psi, \; \sigma_{max} = 30000psi, \; \delta_{max} = 0.25in
\end{aligned}
\tag{5.12}
$$

The data of the *best* fitness function found by MABSA for welded beam design optimisation problem is tabled in Table 5.14. The *best* solution for the problem; 1.6308 is found on the sixth run of MABSA. On the other hand, Figure 5.17 shows the convergence graph for the *best* solution of MABSA. As seen from the figure, the MABSA started to reach the *best* fitness function value of welded beam design problem after 2000 NFEs out of 10000 NFEs used.

Table 5.14: Results of the *best* solution obtained from MABSA for welded beam design optimisation problem

Items	Value
Run No.	6
No. of *Bats*	1000
*NFE*s	100000
Time to converge (*seconds*)	86.3057
Iteration to converge	84
$F(x)$	1.6308
Optimum value of $F(x)$	1.7249

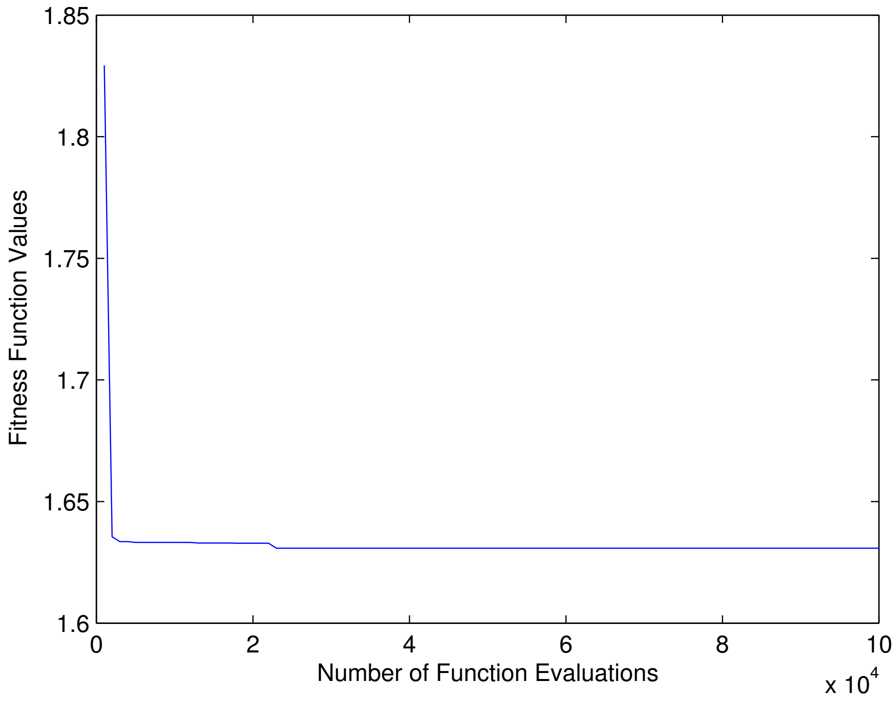


Figure 5.17: Convergence graph of the best solution of MABSA for welded beam design optimisation problem

The results of other algorithms solving the welded beam design optimisation problem taken from the literature are also used to compare with the performance of MABSA. The algorithms included CPSO, HPSO, TLBO, PSO-DE, DELC, ABC1, NM-PSO, GA1, GA2, UPSO, $(\mu + \lambda)$ES and MBA.

The MABSA also outperform as compared to other algorithms considered for this problem. This statement was demonstrated from the statistical results as tabled in Table 5.15 and depicted in bar plot of Figure 5.18. The back to back of outstanding results are achieved by MABSA as compared to all twelve algorithms in every statistical criterion. Except for the *worst* criteria; *median*, *mean* and *best* fitness function values acquired by MABSA were under 1.7000 which become the only algorithm to break that line.

As the *standard deviation* values presented in Table 5.15, the robustness of MABSA to solve the welded beam design optimisation problem also is on a par with most of the established algorithms studied. Although the MBA, PSO-DE and DELC managed to put their robustness ability in a class by itself, the value of $2.8858e^{-02}$ achieved by MABSA is still within the adequate range of robustness as it is approaching 0.0000.

Table 5.15: Comparison of statistical results obtained using different algorithms for welded beam design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CPSO	1.7821	n/a	1.7488	1.7280	$1.2926e^{-02}$	200000
HPSO	1.8143	n/a	1.7490	1.7249	$4.0049e^{-02}$	81000
TLBO	n/a	n/a	1.7284	1.7249	n/a	10000
PSO-DE	1.7249	n/a	1.7249	1.7249	$6.7000e^{-16}$	66600
DELC	1.7249	1.7249	1.7249	1.7249	$4.1000e^{-13}$	20000
ABC1	n/a	n/a	1.7419	1.7249	$3.1000e^{-02}$	30000
NM-PSO	1.7334	n/a	1.7264	1.7247	$3.4970e^{-03}$	80000
GA1	1.7858	1.7736	1.7720	1.7483	$1.1223e^{-02}$	900000
GA2	1.9934	n/a	1.7927	1.7283	$7.4713e^{-02}$	80000
UPSO	2.8441	n/a	1.9682	1.7656	$1.5542e^{-01}$	100000
$(\mu+\lambda)$ES	2.0746	n/a	1.7769	1.7249	$8.8000e^{-02}$	30000
MBA	1.7249	n/a	1.7249	1.7249	$6.9400e^{-19}$	47340
MABSA	1.7241	1.6800	1.6776	1.6308	$2.8858e^{-02}$	86113

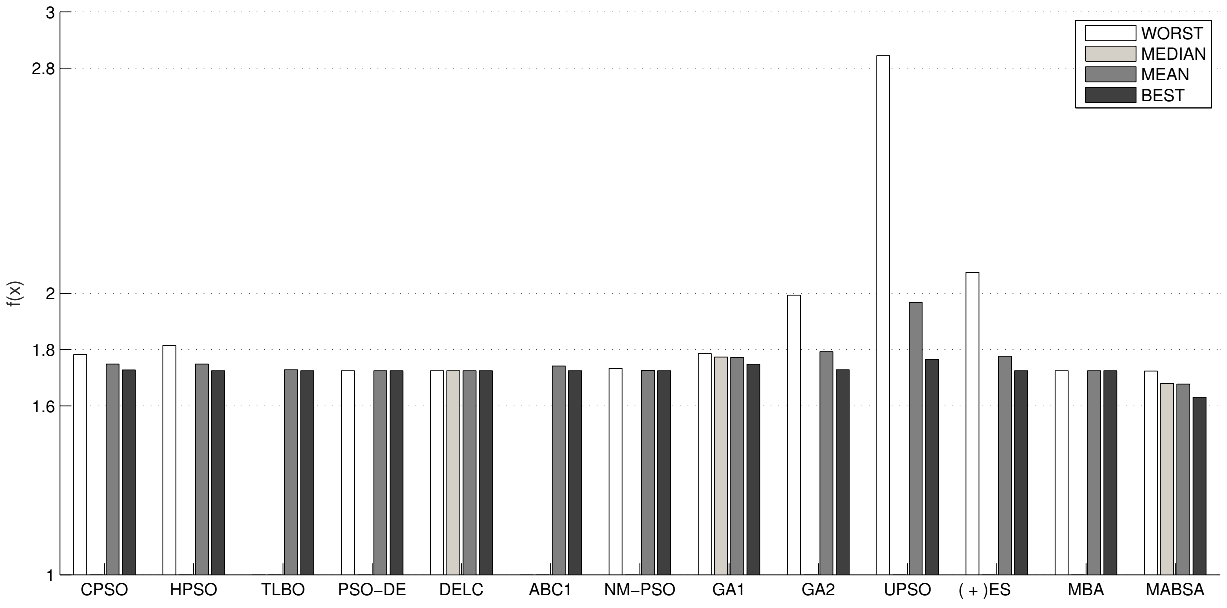


Figure 5.18: Bar plot of statistical results obtained using different algorithms for welded beam design optimisation problem

Tension/compression spring design optimisation problem

The tension/compression spring design optimisation problem is defined as:

$$
\begin{aligned}
&\text{Minimise } F(x) = (x_3 + 2)x_2x_1^2 \\
&\text{subject to} \\
&\qquad g_1(x) = 1 - (x_2^3x_3/71785x_1^4) \leq 0 \\
&\qquad g_2(x) = (4x_2^2 - x_1x_2/12566(x_2x_1^3 - x_1^4)) + (1/5108x_1^2) \leq 0 \\
&\qquad g_3(x) = 1 - (140.45x_1/x_2^2x_3) \leq 0 \\
&\qquad g_4(x) = (x_2 + x_1)/1.5 - 1 \leq 0 \\
&\text{where} \\
&\qquad 0.05 \leq x_1 \leq 2.00 \\
&\qquad 0.25 \leq x_2 \leq 1.30 \\
&\qquad 2.00 \leq x_3 \leq 15.00
\end{aligned}
\tag{5.13}
$$

The data of *best* solution achieved by MABSA solving tension/compression spring design optimisation problem is depicted in Table 5.16. MABSA managed to reach at the *best* fitness function value of the problem; 0.0123 just after sixteenth iterations. Or in NFEs, the problem started to get the *best* solution provided by MABSA after 1800 NFEs (as in Figure 5.19) out of 100000 total NFEs used. These have demonstrated that MABSA has capability to converge faster to the optimum solution of the problem studied.

Table 5.16: Results of the *best* solution obtained from MABSA for tension/compression spring design optimisation problem

Items	Value
Run No.	24
No. of $Bats$	1000
NFEs	100000
Time to converge ($seconds$)	4.7440
Iteration to converge	16
$F(x)$	0.0123
Optimum value of $F(x)$	0.0127

To further demonstrate the capability of MABSA to solve the tension/compression spring design optimisation problem, the statistical results of MABSA have been compared with another set of established algorithms. The statistical results from selected algorithms to solve the problem that appeared in literature are considered as a comparison and are shown in Table 5.17 and also the bar plotted as in Figure 5.20. The algorithms involved are CPSO, HPSO, TLBO, SC, PSO-DE, DELC, DEDS, HEA-ACT, ABC1, NMPSO, GA1, GA2, UPSO, (μ + λ)ES and MBA.

Again, the MABSA is able to perform well in all statistical aspects compared to the fifteen other methods. For instance, MABSA is able to chart 0.0123 in the *best* criteria but a majority of algorithms are able to achieve only 0.0127. In MABSA, the *mean* value for the problem was 0.0125 while other considered algorithms have produced the *mean* value in the range of 0.0126 to 0.0230 which was not the minimum fitness function value as targeted. The *standard deviation* achieved by MABSA; $1.4195e^{-04}$ which was approaching zero indicates that the MABSA is a reliable and robust algorithm to solve the tension/compression spring design optimisation problem. As well as MABSA, other algorithms considered also managed to be a robust algorithm to solve the problem.

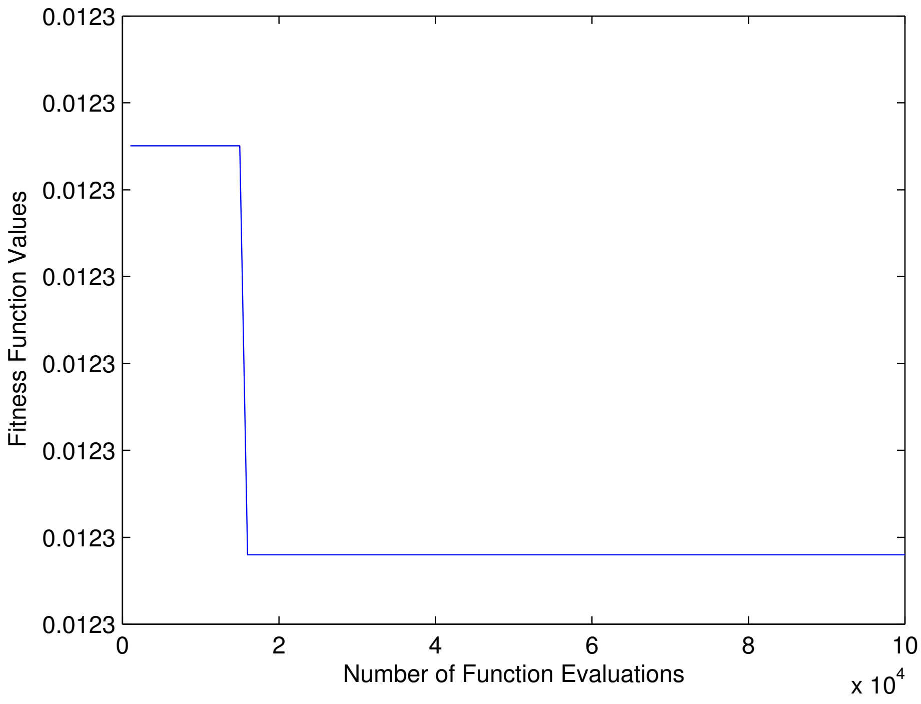


Figure 5.19: Convergence graph of the best solution of MABSA for tension/compression design optimisation problem

Table 5.17: Comparison of statistical results obtained using different algorithms for tension/compression spring design optimisation problem. ("n/a" means not available)

Method	*Worst*	*Median*	*Mean*	*Best*	*Standard deviation*	*NFE*s
CPSO	0.0129	n/a	0.0127	0.0127	$5.1985e^{-05}$	200000
HPSO	0.0127	n/a	0.0127	0.0127	$1.5824e^{-05}$	81000
TLBO	n/a	n/a	0.0127	0.0127	n/a	10000
SC	0.0167	0.0129	0.0129	0.0127	$5.9200e^{-04}$	25167
PSO-DE	0.0127	n/a	0.0127	0.0127	$4.9000e^{-12}$	42100
DELC	0.0127	0.0127	0.0127	0.0127	$1.3000e^{-07}$	20000
DEDS	0.0127	0.0127	0.0127	0.0127	$1.2000e^{-05}$	24000
HEA-ACT	0.0127	0.0127	0.0127	0.0127	$1.4000e^{-09}$	24000
ABC1	n/a	n/a	0.0127	0.0127	$1.2813e^{-02}$	30000
NM-PSO	0.0126	n/a	0.0126	0.0126	$8.7375e^{-07}$	80000
GA1	0.0128	0.0128	0.0128	0.0127	$3.9390e^{-05}$	900000
GA2	0.0130	n/a	0.0127	0.0127	$5.9000e^{-05}$	80000
UPSO	0.0504	n/a	0.0230	0.0131	$7.2057e^{-03}$	100000
$(\mu+\lambda)$ES	0.0141	n/a	0.0132	0.0127	$3.9000e^{-04}$	30000
MBA	0.0129	n/a	0.0127	0.0127	$6.3000e^{-05}$	7650
MABSA	0.0127	0.012480	0.0125	0.0123	$1.4195e^{-04}$	89680

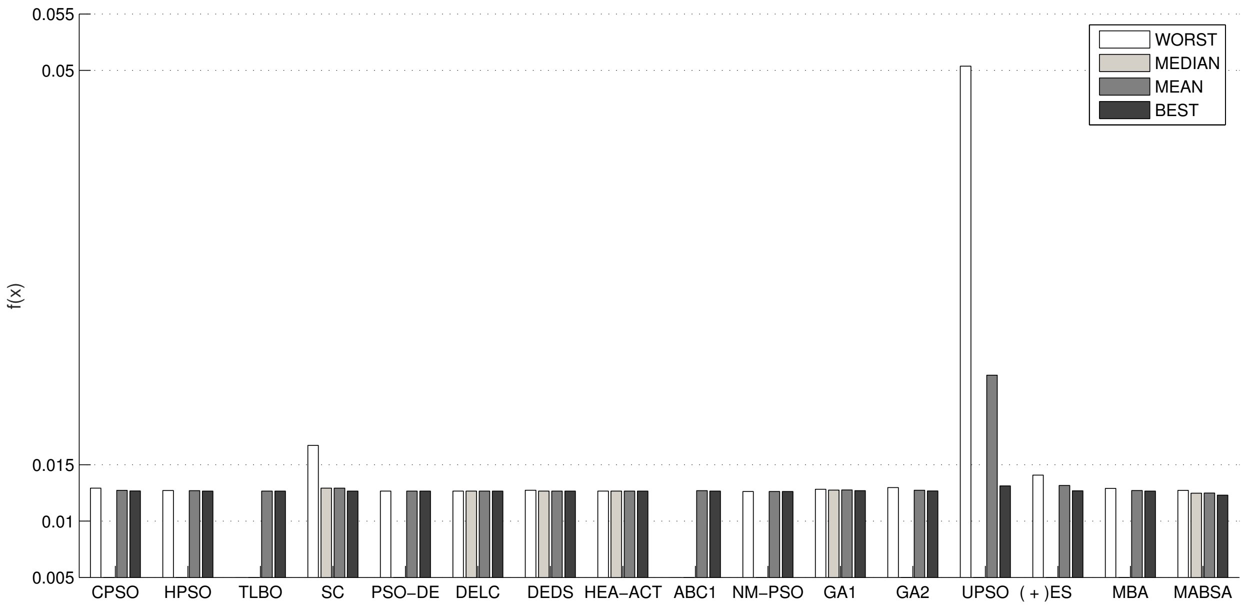


Figure 5.20: Bar plot of statistical results obtained using different algorithms for tension/compression design optimisation problem

5.2.3 Overall comparison of all considered algorithms

The mean absolute error (*MAE*) of all algorithms are computed to rank all considered algorithms. *MAE* is a statistical criterion that indicates how far the results are from the actual values as:

$$\begin{aligned}
&\text{MAE} = \frac{\sum_{z}^{i=1} |m_i - h_i|}{z} \\
&\text{where} \\
&m_i = \text{mean of optimum achieved results} \\
&h_i = \text{global optimum value} \\
&z = \text{number of test functions}
\end{aligned} \tag{5.14}$$

All considered algorithms for constrained optimisation benchmark test functions are ranked in Table 5.18 based on their corresponding MAE's. The table showed that MABSA is at the highest ranking from 15 considered algorithms.

For engineering design optimisation problems, all considered algorithms are ranked as in Table 5.19. However, only MABSA and MBA were compared for all 6 (z = 6) engineering design optimisation problems, while other considered algorithms were compared on three to five (z = 3 or 4 or 5) problems. The MABSA was at the peak of ranking for all 16 considered algorithms without reflecting on the value of z.

Table 5.18: Rank of algorithms for constrained optimisation benchmark test functions

Algorithm	*MAE*	Ranking
MABSA	-66.1095	1
DEDS	-6.9250e^{-5}	2
DELC	-4.4250e^{-5}	3
HEA-ACT	-4.4250e^{-5}	4
ISR	-4.4250e^{-5}	4
α Simplex	5.8500e^{-5}	6
PSO-DE	7.4750e^{-5}	7
ABC2	1.2058e^{-3}	8
CULDE	2.0820e^{-3}	9
MBA	5.737e^{-3}	10
ASCHEA	0.0135	11
SMES	0.1357	12
SAPF	3.9220	13
SR	21.4750	14
CRGA	55.8464	15

Table 5.19: Rank of algorithms for engineering design optimisation problems

Algorithm	z	MAE	Ranking
MABSA	6	-84.8000	1
NM-PSO	3	-37.6408	2
DEDS	3	-0.6271	3
HEA-ACT	3	-0.5797	4
DELC	5	-0.3762	5
PSO-DE	5	-0.0008	6
TBLO	3	0.0013	7
SC	3	1.80454	8
HPSO	3	13.4142	9
MBA	6	23.5588	10
CPSO	3	29.1478	11
ABC1	5	37.2643	12
GA2	3	39.2024	13
GA1	3	78.0588	14
(μ+λ)ES	4	80.0692	15
UPSO	4	743.2724	16

5.3 Application of modified adaptive bats sonar algorithm to solve constrained optimisation problems

5.3.1 Weight optimisation of the car side impact design

The problem is to find the minimum total weight (F) in kg of the car side impact design (as shown in Figure 5.21) that consists of eleven design variables and is subject to ten design constraints.

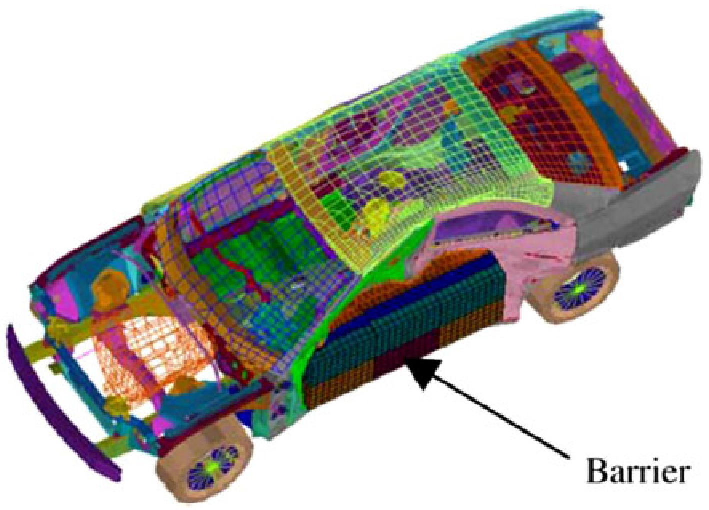


Figure 5.21: A finite element method (FEM) model of car side impact

The design variables are thickness of B-pillar inner (x_1), thickness of B-pillar reinforcement (x_2), thickness of floor side inner (x_3), thickness of cross member (x_4), thickness of door beam (x_5), thickness of door beltline reinforcement (x_6), thickness of roof rail (x_7), materials of B-pillar inner (x_8), materials of floor side inner (x_9), barrier height (x_{10}) and hitting position (x_{11}).

The ten design constraints include: load in abdomen (F_a), dummy upper chest (VC_u), dummy middle chest (VC_m), dummy lower chest (VC_l), upper rib deflection (Δur), middle rib deflection (Δmr), lower rib deflection (Δlr), pubic force (F_p), velocity of V-pillar at middle point (V_{MBP}) and velocity of front door at V-pillar (V_{FD}). The problem is formulated as:

$$\begin{aligned}
&\text{Minimise } F(x) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \\
&\text{subject to} \\
&\quad F_a = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \leq 1 \\
&\quad VC_u = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} \\
&\qquad + 0.080405x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \leq 0.32 \\
&\quad VC_m = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 \\
&\qquad + 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} \\
&\qquad - 0.0005354x_6x_{10} + 0.00121x_8x_{11} \leq 0.32 \\
&\quad VC_l = 0.074 + 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 \leq 0.32 \\
&\quad \Delta_{ur} = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \leq 32 \\
&\quad \Delta_{mr} = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} \\
&\qquad - 9.98x_7x_8 + 22.0x_8x_9 \leq 32 \\
&\quad \Delta_{lr} = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \leq 32 \\
&\quad F_p = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 \leq 4 \\
&\quad V_{MBP} = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} \leq 9.9 \\
&\quad V_{FD} = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 \leq 15.7 \\
&\text{where} \\
&\quad 0.5 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7 \leq 1.5 \\
&\quad x_8, x_9 \in \{0.192, 0.345\} \\
&\quad -30 \leq x_{10}, x_{11} \leq 30
\end{aligned} \tag{5.15}$$

The MABSA is used by 30 independent runs to find the optimum weight of the problem. The MABSA is capable to find the minimum total weight (F) of the car side impact design. Figure 5.22 plotted the optimum fitness function obtained for every independent run. Table 5.20 shows the results of solution obtained by MABSA. The *best* weight recorded using MABSA is 19.29614 *kg* while the *worst* value is 23.05891 *kg*. The *standard deviation* value, 0.805949 reflected the distribution of solutions in 30 independent runs located near to the *mean* value 21.63737 *kg*.

Table 5.20: Performance results of MABSA to optimise the weight of the car side impact design

Item	Value
Best minimum weight (F)	19.2961
Thickness of B-pillar inner (x_1)	0.5237
Thickness of B-pillar reinforcement (x_2)	0.5108
Thickness of floor side inner (x_3)	0.5721
Thickness of cross member (x_4)	0.7122
Thickness of door beam (x_5)	0.6153
Thickness of door beltline reinforcement (x_6)	1.3011
Thickness of roof rail (x_7)	0.7199
Materials of B-pillar inner (x_8)	0.3450
Materials of floor side inner (x_9)	0.1920
Barrier height (x_{10})	0.7828
Hitting position (x_{11})	0.5977
Load in abdomen (F_a)	2.0174
Dummy upper chest (VC_u)	0.5617
Dummy middle chest (VC_m)	0.5368
Dummy lower chest (VC_l)	0.3142
Upper rib deflection (Δur)	61.8428
Middle rib deflection (Δmr)	63.3651
Lower rib deflection (Δlr)	71.0220
Pubic force (F_p)	8.3112
Velocity of V-pillar at middle point (V_{MBP})	19.98274768
Velocity of front door at V-pillar (V_{FD})	31.27357581
Worst minimum weight	23.0589
Median minimum weight	21.7190
Mean minimum weight	21.6374
Standard deviation minimum weight	0.805949

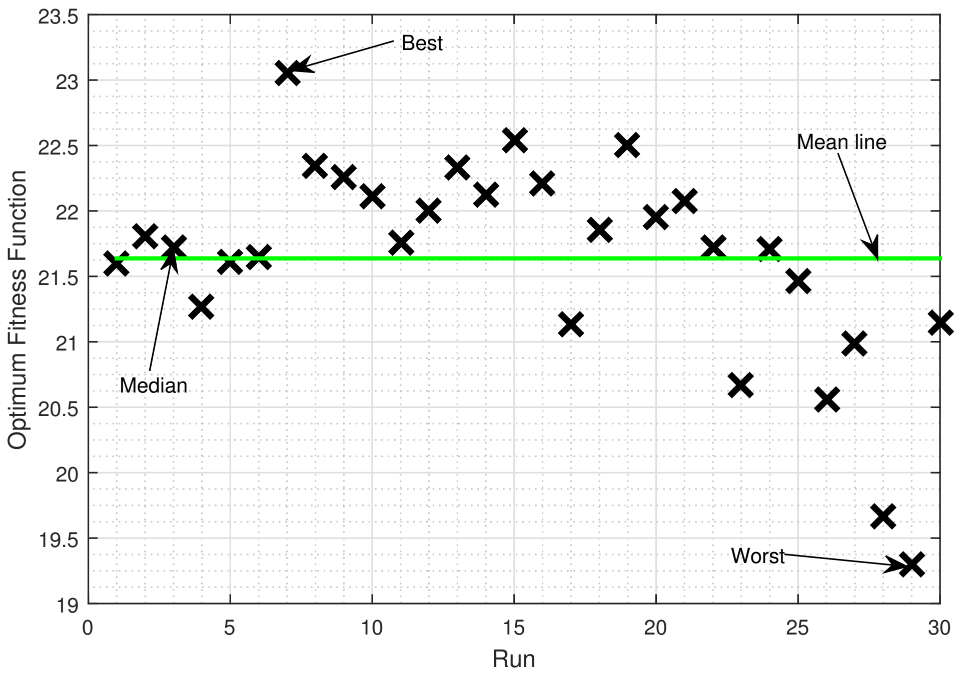


Figure 5.22: Optimum fitness function of car side impact design problem obtained by 30 independent runs

5.3.2 Efficiency optimisation of brushless wheel DC motor

The problem is to maximise the efficiency (η) of brushless wheel DC motor that consist of five variables and subject to six constraints. The five variables are: bore stator diameter (D_s), magnetic induction in the air gap (B_e), current density in the conductor (δ), magnetic induction both in the teeth (B_d) and back iron (B_{cs}). The constraints to be consider are: total mass (M_{tot}), internal diameter (D_{int}), external diameter (D_{ext}), magnetics maximum current (I_{max}), temperature (T_a) and determinant used in the slot height calculation ($Discr$).

The problem can be briefly defined as in:

$$\begin{aligned}
&\text{Maximise } F(x) = \eta(x) \\
&\quad \text{subject to} \\
&\qquad M_{tot} \leq 15\,kg \\
&\qquad D_{ext} \leq 0.340\,m \\
&\qquad D_{int} \geq 0.076\,m \\
&\qquad I_{max} \geq 125\,A \\
&\qquad T_a < 120\,^{\circ}C \\
&\qquad Discr(D_s, \delta, B_d, B_e) \geq 0 \\
&\quad \text{where} \\
&\qquad 0.150\,m \leq D_s \leq 0.330\,m \\
&\qquad 0.9\,T \leq B_d \leq 1.8\,T \\
&\qquad 2.0\,A/mm^2 \leq \delta \leq 5.0\,A/mm^2 \\
&\qquad 0.5\,T \leq B_e \leq 0.76\,T \\
&\qquad 0.6\,T \leq B_{cs} \leq 1.6\,T
\end{aligned} \tag{5.16}$$

The MABSA was used by 30 independent runs to find the optimum efficiency of the problem, and was capable of maximising the efficiency (η) of brushless wheel DC motor. Figure 5.23 plotted the optimum fitness function obtained for every independent run. The performance results of MABSA are shown in Table 5.21. The *best* efficiency of the problem achieved by MABSA is 98.2517% while the *worst* efficiency is 94.4931%. The *mean* efficiency is 95.8900% while the *standard deviation* value of 0.8813 showed that the solutions from 30 independent runs are distributed not far from the *mean*.

Table 5.21: Performance results of MABSA to optimise the efficiency of brushless wheel DC motor

Item	Value
Best maximum efficiency (η)	98.2517
Bore stator diameter (D_s)	0.1507
Magnetic induction in the air gap (B_e)	0.6913
Current density in the conductors (δ)	2.4675
Magnetic induction both in the teeth (B_d)	1.0835
Back iron (B_{cs})	1.5102
Total mass (M_{tot})	8.2476
Internal diameter (D_{int})	0.1704
External diameter (D_{ext})	0.51831
Magnetic maximum current (I_{max})	130.0214
Temperature (T_a)	67.7746
Determinant ($Discr$)	0.1026
Worst maximum efficiency	94.4931
Median maximum efficiency	95.6494
Mean maximum efficiency	95.8900
Standard deviation maximum efficiency	0.8813

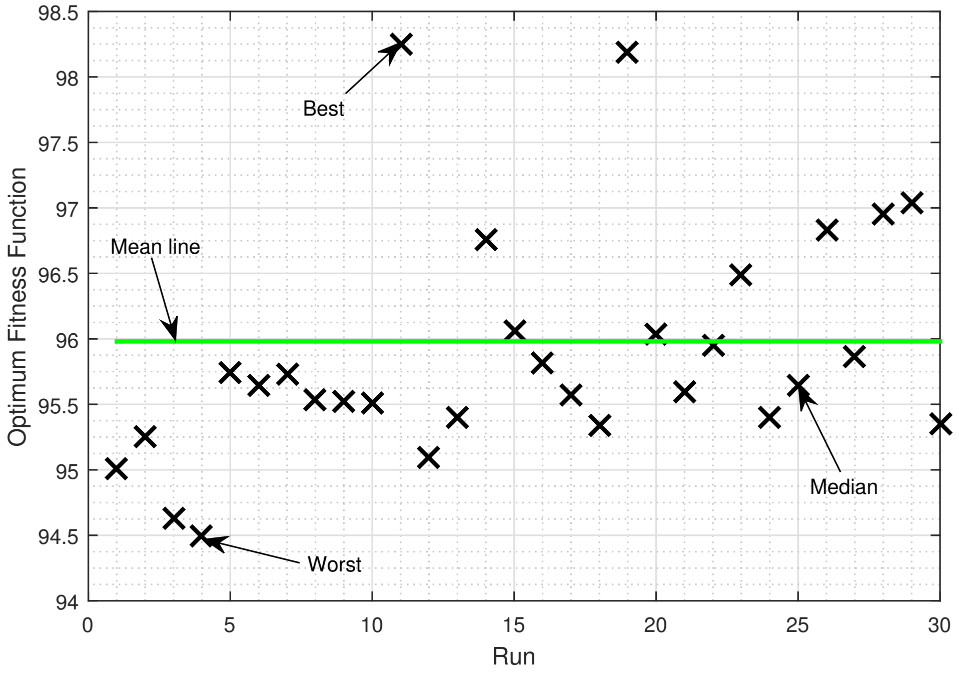


Figure 5.23: Optimum fitness function of brushless wheel DC motor problem obtained by 30 independent runs

Chapter 6

Hybrid modified adaptive bats sonar algorithm and particle swarm optimisation algorithm

6.1 A necessity to hybrid algorithm

For several optimisation problems, a swarm intelligence algorithm might be good enough to find the desired solution. But the challenge is in the case of multi objective optimisation problems, where the objectives are conflicting between each other. A rugged and efficient swarm intelligence algorithm is needed to acquire a set of Pareto optimum solutions that compromise all objectives considered.

The need has paved way to the need for hybridisation of swarm intelligence algorithms with other algorithms. The hybrid algorithm can make good use of the characteristics of different algorithms to achieve complementary advantages to improve the algorithm optimum performance and efficiency as well as the quality of the solution obtained by the algorithm.

There are lots of opportunities to a hybrid between the swarm intelligence algorithms. For instance, an algorithm population may be initialised by incorporating known solutions of another algorithm or the local search method of one algorithm may be hybridised with the new generations of another algorithm.

6.2 Particle swarm optimisation algorithm

6.2.1 The PSO algorithm in brief

Particle swarm optimisation is an evolutionary computation technique introduced by a psychologist, James Kennedy and an electrical engineer, Russell Eberhart in 1995. This algorithm has been inspired by the social behaviour of a swarm of birds and fishes. PSO has characteristics that are more attractive than the existing evolutionary computation. The characteristics include memory that can be maintained by any individual in the algorithm, build cooperation between the individuals and share information between the individuals. The algorithm has a simple theoretical framework, which is easy to code into a computer programme, and can generate high quality and focused solutions in relatively shorter computation times than other metaheuristic methods.

The term particles is used in the PSO algorithm for referring to individuals because each is associated with the velocity and acceleration though the particles do not have mass and volume. Meanwhile, the term swarm used in PSO is in accordance with the main principles of swarm intelligence that are proximity, quality, diversity of responsiveness, stability and adaptability.

The principle of proximity is represented in the PSO algorithm as a multi dimensional calculation in each iteration while the swarm of particles respond to quality criteria of personal and neighbourhood best positions. Besides, the principle of diversity of responsiveness in the PSO algorithm is also well represented by the provisions of reactions between personal best position and neighbourhood best position.The principle of stability is the ability of swarms to change their positions if and only if there is a change in the position of the personal best and global best position that also meets the principle of adaptability of the PSO algorithm.

6.2.2 The standard PSO algorithm

In PSO, all particles are treated as valueless particles of g-dimensional search space. Each particle will record its current coordinate in the problem space associated with its personal best solution, *pbest*. Meanwhile, the overall best solution and the location obtained so far by any particle in the swarm is labelled as *gbest*.

The concept of PSO involves changing the velocity of every particle toward the *pbest* and *gbest*. For instance, the position of jth particle is represented as:

$$x_j = x_{j,1}, x_{j,2}, \dots, x_{j,g} \tag{6.1}$$

where

g total dimension of the space

The jth best previous position represented as:

$$pbest_j = pbest_{j,1}, pbest_{j,2}, \dots, pbest_{j,g} \tag{6.2}$$

where the best $pbest_j$ among all particles in the swarm is denoted as $gbest$. The velocity of jth particle is represented as:

$$v_j = v_{j,1}, v_{j,2}, \dots, v_{j,g} \tag{6.3}$$

The new velocity and position of each particle at each iteration can be calculated as:

$$v_{j,g}^{(t+1)} = w.v_{j,g}^{(t)} + c_1 * rand() * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * rand() * (gbest_g - x_{j,g}^{(t)})$$

and

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)}, \quad j = 1,2,\ldots,n \quad and \quad g = 1,2,\ldots,m$$

where

$$-V_{max} \leq v_{j,g}^{(t)} \leq Vmax$$

n number of particles in a group

m number of members in a particle

t pointer of iterations (generations) (6.4)

$v_{j,g}^{(t)}$ velocity of particle j at iteration t

V_{max} maximum velocity

c_1, c_2 acceleration constant

$rand()$ random number between 0 and 1

$pbest_j$ $pbest$ of particle j

$gbest$ $gbest$ of the swarm

Here, the parameter maximum velocity (V_{max}) determines the resolution (or fineness) in the search space between the current velocity and target velocity . V_{max} is applied to provide damping the particles velocity to avoid the swarm system from exploding when the particles' searching process increase with time. So each particle's velocity in every dimension is tied to the V_{max} value. V_{max} value is set at the start of the iteration process and remains constant till iterations end. Critically, V_{max} value should not be either too high or too low. The particle will pass the good solution if the value is too high. In another way, the particle will be unable to explore beyond local solution sufficiently if V_{max} is too small. Instead,the researchers suggested that V_{max} is limit to x_{max}, the dynamic range of each variable range in every dimension.

Acceleration constants (c_1) and (c_2) are important in determining the motion trajectory of particles and controlling the influence of stochastic components of social and cognitive on overall particle's velocity.The constants are divided to c_1 as self-confidence factor to represent confidence level in every particle while c_2 is a swarm-confidence factor that represents the confidence level of particles to their neighbourhood. The value of c_1 and c_2 are set to 2.0 so that the particles will be attracted to the *pbest* and *gbest* positions equally. Setting to this value also enables smooth particles trajectory and permits particles to explore far from the target location before being tugged back to the appropriate region.

In general, inertia weight (w) is set in iteration decreasing mode as follows:

$$w = \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{6.5}$$

Here, $iter$ is current iteration while $iter_{max}$ is total number of iteration used. A suitable value of w_{max} is 0.9 while w_{min} is 0.4. This w is a mechanism to control the exploration and exploitation abilities in the swarm. The w value will drive the momentum of particles on current velocity influencing a new velocity. The w_{max} value diversifies the global exploration process while the w_{min} will concentrate on local exploitation. So, this parameter will be balanced between local and global search, besides it encourages the algorithm to shift from exploration mode to exploitation mode in order to find optimum solution. Algorithm 5 shows the PSO pseudo code.

Algorithm 5 Particle swarm optimisation algorithm

```
1: Objective function $F(x)$, $x = (x_1, \dots, x_d)^T$
2: Initialise: *number of iteration* (*MaxIter*), *number of particles* ($n$), *dimension* ($d$) and *maximum velocity* ($V_{max}$)
3: for $s \leftarrow 1$ to $n$ do
4:     Generate random *position* ($x_d$) and *velocity* ($v_d$)
5:     Evaluate the *fitness* ($F(x)$) for each particle $x_d$ and $v_d$
6: end for
7: Set the $F(x)$ as *pbest* for each particle
8: Set the *min* $F(x)$ as *gbest* for the swarm
9: while $t \leq MaxIter$ do
10:     Define the *inertia weight* ($w$) (Equation 6.5)
11:     Generate new $v_d$ and $x_d$ of each particles (Equation 6.4)
12:     Evaluate the $F(x)$ for each particle $v_d$ and $x_d$
13:     if $F(x) \leq pbest$ then
14:         Assign $F(x)$ as new *pbest* and its position as new *pbest* position
15:     else
16:         Remain the previous *pbest* and its position
17:     end if
18:     if *min* $(F(x)) \leq gbest$ then
19:         Assign *min* $(F(x))$ as new *gbest* and its position as new *gbest* position
20:     else
21:         Remain the previous *gbest* and its position
22:     end if
23: end while
24: Declare the *gbest* as optimum fitness evaluated and its position as optimum value(s)
```

6.3 A dual-particle swarm optimisation-modified adaptive bats sonar algorithm

MABSA was researched in chapter 5 as a combination of ABSA and a reformulated version of the original BSA to solve constrained optimisation problems. A hybridisation between the MABSA and PSO algorithm is considered in this section. The purpose of the hybrid algorithm is to solve multi objective optimisation problems.

A dual level search strategy is adopted through integration of the two algorithms for getting the Pareto optimum set of the problem considered. A pseudo-code of the algorithm is shown as Algorithm 6. This hybrid algorithm is named dual-particle swarm optimisation-modified adaptive bats sonar algorithm (D-PSO-MABSA). The D-PSO-MABSA algorithm uses the weighted sum approach to combine all objectives into a single objective. The weights are generated randomly from a uniform distribution. By doing so, the Pareto optimum set can be acquired efficiently as well as the Pareto front would be estimated appropriately.

Here, the dual level searching process means that at every time to obtain one Pareto optimum point, there are always two levels of search. During the first level, PSO acts as a global search agent of the algorithm with its embedded global (exploration) and local (exploitation) search components. As an explorer, the PSO is first to discover and mark a potential location of a solution in the compound of designated search space. The PSO will run according to its standard algorithmic procedures such as locating new velocity and position to obtain the *pbest* and *gbest*.

Algorithm 6 Dual-particle swarm optimisation-modified adaptive bats sonar algorithm

```
1: Objective function $F(x) = [F_1(x), F_2(x) \ldots, F_N(x)]^T$,    $x = (x_1, \ldots, x_d)^T$
2: Initialise: $Bats$, $MaxIter$, $Dim$, $SS_{Size}$, $NBeam_{MAX}$, $NBeam_{MIN}$, $n$, $V_{max}$ and $d$
3: for $j \leftarrow 1$ to $N$ (points on Pareto set) do
4:     Generate $K$ weights ($w_k \geq 0$) to form (Equation 2.6)
5:     for $s \leftarrow 1$ to $n$ do
6:         Generate random $position$ ($x_d$) and $velocity$ ($v_d$)
7:         Evaluate the $fitness$ ($F(x)$) for each particle $x_d$ and $v_d$
8:     end for
9:     Set the $F(x)$ as $pbest$ for each particle
10:    Set the $min\ F(x)$ as $gbest$ for the swarm
11:    while $t \leq MaxIter$ do
12:        Define the $inertia\ weight$ ($w$) (Equation 6.5)
13:        Generate new $v_d$ and $x_d$ of each particles (Equation 6.4)
14:        Evaluate the $F(x)$ for each particle $v_d$ and $x_d$
15:        if $F(x) \leq pbest$ then
16:            Assign $F(x)$ as new $pbest$ and its position as new $pbest$ position
17:        else
18:            Remain the previous $pbest$ and its position
19:        end if
20:        if $min\ (F(x)) \leq gbest$ then
21:            Assign $min\ (F(x))$ as new $gbest$ and its position as new $gbest$ position
22:        else
23:            Remain the previous $gbest$ and its position
24:        end if
25:    end while
26:    Assign $pbest$ as $F_{SP}$; its position as $pos_{SP}$ and $gbest$ as $F_{GB}$; its position as $pos_{GB}$
27:    while $t \leq MaxIter$ do
28:        Define $NBeam$ to transmit by using $BNI$ (Equation 4.4 and Equation 4.5)
29:        for $n \leftarrow 1$ to $Bats$ do
30:            for $N \leftarrow 1$ to $NBeam$ do
31:                for $d \leftarrow 1$ to $Dim$ do
32:                    Set $L$ and limit $\mu$ (Equation 5.1 and Equation 4.3)
33:                end for
34:            end for
35:            Generate random $\theta_m$ and $\theta$ (Equation 4.6)
36:            Transmit $NBeam$ starting from $pos_{SP}$
37:            for $N \leftarrow 1$ to $NBeam$ do
38:                for $d \leftarrow 1$ to $Dim$ do
39:                    Determine $pos_i$ for each transmitted beam (Equation 5.2)
40:                    Verify $pos_i$ for each transmitted beam within $SS_{Size}$
41:                    if $pos_i \geq SS_{Max}$ then
42:                        Update $pos_i$ (Equation 5.3a)
43:                    end if
44:                    if $pos_i \leq SS_{Min}$ then
45:                        Update $pos_i$ (Equation 5.3b)
46:                    end if
47:                end for
48:                Evaluate $F_i$ value for $F(pos_i)$
49:                Assign the optimum value of $F_i$ as $F_{LB}$ and its position as $pos_{LB}$
```

Algorithm 6 Dual-particle swarm optimisation-modified adaptive bats sonar algorithm - cont.

```
50:             if $F_{LB} \leq F_{SP}$ then
51:                 Assign $F_{LB}$ as $F_{RB}$ and $pos_{LB}$ as $pos_{RB}$
52:             else
53:                 Assign $F_{SP}$ as $F_{RB}$ and $pos_{SP}$ as $pos_{RB}$
54:             end if
55:         end for
56:     end for
57:     Select the optimum value among $F_{RB}$ as current $F_{GB}$ and its $pos_{RB}$ as current $pos_{GB}$
58:     if current $F_{GB} \leq$ previous $F_{GB}$ then
59:         Update current $F_{GB}$ as new $F_{GB}$ and current $pos_{GB}$ as new $pos_{GB}$
60:     else
61:         Retain previous $F_{GB}$ and $pos_{GB}$
62:     end if
63:     for $n \leftarrow 1$ to $Bats$ do
64:         Determine new $pos_{SP}$ using (Equation 4.8)
65:         Evaluate new $F_{SP}$ value for $F(pos_{SP})$
66:     end for
67:   end while
68: end for
69: Declare $F_{GB}$ as optimum fitness evaluated and $pos_{GB}$ as its optimum value(s)
```

In the second level search process, the optimum solutions obtained by the PSO are used to initialise the starting positions of the population in the MABSA. The MABSA is considered as a local search agent of the algorithm and also has its global search (diversification component) and local search (intensification component). Here, MABSA works as a follower to find the optimum solutions starting from the prospective location previously marked by the PSO within the designated search space.

The MABSA first sets the number of individuals in the population randomly between 700-1000 bats at every iteration. The value has been inspired from the real population of bats in a colony. Then, PSO will follow suit although the standard PSO algorithm has 100-200 number of particles only. The equivalence of population size between PSO and MABSA is crucial to a smooth phase transition of the final solution found by the PSO and inherited by MABSA during the algorithm runs. Thus, the population size criterion will act as a look-alike handshaking or acknowledgement procedure of the dual level search process.

MABSA proceeds through its normal search procedure in transmitting the sound beams by bats into the dedicated search space to get pos_{LB} and F_{LB} and finally pos_{RB} and F_{RB}. This operation runs until the specified maximum iterations. As in the original MABSA, the pos_{GB} with its F_{GB} resulting from the overall iterations will be declared as the best optimum solution to the problem studied. Thus, the optimum solution obtained is considered as one Pareto optimum point. The algorithm will repeatedly run until the total number of Pareto optimum points are obtained to get a complete set of Pareto.

There are two factors to be considered to set PSO as global search agent and MABSA as local search agent. These factors are inspired by the real behaviour of both swarm groups. As noted, PSO is represented based on a swarm of birds flying in search of food while MABSA is based on a colony of bats flying for capturing preys. The factors are swarm flight attitude and swarm searching strategy.

The first factor is the flight attitude of the swarm. A good global search agent has a capability of viewing and monitoring the search space from the highest position. The broad perspective from the higher ground makes it easier for the agent to mark possible areas within the search space containing potential solutions that would be a true exploration process in swarm intelligence. A local search agent is, on the other hand, needed to verify the location of potential solutions found by a global search agent. To be a good local search agent, the agent must have the ability to observe and inspect the solutions from a close view. This exploitation process should be put after the exploration process so that the solutions explored by a global agent could be validated properly by the local search agent. In reality, the bar-headed goose that is a family of birds can fly to the highest point up to 6437m. Meanwhile, bats only fly less than 10m above the sea level. These facts have enthused PSO to be defined as global search agent while MABSA as local search agent.

Looking at the proposed swarm searching strategy, there is a distinct line between the searching strategy of PSO and MABSA. In the PSO, the algorithm utilises the velocity and positioning of particles to evaluate the obtained solution whereas MABSA depends on the transmission and positioning of sound beams. In the real world, birds can fly with a velocity between 20 to 30 mph. With this fast speed, the searching process of PSO may miss the locations of good solutions on their way towards other possible target solutions. Moreover, the velocity of particles in PSO itself makes the particle or bird to move in a single line thus not covering a broad search area at one time. The sound beams transmitted in MABSA are multi line that are able to disperse and sweep a large search envelope. Thus, the issue of missing good solutions in a smaller area of designated search space does not arise. Hence, the sequence of searching process as applied in any good swarm intelligence method is followed here where coarse searching (diversification) is done first by PSO followed by fine searching (intensification) by MABSA. In this context, labelling PSO as global search agent and MABSA as local search agent in the proposed hybrid algorithm D-PSO-MABSA is a reasonable choice given their characteristics.

6.4 Computer simulation and discussion

6.4.1 Introduction

The computer simulation is divided into two parts. The first part is to demonstrate the performance of the D-PSO-MABSA on eight established multi objective benchmark test functions. The test functions are Zitzler-Deb-Thiele's function (ZDT) 1, Scheffer function 1, Binh and Korn function, Chankong and Haimes function, Kursawe function, Osyczka and Kundu function, Constr-Ex function as well as CTP1 function. Some of the test functions have constraints inside.

However, with exception to Zitzler-Deb-Thiele's function (ZDT) 1 and Scheffer function 1, the computer simulation for other six benchmark test functions have been extended to study the parameters used in D-PSO-MABSA. Variable values of position adaptability factor (α) and collision factor (β) of MABSA component of D-PSO-MABSA are used (including theoretical values from the prior chapter) in this test. Other parameters remain the same, and the standard parameters discussed in the earlier section are adopted in the PSO component. Both α and β parameters are chosen because they have a major influence on the search process of bats in a colony. If both factors are properly controlled, the overall algorithm will be able to produce significant results for any problem handled. However, the sample study presented in this work is aimed to demonstrate that the theoretical MABSA parameter values as elaborated in the previous chapter are the best choices to be used in the D-PSO-MABSA algorithm.

The second part is to test the performance of the D-PSO-MABSA algorithm on an engineering design problem. A four bar plane truss problem is selected as a platform for the algorithm. The problem is run for several different suit of Pareto points.

The computer simulation involved the multi objective optimisation benchmark test functions and an engineering problem that consist of only two objective functions but all these test functions have various difficulties. However, these test functions could simply be used to investigate and monitor the performance of the D-PSO-MABSA to form Pareto front of the well-represented set of Pareto optimum solutions. If the performance of D-PSO-MABSA going to suffer, it gets easy to analyse and launch the algorithm improvement plan. By using a bottom-up approach, the D-PSO-MABSA also expected to perform on the multi objective optimisation problem with more than two objective functions or on many objective optimisation. The reason is the algorithm procedure remain similar but only the number of objective functions involved will increase. A validation work toward this is allocated for the future research work.

6.4.2 Performance of D-PSO-MABSA on established multi objective benchmark test functions

Zitzler-Deb-Thiele's function (ZDT 1)

This function was among the well-known benchmark test functions used to evaluate an algorithm for solving the multi objective optimisation problem. The function constitutes an unconstrained problem and has a convex Pareto front. The function is defined as:

Minimise

$$F_1(x) = x_1$$

and

$$F_2(x) = \left(1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i\right)\left(1 - \sqrt{\frac{F_1}{\left(1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i\right)}}\right) \tag{6.6}$$

where

$$0 \le x_i \le 1$$

$$1 \le i \le 20$$

Table 6.1 shows 15 Pareto optimum point tabulated in terms of F_1 and F_2. The values of w_1 and w_2 are recorded to show linear increasing and decreasing in weighted sum values respectively. The search for each single Pareto optimum point was conducted over 100 iterations of D-PSO-MABSA algorithm. Figure 6.1 shows the Pareto optimum set of ZDT 1 function. It is noted that the proposed algorithm achieved a set of Pareto optimum points each comprising a non-dominated solution. Moreover, the set of non-dominated solutions successfully formed convex Pareto front as expected.

Table 6.1: ZDT 1 function test results

w_1	w_2	F_1	F_2
0.0667	0.9333	1.0000	0.0000
0.1333	0.8667	0.9999	0.0000
0.2000	0.8000	0.9999	0.0000
0.2667	0.7333	1.0000	0.0000
0.3333	0.6667	1.0000	0.0000
0.4000	0.6000	0.5625	0.2500
0.4667	0.5333	0.3265	0.4286
0.5333	0.4667	0.1914	0.5625
0.6000	0.4000	0.1110	0.6668
0.6667	0.3333	0.0625	0.7500
0.7333	0.2667	0.0330	0.8183
0.8000	0.2000	0.0156	0.8750
0.8667	0.1333	0.0059	0.9229
0.9333	0.0667	0.0012	0.9652
1.0000	0.0000	0.0003	0.9838

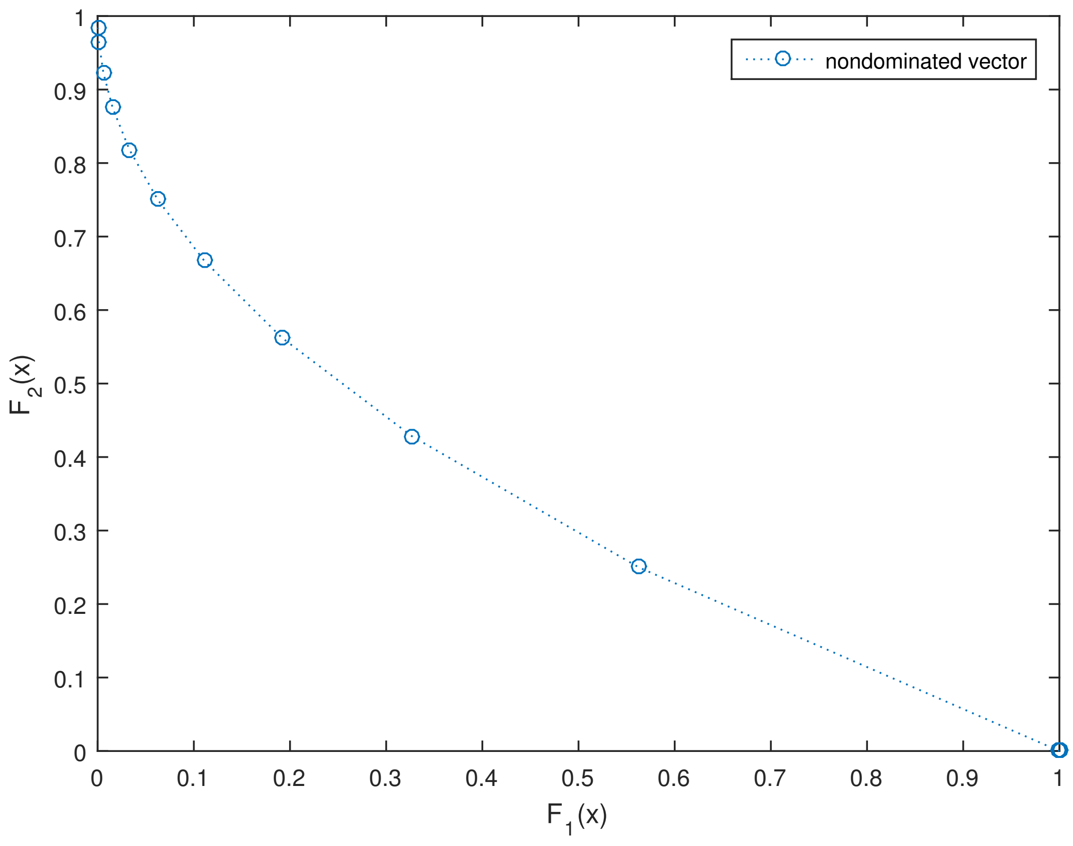


Figure 6.1: Pareto front for ZDT 1 function

Schaffer function 1

This function has been used to evaluate the Pareto archived evaluation strategy (PAES) in solving the multi objective optimisation problem. The function constitutes an unconstrained problem, has a convex Pareto front and is defined as:

Minimise

$$F_1(x) = x^2$$

and

$$F_2(x) = (x-2)^2 \tag{6.7}$$

where

$$-10 \leq x_i \leq 10$$

$$1 \leq i \leq 20$$

In this case study, the D-PSO-MABSA is applied to find 30 Pareto optimum points. Table 6.2 shows the results of F_1 and F_2 after using the values of w_1 and w_2 accordingly. The algorithm was run over 100 iterations for the search of each Pareto optimum point.

As noted in Figure 6.2, the algorithm performed well with the Scheffer function 1; the Pareto optimum points obtained were non-dominated solutions and formed a smooth Pareto front. Thus, the results thus obtained match those reported particularly when considering the values of both objective functions F_1 and F_2 as shown in Figure 6.3.

Table 6.2: Schaffer function 1 function test results

w_1	w_2	F_1	F_2	w_1	w_2	F_1	F_2
0.0333	0.9667	3.7357	0.0045	0.5333	0.4667	0.8709	1.1380
0.0667	0.9333	3.4837	0.0178	0.5667	0.4333	0.7511	1.2845
0.1000	0.9000	3.2401	0.0400	0.6000	0.4000	0.6402	1.4397
0.1333	0.8667	3.0054	0.0710	0.6333	0.3667	0.5361	1.6074
0.1667	0.8333	2.7762	0.1114	0.6667	0.3333	0.4454	1.7759
0.2000	0.8000	2.5624	0.1594	0.7000	0.3000	0.3592	1.9618
0.2333	0.7667	2.3514	0.2177	0.7333	0.2667	0.2847	2.1505
0.2667	0.7333	2.1495	0.2850	0.7667	0.2333	0.2154	2.3589
0.3000	0.7000	1.9604	0.3598	0.8000	0.2000	0.1603	2.5588
0.3333	0.6667	1.7755	0.4456	0.8333	0.1667	0.1130	2.7685
0.3667	0.6333	1.6085	0.5355	0.8667	0.1333	0.0712	3.0042
0.4000	0.6000	1.4365	0.6423	0.9000	0.1000	0.0407	3.2338
0.4333	0.5667	1.2856	0.7502	0.9333	0.0667	0.0183	3.4771
0.4667	0.5333	1.1383	0.8706	0.9667	0.0333	0.0046	3.7341
0.5000	0.5000	1.0000	1.0000	1.0000	0.0000	0.0000	3.9985

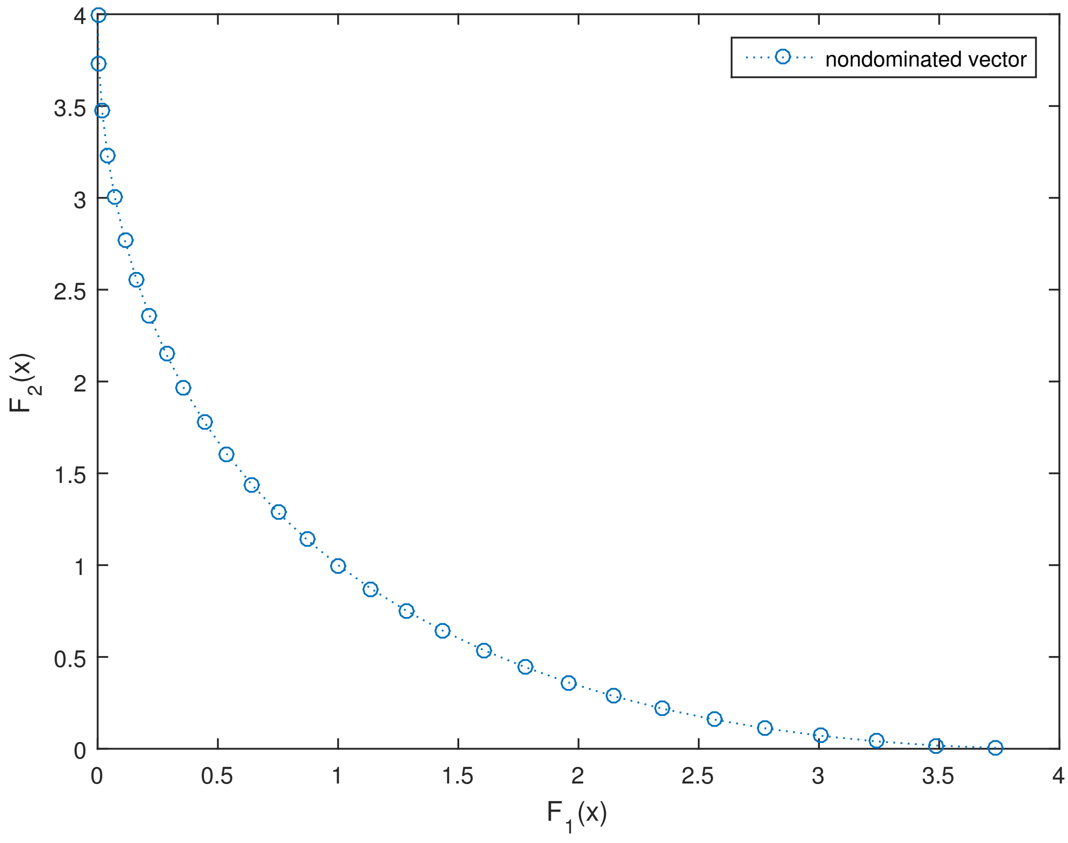


Figure 6.2: Pareto front for Schaffer function 1

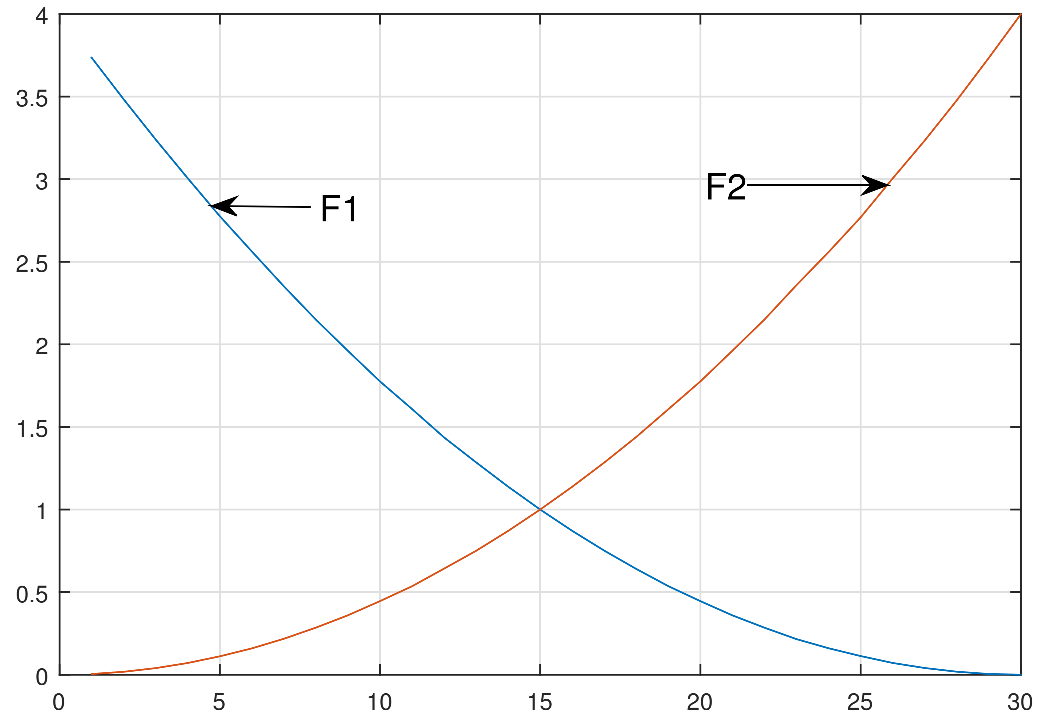


Figure 6.3: Plot of separated F_1 and F_2 of Schaffer function 1

Binh and Korn function

This is the function presented previously to test the multi objective evolutionary strategy (MOBES) for multi objective optimisation problem with constraints. The function constitutes a constrained problem and has a convex Pareto front. The function is defined as:

$$
\begin{aligned}
&\text{Minimise}\\
&\quad F_1(x) = 4x_1^2 + 4x_2^2\\
&\quad \text{and}\\
&\quad F_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2\\
&\text{subject to}\\
&\quad g_1(x) = (x_1 - 5)^2 + x_2^2 \leq 25\\
&\quad g_2(x) = (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7\\
&\text{where}\\
&\quad 0 \leq x_1 \leq 5\\
&\quad 0 \leq x_2 \leq 3
\end{aligned}
\tag{6.8}
$$

The D-PSO-MABSA algorithm will determine sets of 50 Pareto optimum points for this test function by using four different combinations of α and β values respectively. The values are: $\alpha = 0.00$; $\beta = 3.50$, $\alpha = 0.00$; $\beta = 0.00$, $\alpha = 2.50$; $\beta = 0.00$, along with the theoretical values $\alpha = \alpha 1$; $\beta = \beta 1$ (here $\alpha 1$ and $\beta 1$ are two numbers between 0 and 1).

Figure 6.4 shows results of Pareto optimum points recorded of Binh and Korn function using four different settings of α and β of the D-PSO-MABSA algorithm. As noted, the algorithm was able to converge with each setting to a Pareto front of the test function that was similar to the original results recorded. However, in general, by using theoretical values; $\alpha = \alpha 1$ and $\beta = \beta 1$, all the points of the Pareto optimum set attained are non-dominated vectors. Thus, these solutions perfectly formed a recognisable Pareto front. The stability of final location of non-dominated solutions acquired by D-PSO-MABSA algorithm through theoretical α and β settings show a high prospect of the D-PSO-MABSA to solve any multi objective optimisation problem.

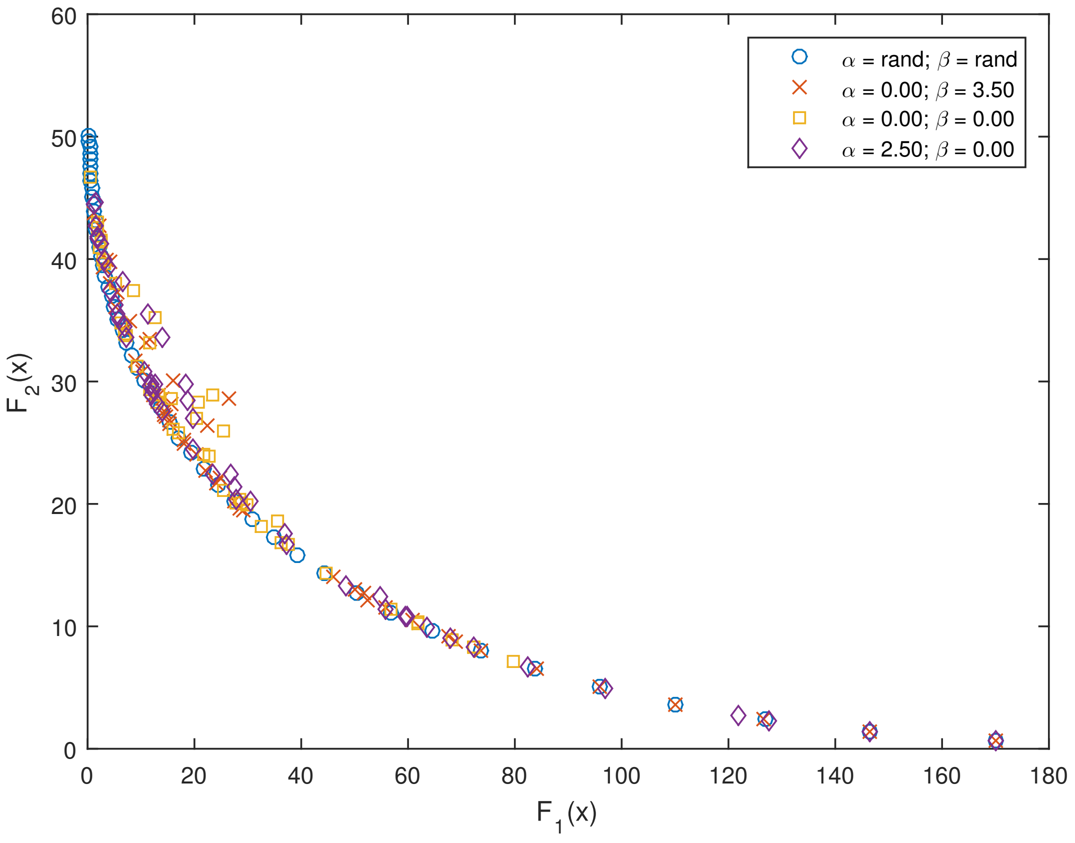


Figure 6.4: Pareto optimum solutions for Binh and Korn function with different values of α and β

Chankong and Haimes function

The function was introduced by Chankong and Haimes in 1983 and was named after them. The function constitutes a constrained problem and has a convex Pareto front. The function is defined as:

$$
\begin{aligned}
&\text{Minimise} \\
&\quad F_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2 \\
&\quad \text{and} \\
&\quad F_2(x) = 9x_1 - (x_2 - 1)^2 \\
&\text{subject to} \\
&\quad g_1(x) = x_1^2 + x_2^2 \leq 225 \\
&\quad g_2(x) = x_1 - 3x_2 + 10 \leq 0 \\
&\text{where} \\
&\quad -20 \leq x_1, x_2 \leq 20
\end{aligned}
\tag{6.9}
$$

For this test function, four set of 50 Pareto optimum points are searched. The D-PSO-MABSA algorithm operated on three different sets of α and β values in conjunction with the theoretical values; $\alpha = \alpha 2$; $\beta = \beta 2$ (here $\alpha 2$ and $\beta 2$ are two numbers between 0 and 1). These three sets considered were: $\alpha = 0.00$; $\beta = 3.50$, $\alpha = 0.00$; $\beta = 0.00$, $\alpha = 2.50$; $\beta = 0.00$.

Figure 6.5 shows the Pareto optimum sets with different values of α and β. A Pareto front is properly drawn by a set of non-dominated solutions acquired by the theoretical value of $\alpha = \alpha 2$ and $\beta = \beta 2$. The result is comparable to the result acquired by previous researchers. Even the remaining sets of α and β managed to search the points that settle on the Pareto front, but there were still, few dominated solutions scattered far from the true front. Thus, it is shown that the D-PSO-MABSA algorithm with theoretical parameter values was able to achieve a perfect Pareto front with this test function from the set of Pareto optimum points attained. This performance makes the D-PSO-MABSA algorithm at par with other multi objective optimisation algorithms and may be used widely to solve any multi objective optimisation problems.

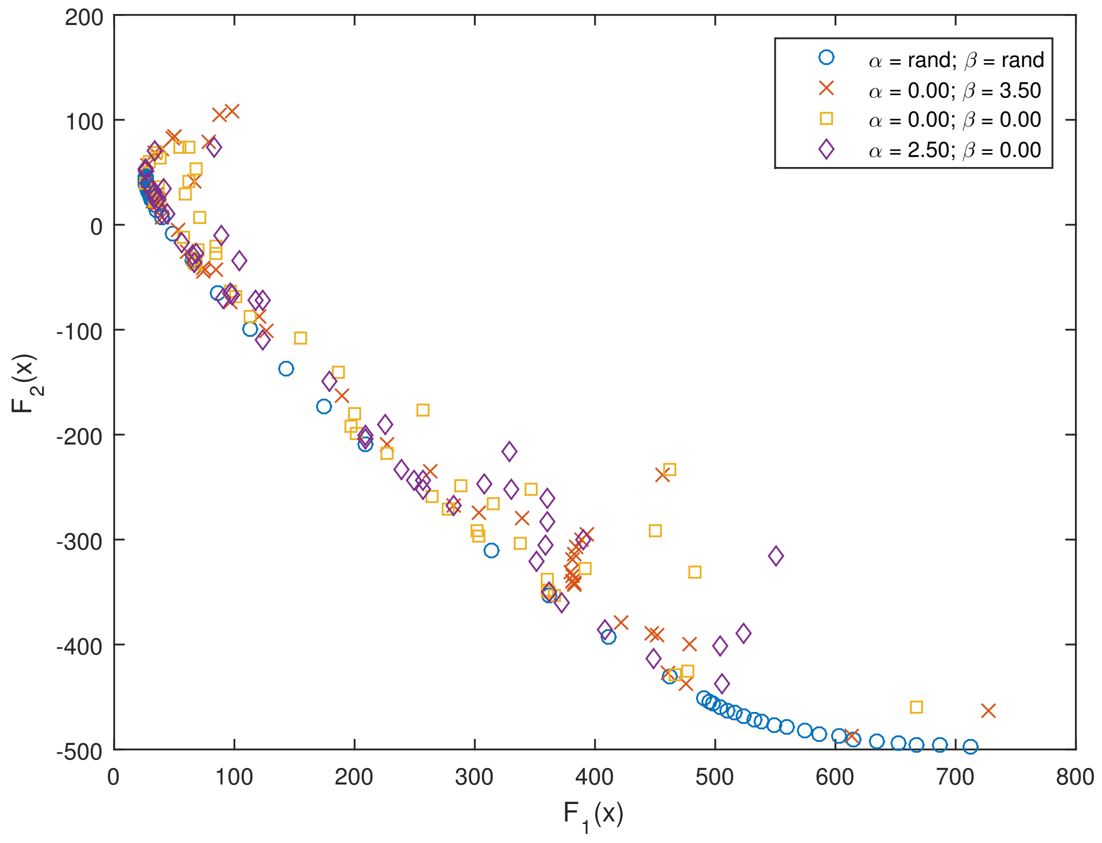


Figure 6.5: Pareto optimum solutions for Chankong and Haimes function with different values of α and β

Kursawe function

This function is a multimodal function in one component and has pair-wise interactions among the variables in the other component. The function constitutes an unconstrained problem and has a discrete convex Pareto front. The function is defined as:

Minimise

$$F_1(x) = \sum_{i=1}^{2} \left[-10e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} \right]$$

and

$$F_2(x) = \sum_{i=1}^{3} \left[|x_i|^{0.8} + 5sinx_i^3 \right] \tag{6.10}$$

where

$$-5 \leq x_1 \leq 5$$

$$1 \leq x_2 \leq 3$$

This test function involved searching of four sets of 200 Pareto optimum solutions with the D-PSO-MABSA algorithm. The theoretical values of $\alpha = \alpha 3$; $\beta = \beta 3$ (here $\alpha 3$ and $\beta 3$ are two numbers between 0 and 1) were adopted along with another three sets of α and β for performance comparison purpose. The three sets used were: $\alpha = 2.00$; $\beta = 4.00$, $\alpha = 3.00$; $\beta = 2.00$, $\alpha = -2.00$; $\beta = -2.00$.

Figure 6.6 shows the Pareto optimum sets obtained for Kursawe function using D-PSO-MABSA approach. As noted, the algorithm with theoretical values of $\alpha = \alpha 3$ and $\beta = \beta 3$ achieved the best performance compared to when the other three sets of α and β were used. Most of the points in the Pareto optimum set were non-dominated solutions, successfully exhibiting a Pareto front of the test function. The pattern of Pareto front with the three discontinuous regions also developed nearly a matched result that was obtained by previous researchers. With the remaining three sets of α and β the algorithm could not form a Pareto front of this test function, and only a few of the solutions were non-dominated. The Pareto optimum point generated from the set $\alpha = 3.00$; $\beta = 2.00$ is likely to work, but most of the points with this set are dominated solutions and scattered far from the true front. As far as the values of α and β are concerned, negative values do not lead to a Pareto front. When set of $\alpha = -2.00$ and $\beta = -2.00$ was applied, no non-dominated solutions were achieved. Nonetheless, the D-PSO-MABSA algorithm with the right setting of its parameters would be good alternative multi objective algorithm for solving discrete convex Pareto front-type multi objective optimisation problems.

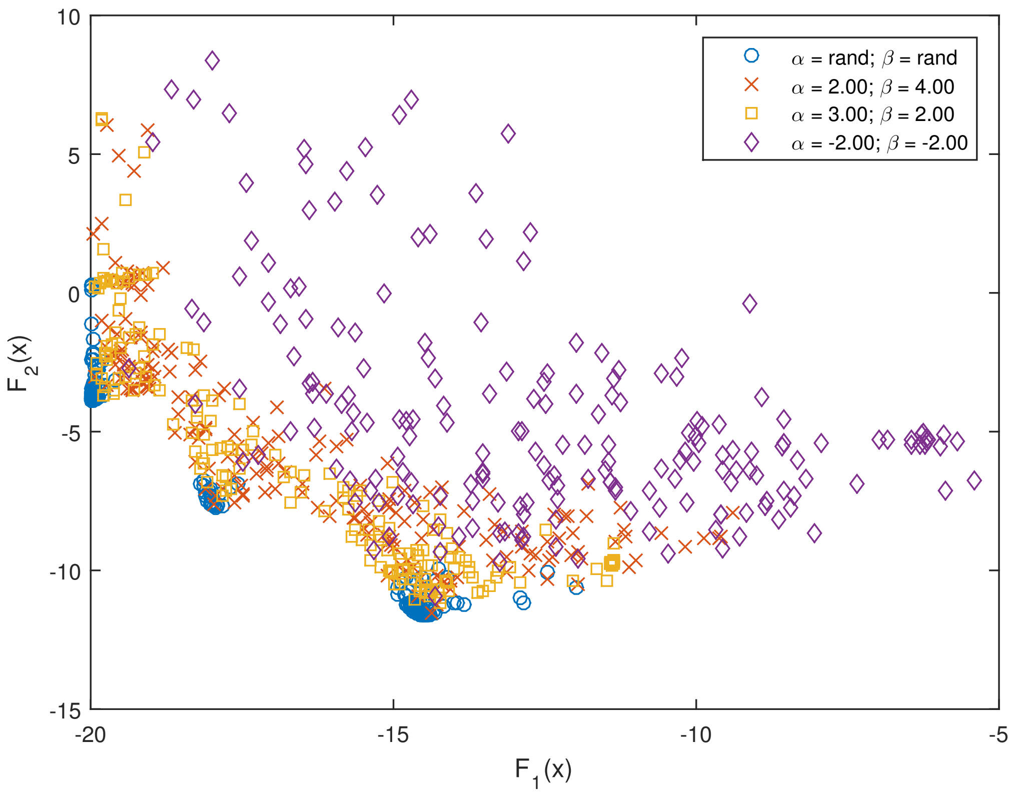


Figure 6.6: Pareto optimum solutions for Kursawe function with different values of α and β

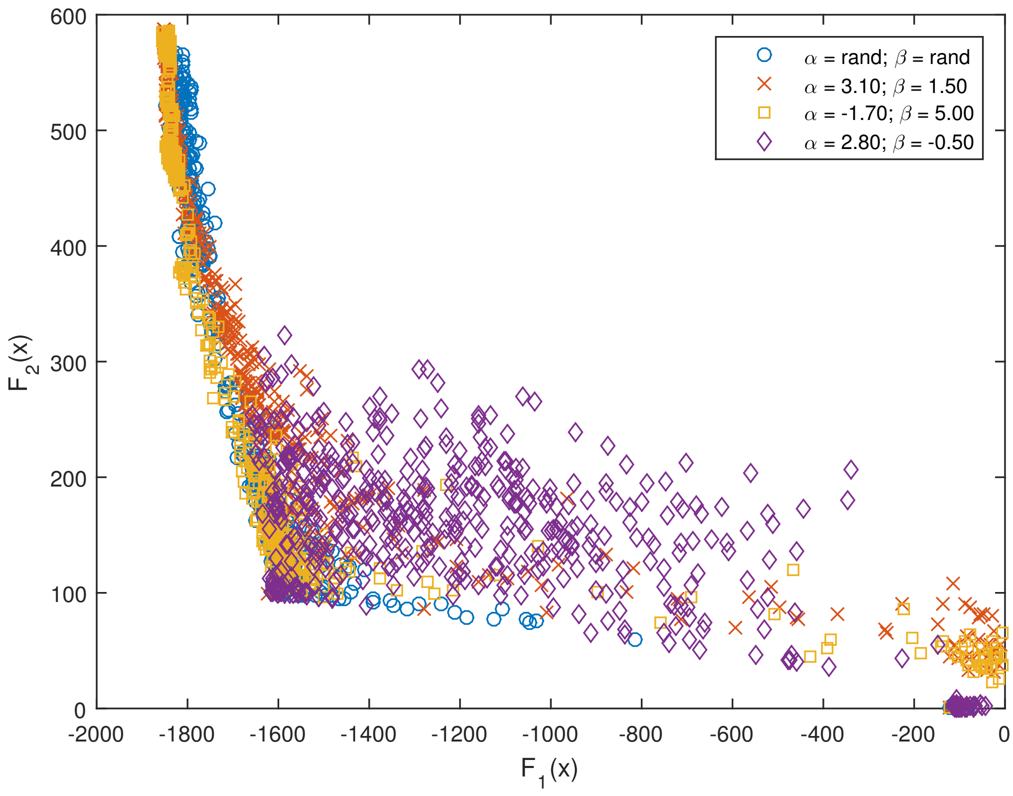


Figure 6.7: Pareto optimum solutions for Osyczka and Kundu function with different values of α and β

Osyczka and Kundu function

The function constitutes a constrained problem and has a convex Pareto front. The function is defined as:

$$
\begin{aligned}
&\text{Minimise} \\
&\qquad F_1(x) = -25(x_1-2)^2 + (x_2-2)^2 - (x_3-1)^2 - (x_4-4)^2 - (x_5-1)^2 \\
&\qquad \text{and} \\
&\qquad F_2(x) = \sum_{i=1}^{6} x_i^2 \\
&\text{subject to} \\
&\qquad g_1(x) = x_1 + x_2 - 2 \geq 0 \\
&\qquad g_2(x) = 6 - x_1 - x_2 \geq 0 \\
&\qquad g_3(x) = 2 - x_2 + x_1 \geq 0 \\
&\qquad g_4(x) = 2 - x_1 + 3x_2 \geq 0 \\
&\qquad g_5(x) = 4 - (x_3-3)^2 - x_4 \geq 0 \\
&\qquad g_6(x) = (x_5-3)^2 + x_6 - 4 \geq 0 \\
&\text{where} \\
&\qquad 0 \leq x_1, x_2, x_6 \leq 10 \\
&\qquad 1 \leq x_3, x_5 \leq 5 \\
&\qquad 0 \leq x_4 \leq 6
\end{aligned}
\tag{6.11}
$$

For this test function, four sets of 500 Pareto optimum points are searched by using the D-PSO-MABSA algorithm. Each set is examined by different value of α and β. The theoretical values $\alpha = \alpha 4$; $\beta = \beta 4$ (here $\alpha 4$ and $\beta 4$ are two numbers between 0 and 1) were applied along with the three sets $\alpha = 3.10$; $\beta = 1.50$, $\alpha = -1.70$; $\beta = 5.00$, $\alpha = 2.80$; $\beta = -0.50$.

Figure 6.7 shows the effect of different values of α and β on the Pareto optimum solutions of Osyczka and Kundu function. When the theoretical values of $\alpha = \alpha 4$ and $\beta = \beta 4$ were used, all the Pareto optimum points were non-dominated vectors. Although the ranges for F_1 and F_2 recorded were wider than the result reported by previous researchers, the shapes of the Pareto front were nearly similar as all the Pareto optimum points contributed to form that front.

In the meantime, the three sets of α and β produced many dominated vectors of Pareto optimum sets thus unable to form a viable Pareto front. Indeed, the Pareto optimum set gathered by $\alpha = 2.80$; $\beta = -0.50$ was more obvious as the points were scattered outlying from the true front. However, if the theoretical values of α and β are retained by the D-PSO-MABSA, the algorithm will be able to perform well in comparison to available algorithms in solving multi objective optimisation problems.

Constr-Ex function

This function was designed by Deb in 2001 as a multi objective benchmark test function. The function constitutes a constrained problem and has a convex Pareto front. The function is defined as:

$$
\begin{aligned}
&\text{Minimise}\\
&\qquad F_1(x) = x_1\\
&\qquad \text{and}\\
&\qquad F_2(x) = \frac{1+x_2}{x_1}\\
&\text{subject to}\\
&\qquad g_1(x_1,x_2) = x_2 + 9x_1 \geq 6\\
&\qquad g_2(x_1,x_2) = -x_2 + 9x_1 \geq 1\\
&\text{where}\\
&\qquad 0.1 \leq x_1 \leq 1\\
&\qquad 0 \leq x_2 \leq 5
\end{aligned}
\tag{6.12}
$$

The D-PSO-MABSA algorithm was evaluated with this function by searching four sets of 50 Pareto optimum solutions. Here, four sets of different values of α and β were used. These included the theoretical values $\alpha = \alpha 5$; $\beta = \beta 5$ (here $\alpha 5$ and $\beta 5$ are two numbers between 0 and 1), $\alpha = -4.00$; $\beta = 3.00$, $\alpha = 0.00$; $\beta = -1.70$, $\alpha = 3.50$; $\beta = 3.50$.

As noted in Figure 6.8, all four sets of 50 Pareto optimum solutions generated from four different values of α and β of D-PSO-MABSA were non-dominated vectors. So, the entire sets produced a Pareto front similar to that results reported by previous researchers. It was noted that the convex shape of Pareto fronts produced by the D-PSO-MABSA algorithm was smoother than that reported. It is clear that the D-PSO-MABSA algorithm generates distinctly better Pareto optimum points in solving multi objective optimisation problems.

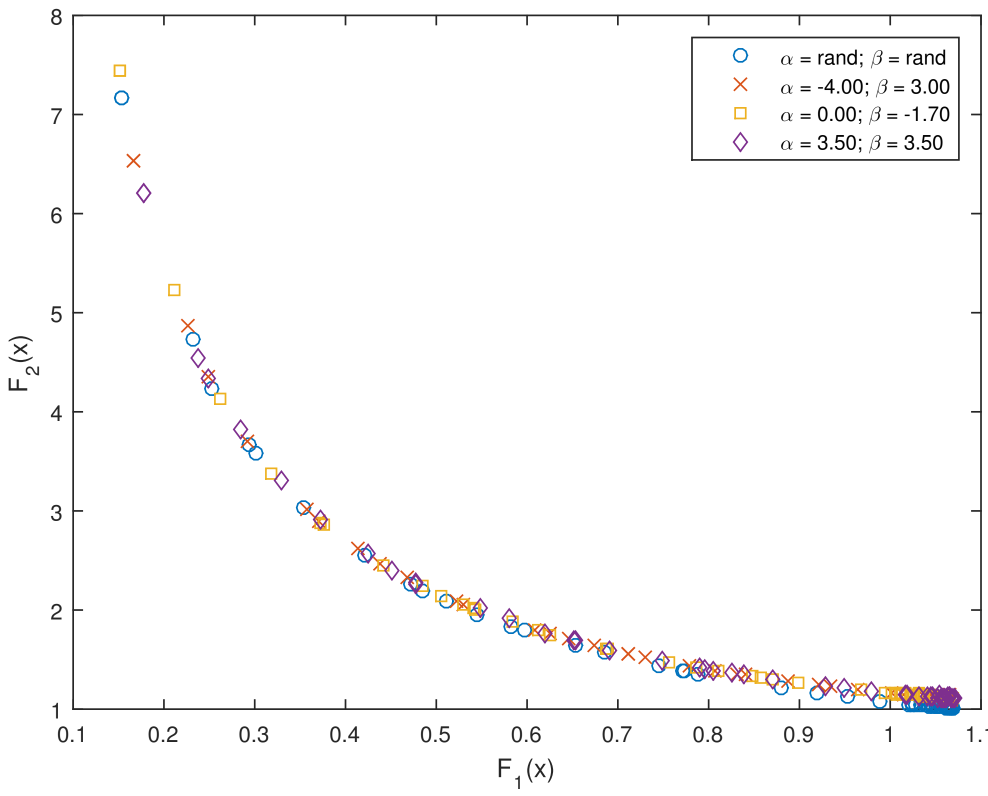


Figure 6.8: Pareto optimum solutions for Constr-Ex function with different values of α and β

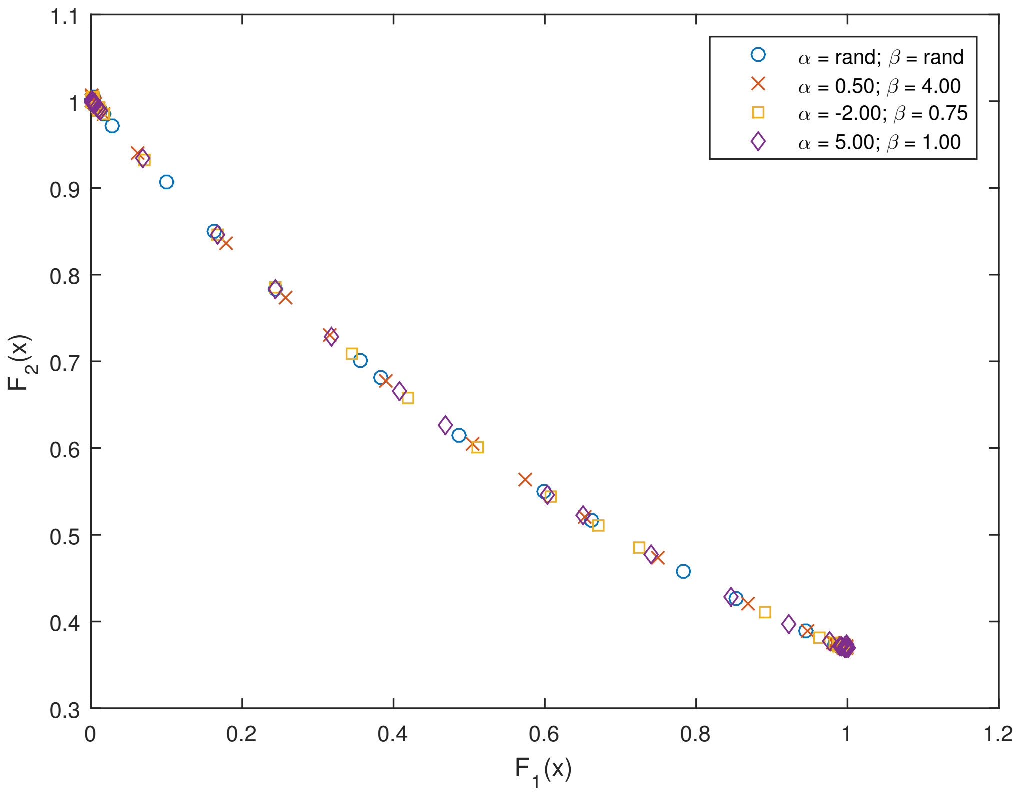


Figure 6.9: Pareto optimum solutions for CTP1 function with different values of α and β

CTP1 function

The function constitutes a constrained problem and has a convex Pareto front. The function is defined as:

$$
\begin{aligned}
&\text{Minimise}\\
&\qquad F_1(x) = x_1\\
&\qquad \text{and}\\
&\qquad F_2(x) = (1+x_2)e^{\left(-\frac{x_1}{1+x_2}\right)}\\
&\text{subject to}\\
&\qquad g_1(x) = \frac{F_2(x_1,x_2)}{0.858e^{(-0.541F_1(x_1,x_2))}} \geq 1\\
&\qquad g_2(x) = \frac{F_2(x_1,x_2)}{0.728e^{(-0.295F_1(x_1,x_2))}} \geq 1\\
&\text{where}\\
&\qquad 0 \leq x_1, x_2 \leq 1
\end{aligned}
\tag{6.13}
$$

Here, four sets of 50 Pareto optimum solutions are searched for CTP1 function using the D-PSO-MABSA algorithm. These were the theoretical values $\alpha = \alpha 6$; $\beta = \beta 6$ (here $\alpha 6$ and $\beta 6$ are two numbers between 0 and 1), $\alpha = 0.50$; $\beta = 4.00$, $\alpha = -2.00$; $\beta = 0.75$, $\alpha = 5.00$; $\beta = 1.00$.

The results of Pareto optimum solution for the CTP1 are shown in Figure 6.9. It is noted that all the solutions generated using D-PSO-MABSA algorithm with four different sets of α and β values were non-dominated vectors. The Pareto fronts formed from the solutions were identical to the result reported previously.

Furthermore, these also reflected the real advantage when using the set of theoretical values; $\alpha = \alpha 6$ and $\beta = \beta 6$ as the non-dominated solutions produced were uniformly distributed along the front. Hence, the outcomes resulted from a good leverage of minimising both F_1 and F_2 and none was extremely good while other suffered. The performances shown with the test functions demonstrate the strong ability of the D-PSO-MABSA algorithm in producing good trade-off solutions for multi objective optimisation problems.

6.4.3 Performance of D-PSO-MABSA in engineering design problem

A four bar plane truss problem

This multi objective engineering design problem was introduced by Stadler and Dauer in 1992. The problem is to design a four bar plane truss as shown in Figure 6.10. The design has two objectives, namely to minimise the volume of the truss (F_1) and at the same time to minimise its joint displacement (F_2). This can be expressed as:

Minimise

$$F_1(x) = L\left(2x_1 + \sqrt{2}x_2 + \sqrt{x_3} + x_4\right)$$

and

$$F_2(x) = \frac{FL}{E}\left(\frac{2}{x_1} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{2}}{x_3} + \frac{2}{x_4}\right)$$

subject to

$$\begin{aligned}
&(\frac{F}{\sigma}) \leq x_1 \leq 3(\frac{F}{\sigma})\\
&\sqrt{2}(\frac{F}{\sigma}) \leq x_2 \leq 3(\frac{F}{\sigma})\\
&\sqrt{2}(\frac{F}{\sigma}) \leq x_3 \leq 3(\frac{F}{\sigma})\\
&(\frac{F}{\sigma}) \leq x_4 \leq 3(\frac{F}{\sigma})
\end{aligned} \tag{6.14}$$

where

$$F = 10kN$$

$$E = 2 \times \frac{10^5 kN}{cm^2}$$

$$L = 200cm$$

$$\sigma = \frac{10kN}{cm^2}$$

It is expected that the non-dominated solutions forming the Pareto front will be as shown in Figure 6.11.

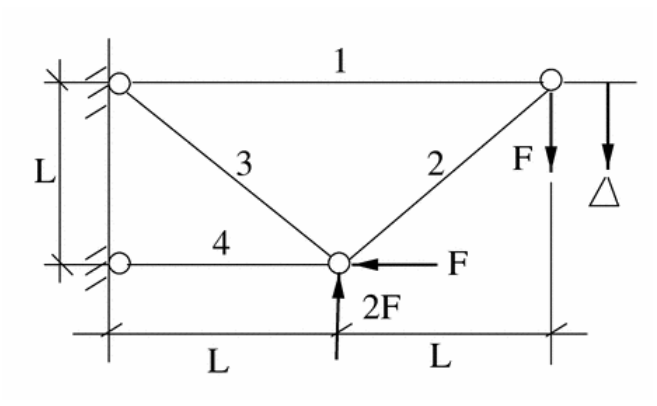


Figure 6.10: A four bar plane truss

To show the ability of the D-PSO-MABSA algorithm to find the trade-off solutions of the problem, five dissimilar number of Pareto optimum sets were used. The sets adopted were 40, 100, 500, 1000 and 4000. Figure 6.12, Figure 6.13, Figure 6.14, Figure 6.15 and Figure 6.16 show the results for the different number of Pareto optimum sets respectively.

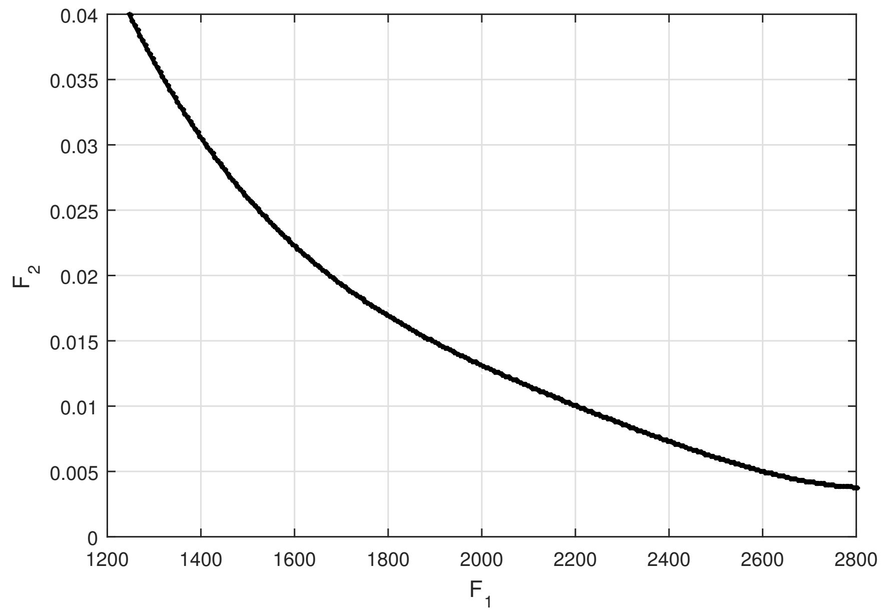


Figure 6.11: The true (or global) Pareto front of a four bar plane truss problem

Referring to the Figure 6.12, when a set of 40 Pareto points is used, there were four non-dominated solutions produced by the algorithm. These four points were non-dominated solutions that formed a Pareto front as a basis to relate to the two objectives studied. With the number of Pareto points increased to 100, as shown in Figure 6.13, there were seven non-dominated solutions forming the Pareto front approximately similar to that reported previously.

After the number of Pareto points had been increased to 500 and 1000 as in Figure 6.14 and Figure 6.15 respectively, both cases resulted in a few of non-dominated vectors besides the huge amount of dominated vectors. Nevertheless, these small groups of non-dominated vectors successfully resulted in a Pareto front that connected the true relationship between both objectives to minimise the volume and minimise joint displacement of the truss.

However, when 4000 Pareto points were considered as shown in Figure 6.16 the solutions concentrated more toward the centre of the designated search space. Here also, the non-dominated solutions appear to be significantly clearer. These non-dominated solutions formed a Pareto front similar to that reported previously. Indeed, the value of F_1 here was smaller as compared to the reference figure while the value of F_2 remained similar.

To conclude, the D-PSO-MABSA algorithm performed well to optimise the design of a four bar plane truss. The performance was shown by the ability of the D-PSO-MABSA algorithm to result in a Pareto front from non-dominated solutions with any number of Pareto optimum solution considered. These Pareto fronts provided good compromise solutions of minimising two different objectives named the volume and the joint displacement of the truss.

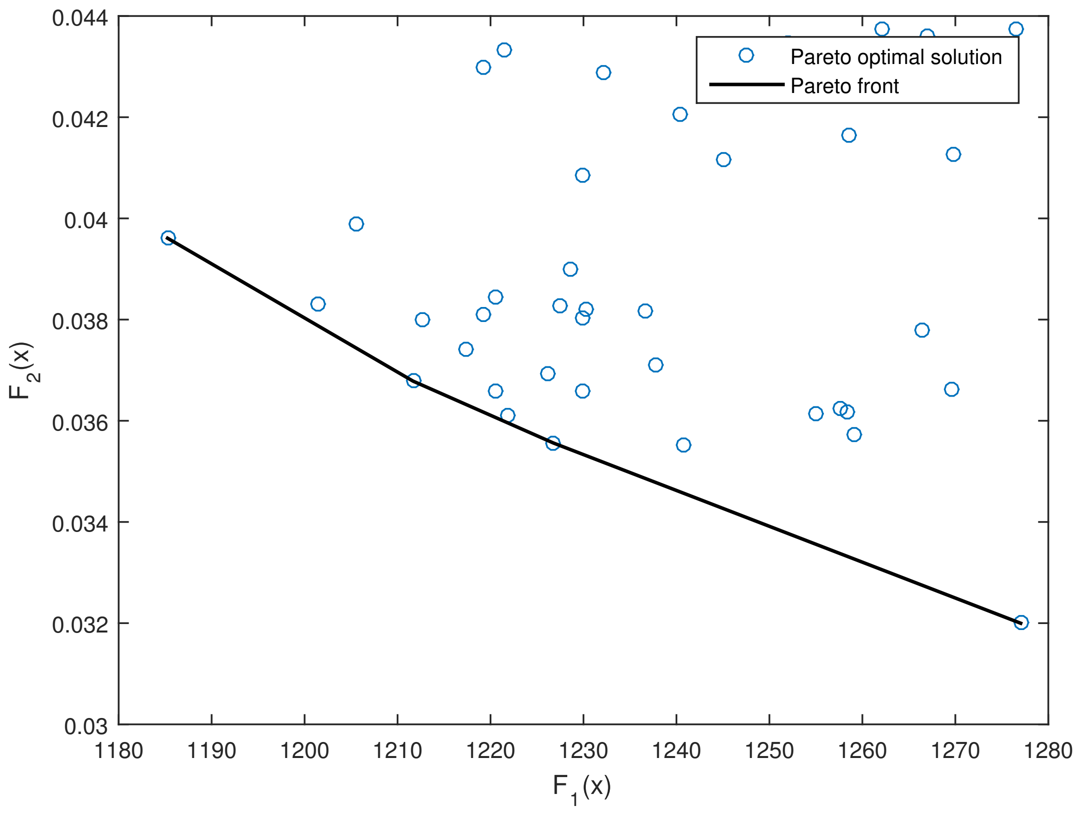


Figure 6.12: A four bar plane truss problem with 40 Pareto optimum solutions

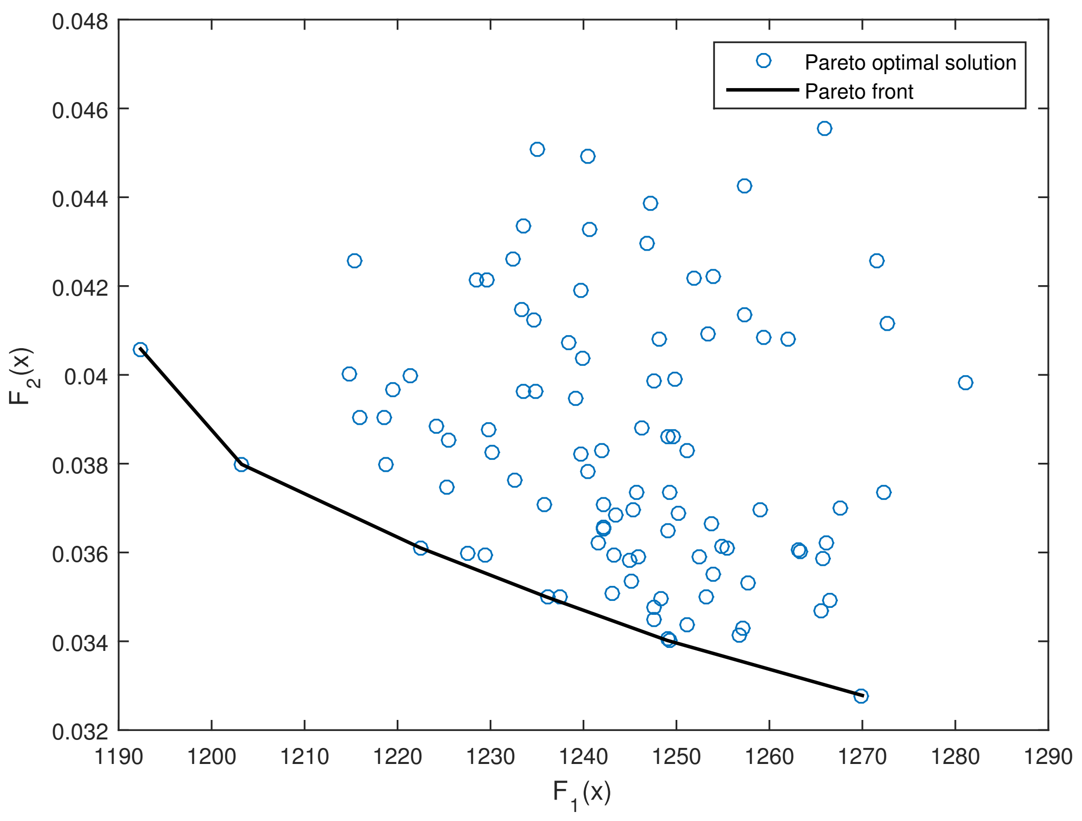


Figure 6.13: A four bar plane truss problem with 100 Pareto optimum solutions

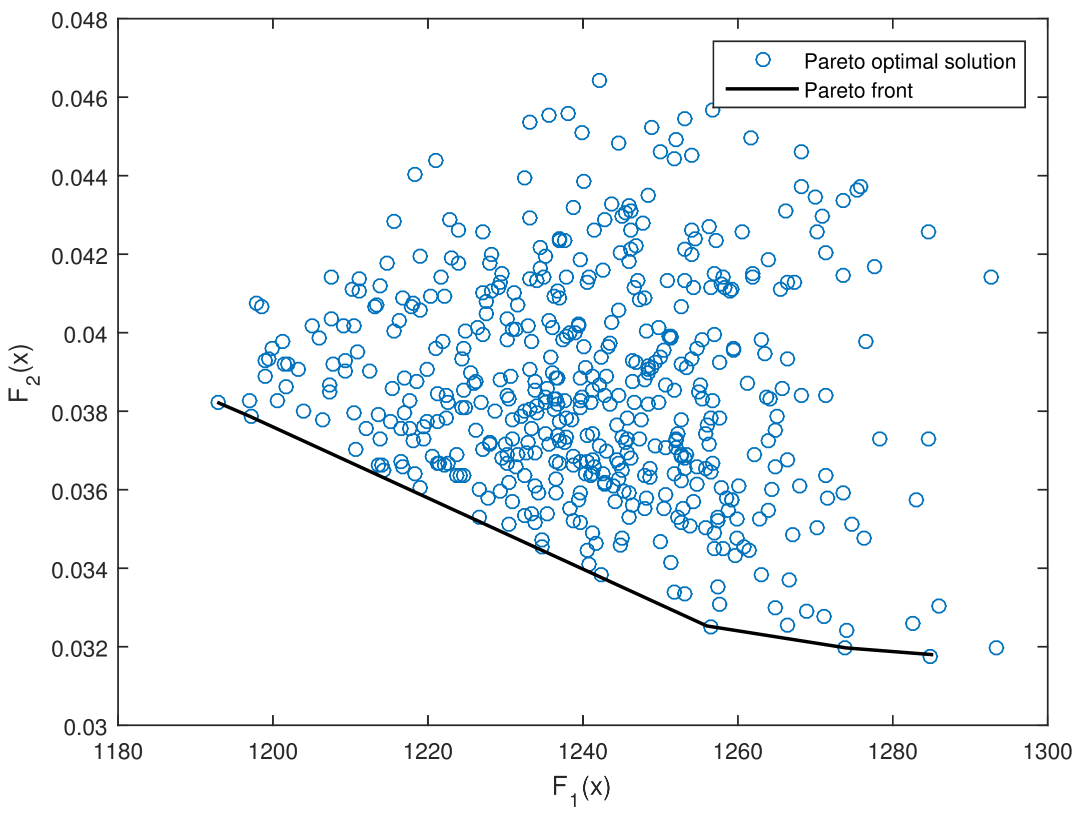


Figure 6.14: A four bar plane truss problem with 500 Pareto optimum solutions

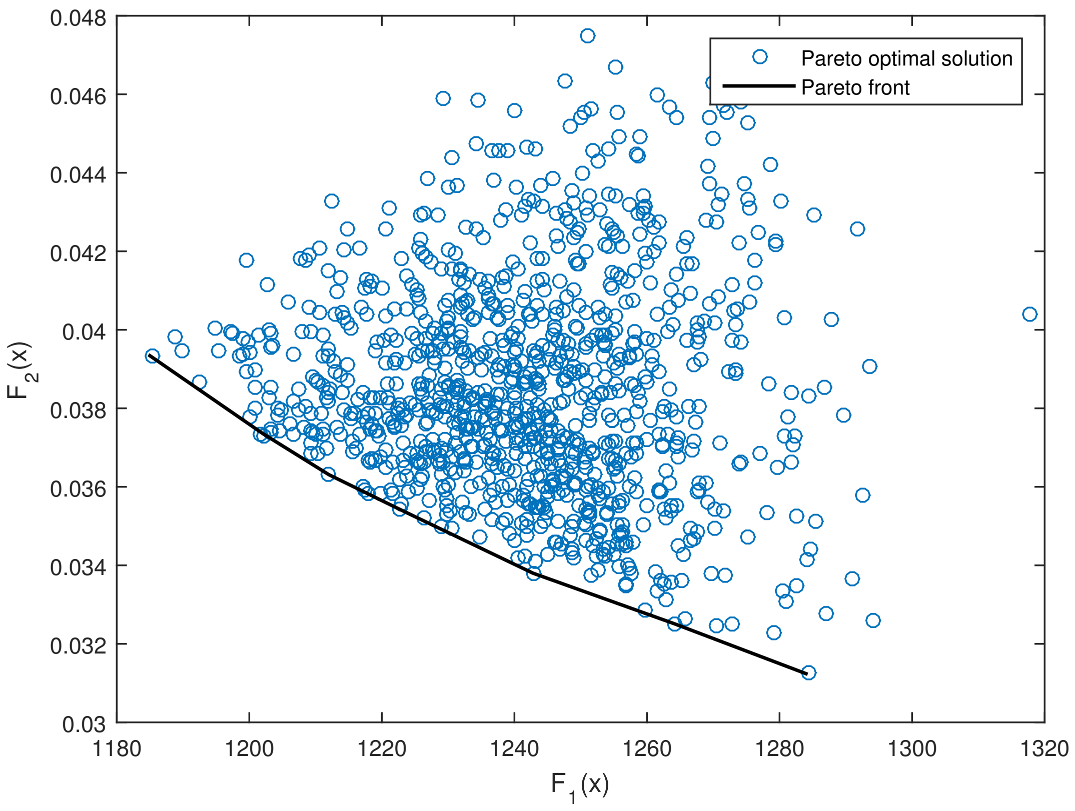


Figure 6.15: A four bar plane truss problem with 1000 Pareto optimum solutions

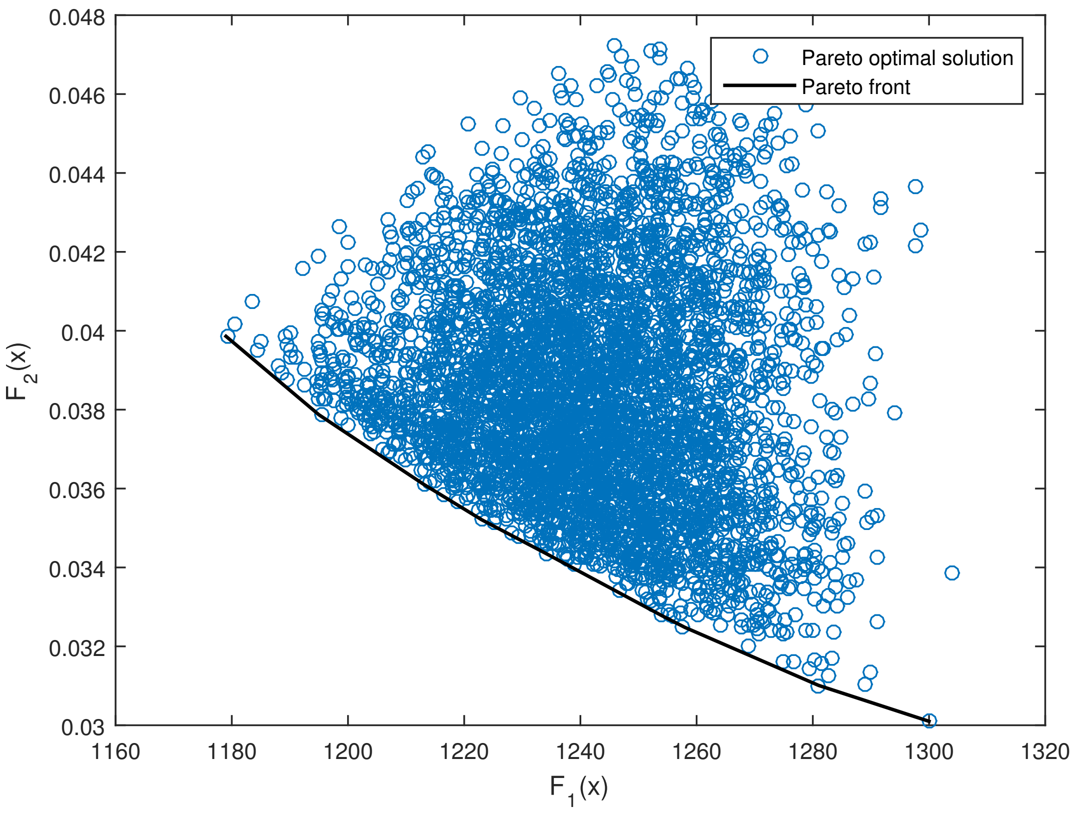


Figure 6.16: A four bar plane truss problem with 4000 Pareto optimum solutions

6.5 Application of dual-particle swarm optimisation-modified adaptive bats sonar algorithm to solve multi objective optimisation problems

6.5.1 Optimisation of the metal cutting process problem

The problem is to minimise operation time (τ) in *minute* and used tool life (ζ) in % during cutting process of a steel bar. There are three variables bound to the problem that are; depth of cut (x_1), feed rate (x_2) and cutting speed (x_3). Other parameters included in the problem are tool life (T), cutting force (F_c), maximum cutting power (P), material removal rate (MRR) and surface roughness (R). The problem formulation is given as:

$$
\begin{aligned}
&\text{Minimise}\\
&\quad F_1(x) = \tau(x)\\
&\quad \text{and}\\
&\quad F_2(x) = \zeta(x)\\
&\text{subject to}\\
&\quad g_1(x) \equiv 1 - \frac{P(x)}{(0.75)(10000)} \geq 0\\
&\quad g_2(x) \equiv 1 - \frac{F_c(x)}{5000} \geq 0\\
&\quad g_3(x) \equiv 1 - \frac{R(x)}{50e^{-6}} \geq 0\\
&\text{where}\\
&\quad \tau(x) = 0.15 + 219912\left(\frac{1+\frac{0.20}{T(x)}}{MRR(x)}\right) + 0.05\\
&\quad \zeta(x) = \frac{219912}{MRR(x)T(x)} \times 100\\
&\quad T(x) = \frac{5.48e^9}{(x_1)^{0.460}(x_2)^{0.696}(x_3)^{3.46}}\\
&\quad F_c(x) = \frac{6.56e^3(x_1)^{1.10}(x_2)^{0.917}}{(x_3)^{0.286}}\\
&\quad P(x) = \frac{x_3F_c(x)}{60000}\\
&\quad MRR(x) = 1000x_1x_2x_3\\
&\quad R(x) = \frac{125(x_2)^2}{0.0008}\\
&\quad 0.5\ mm \leq x_1 \leq 6\ mm\\
&\quad 0.15\ mm/rev \leq x_2 \leq 0.55\ mm/rev\\
&\quad 250\ m/min \leq x_3 \leq 400\ m/min
\end{aligned}
\tag{6.15}
$$

The D-PSO-MABSA is applied to solve this problem. 10 Pareto optimum points are considered for this problem. The D-PSO-MABSA was capable of minimising both operation time and used tool life objectives. Figure 6.17 shown the 10 Pareto optimum points achieved using D-PSO-MABSA. All 10 points are non-dominated solutions that formed a smooth Pareto front. According to the figure, the 3rd Pareto optimum solution was the best compromise solution acquired by the D-PSO-MABSA where operation time is 0.4584 *minutes* at 1.8235% of used tool life. The detailed results are shown in

Table 6.3.

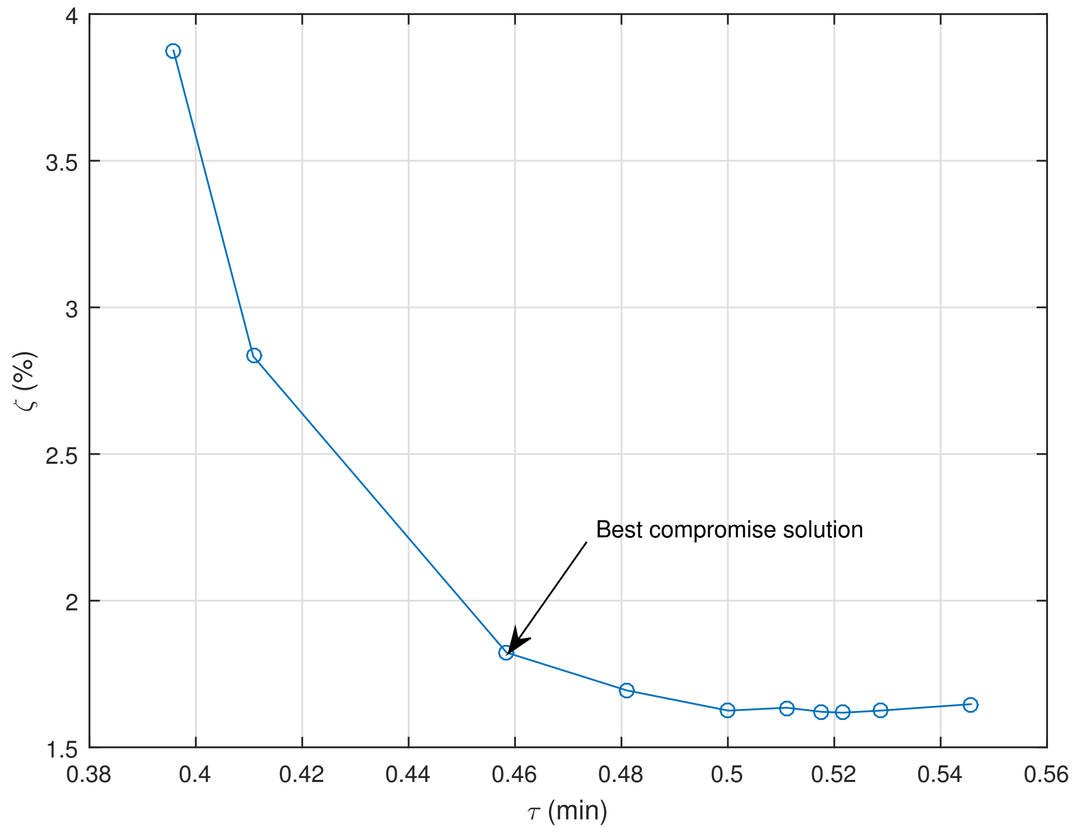


Figure 6.17: Pareto front of the metal cutting process problem

6.5.2 Optimisation of environmental/economic power dispatch of IEEE 30-bus 6-generator unit electrical network problem

The problem is to minimise fuel cost (F_1) in ($\$/h$) and amount of pollutant emission (F_2) in (ton/h) in order to set the real power of 6 generator unit ($P_{G_i},\ i = 1,2,\ldots,6$). The problem is formulated as:

Minimise

$$F_1(P_G) = \sum_{i=1}^{6} a_i + b_i P_{G_i} + c_i P_{G_i}^2$$

and

$$F_2(P_G) = \sum_{i=1}^{6} 10^{-2}(\alpha_i + \beta_i P_{G_i} + \gamma_i P_{G_i}^2) + \xi_i \exp(\lambda_i P_{G_i}) \tag{6.16}$$

where

$a_i, b_i, c_i =$ coefficients of the i^{th} generator cost

$\alpha_i, \beta_i, \gamma_i, \xi_i, \lambda_i =$ coefficients of the i^{th} generator emission characteristics

$0 \leq P_{G_1}, P_{G_2}, P_{G_3}, P_{G_4}, P_{G_5}, P_{G_6} \leq 1$

Table 6.3: Results of D-PSO-MABSA to optimise the metal cutting process problem

Number of Pareto	Depth of cut (x_1)	Feed rate (x_2)	Cutting speed (x_3)	Tool life (T)	Cutting force (F_c)	Maximum cutting power (P)	Material removal rate (MRR)	Surface roughness (R)	Operation time (τ)	Used tool life (ζ)
1	5.8268	0.5426	369.9320	4.8505	4796.3632	29.5721	1169585.0313	0.0460	0.3958	3.8764
2	5.9882	0.5461	327.8771	7.2395	5146.7949	28.1253	1072268.1742	0.0466	0.4108	2.8330
3	5.9758	0.5293	272.9303	13.9711	5258.0621	23.9181	863194.5173	0.0438	0.4584	1.8235
4	5.8823	0.5122	262.9278	16.3829	5068.5157	22.2109	792163.3992	0.0410	0.4810	1.6945
5	5.4309	0.5342	255.3513	18.2632	4865.2306	20.7057	740780.2122	0.0446	0.5001	1.6255
6	5.8935	0.4725	256.6405	18.8255	4749.9689	20.3172	714691.5970	0.0349	0.5110	1.6345
7	5.7135	0.4810	254.5911	19.3919	4676.7471	19.8443	699651.0601	0.0361	0.5176	1.6209
8	5.2549	0.5211	252.2744	19.6729	4602.5445	19.3517	690796.8948	0.0424	0.5216	1.6182
9	5.7191	0.4655	253.8825	20.0220	4546.9657	19.2399	675896.4134	0.0339	0.5286	1.6250
10	5.6024	0.4527	253.2317	20.7915	4336.3328	18.3016	642304.6267	0.0320	0.5457	1.6467

Figure 6.18 shows the network configuration of 6-generator unit electrical network. The coefficients of cost and emission characteristics for 6 generators are given in Table 6.4 respectively.

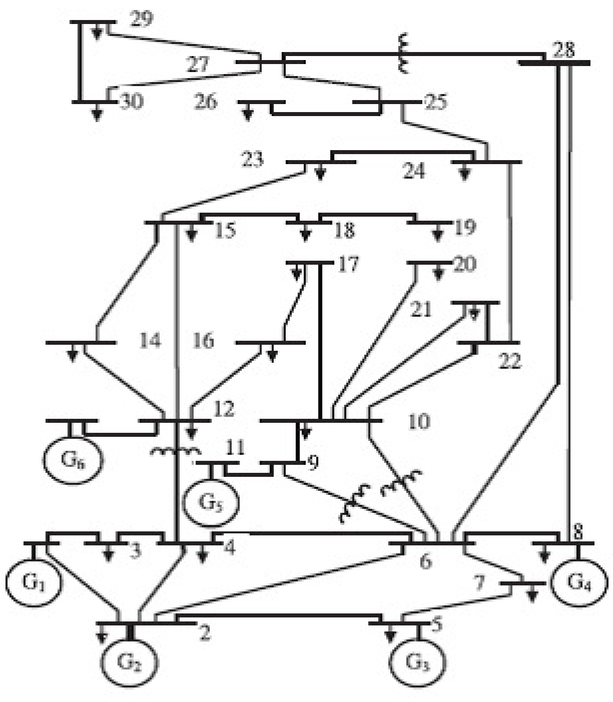


Figure 6.18: The IEEE 30-bus 6-generator electrical network configuration

Table 6.4: Generator cost and emission coefficients of the IEEE 30-bus 6-generator unit electrical network

Item	a	b	c	α	β	γ	ξ	λ
P_{G_1}	10	200	100	4.091	-5.543	6.490	$2.0e^{-4}$	2.857
P_{G_2}	10	150	120	2.543	-6.047	5.638	$5.0e^{-4}$	3.333
P_{G_3}	20	180	40	4.258	-5.094	4.586	$1.0e^{-6}$	8.000
P_{G_4}	10	100	60	5.326	-3.550	3.380	$2.0e^{-3}$	2.000
P_{G_5}	20	180	40	4.258	-5.094	4.586	$1.0e^{-6}$	8.000
P_{G_6}	10	150	100	6.131	-5.555	5.151	$1.0e^{-5}$	6.667

The D-PSO-MABSA was applied to solve this problem. 200 Pareto optimum points were considered for this problem. The D-PSO-MABSA was capable of determining the optimum real power setting of 6-generator unit by minimising fuel cost and pollutant emission objectives. The location of 200 Pareto optimum solutions acquired using D-PSO-MABSA is shown in Figure 6.19. As referred to the figure, there were five non-dominated solutions that formed a Pareto front. One of them is the best compromise solution between the two objectives studied.

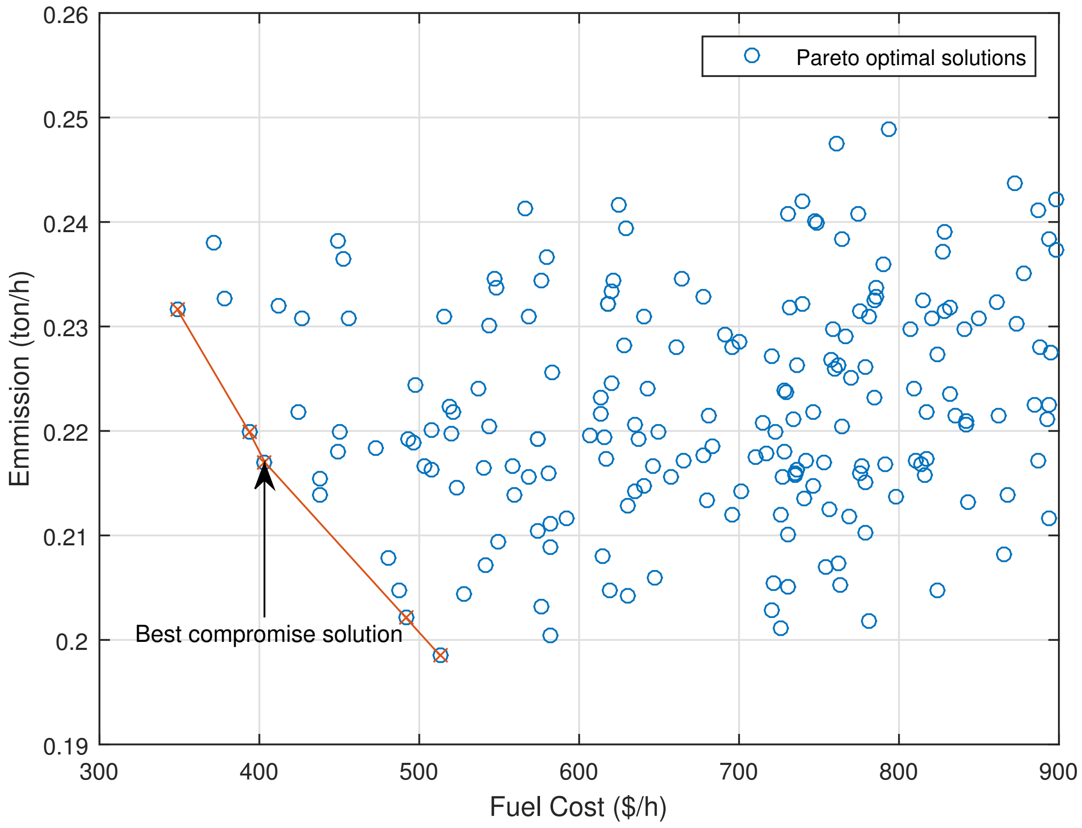


Figure 6.19: Pareto optimum solutions with Pareto front of the IEEE 30-bus 6-generator unit electrical network problem

Table 6.5 shows the best compromise solution for the problem achieved by using D-PSO-MABSA. According to the table, the cost of \$403.5647 / *hour* and the emission of 0.2170 *ton / hour* will be the best trade-off solution between fuel cost and pollutant emission objectives of the problem. Meanwhile, the 5th generator unit which obtained 0.628052 *MW* was the highest real power but recorded \$158.8273 / *hour* which is the highest fuel cost among other generator units. On the other hand, the 2nd generator unit recorded 0.0240 *ton / hour* of emission individually which was the minimum amount of emission of all, that was only 11 % from overall amount of pollutant emission by the system.

Table 6.5: Best simulation result of the IEEE 30-bus 6-generator unit electrical network

Generator	Real power obtained MW	Individual best fuel cost $\$/h$	Individual best pollutant emission ton/h
P_{G_1}	0.1601	44.5913	0.0340
P_{G_2}	0.0338	15.2008	0.0240
P_{G_3}	0.2448	66.4623	0.0329
P_{G_4}	0.3624	54.1182	0.0490
P_{G_5}	0.6281	158.8273	0.0288
P_{G_6}	0.3482	75.1681	0.0483
Best compromise solution		Fuel cost ($\$/h$)	Pollutant emission (ton/h)
		403.5647	0.2170

Chapter 7

Conclusion

A study of the investigation of bats echolocation-inspired algorithms for solving continuous optimisation problems has been presented. The work has focused on the modification of bats sonar algorithm (BSA) that was previously introduced by Tawfeeq in 2012 to produce a new set of algorithms. The newly bats echolocation-inspired algorithms have the ability to balance between exploration and exploitation processes of the algorithm to find the global optimum. Moreover, the bats-echolocation-inspired algorithms perform well to achieve better accuracy while maintaining good precision and fast convergence to the optimum solution of continuous optimisation problems considered.

The bats echolocation-inspired algorithms similar to particle swarm optimisation (PSO), artificial bee colony (ABC) or ant colony optimisation (ACO) are categorised under swarm intelligence algorithms. In a wide perspective, swarm intelligence algorithms fall under evolutionary algorithms as they are similar to genetic algorithm, evolutionary strategy and differential evolution. The evolutionary algorithms are a part of metaheuristic methods that work based on a combination of a set of rules and randomness.

In brief, the continuous optimisation problem type can be divided into three major categories; single objective optimisation problem, constrained optimisation problem and multi objective optimisation problem. Solving a single objective optimisation problem is about finding an optimised solution to a no constraint problem based on a single objective. Constrained optimisation problem on the other hand is dealing with problems with one or more constraint(s) based on single objective. The multi objective optimisation problem is more complicated where the problem is either with or without constraint(s), solving the problem is to seek compromised solutions based on a set of conflicting two and more objectives.

There are two major algorithms found in the literature that are inspired from the bats echolocation. First is a bat algorithm (BA) by Yang in 2010 that is based on the loudness, frequency and rate of pulse emitted. Second is BSA by Tawfeeq in 2012 which models the principles of bats sonar used in echolocation. The research carried out in this thesis has focused on the investigation of modified and enhanced versions of bats echolocation-inspired algorithms based on BSA.

An adaptive bats sonar algorithm (ABSA) has been proposed for solving single objective optimisation problems. The ABSA has been researched as an improved version of BSA by altering and incorporating new bats echolocation characteristics into it. The reciprocal altruism characteristic of a colony of bats has further been incorporated into ABSA to strengthen the procedure of algorithm searching for the best solution. A series of computer simulations has been carried out to evaluate the performance of the ABSA. The simulation is used to study the effects of number of bats and number of iterations in ABSA, to investigate the performance of ABSA to solve *black-box optimisation benchmarking 2013* as compared to PSO, and to analyse the performance of ABSA to solve established single objective optimisation benchmark test functions as compared to BSA and BA. The obtained results have demonstrated the superior performance of ABSA to achieve better accuracy while maintaining good precision and fast convergence to the optimum solution.

A modified adaptive bats sonar algorithm (MABSA) has been proposed for solving constrained optimisation problems. The MABSA has been formulated as an improved version of ABSA and BSA. In addition to redefining ABSA parameters, a new strategy, namely the bounce back strategy as a mechanism to control the transmitted beam to fall only within the designated search space, has been incorporated into MABSA. The MABSA hsa achieved competitive results on four constrained optimisation benchmark test functions adopted from *CEC 2006* and six well-known engineering design optimisation problems at a relatively better optimum solution value with a low computational cost. From the comparative study, MABSA has shown its ability to handle various constrained optimisation, and its outstanding performance is much better, in terms of statistical metrics, than the established set of algorithms selected from various literatures.

A dual-particle swarm optimisation-modified adaptive bats sonar algorithm (D-PSO-MABSA) has been proposed for solving multi objective optimisation problems. The D-PSO-MABSA is a hybrid algorithm integrating PSO and MABSA. This dual level search strategy works where at every time to obtain one Pareto optimum point, there are always two levels of the search process. PSO acts as a global search agent in the first level while in the second level, MABSA works as a local search agent and utilises the optimum solutions obtained by the PSO to initialise the bats in the MABSA. Swarm flight attitude and swarm searching strategy are two factors taken into consideration in setting PSO as a global search agent and MABSA as a local search agent. The proficiency of the D-PSO-MABSA to solve the multi objective optimisation problems has been examined through two different sets of computer simulation tests. The first test was about the performance and the reflection of algorithm parameters on the established multi objective optimisation benchmark test functions. The second test was to show the capability of the D-PSO-MABSA to solve an engineering design problem. The computer simulation results have demonstrated the potential of the D-PSO-MABSA to solve a variety of multi objective optimisation problems.

The performances of bats echolocation-inspired algorithms have been assessed to selected practical problems in business, mechanical/manufacturing engineering and electrical engineering fields. First, the ABSA was applied to solve two single objective optimisation problems named cost optimisation of shipping refined oil and profit optimisation of selling television sets. Next, the MABSA was utilised to solve two constrained optimisation problems; weight optimisation of the car side impact design and efficiency optimisation of brushless wheel DC motor. Lastly, the D-PSO-MABSA was used to solve two multi objective optimisation problems that are: optimisation of the metal cutting process problem and optimisation of environmental/economic power dispatch of IEEE 30-bus 6-generator unit electrical network problem. The results indicated that the bats echolocation-inspired algorithms demonstrated good capability and promising performance to handle single objective optimisation, constrained optimisation and multi objective optimisation real problems in various areas.

Bibliographies

Abbass, H., Sarker, R., and Newton, C. (2001). Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems. In *Evolutionary Computation (CEC'01), Proceedings of the 2001 Congress on*, volume 2, pages 971–978, Seoul, South Korea.

Abido, M. A. (2009). Multiobjective particle swarm optimization for environmental/economic dispatch problem. *Electric Power Systems Research*, 79(7):1105–1113.

Afshar, A., Haddad, O. B., Mariño, M. A., and Adams, B. (2007). Honey-bee mating optimization (hbmo) algorithm for optimal reservoir operation. *Journal of the Franklin Institute*, 344(5):452–462.

Afshar, M. H. (2013). Extension of the constrained particle swarm optimization algorithm to optimal operation of multi-reservoirs system. *International Journal of Electrical Power & Energy Systems*, 51:71–81.

Ahlén, I., Baagøe, H. J., and Bach, L. (2009). Behavior of scandinavian bats during migration and foraging at sea. *Journal of Mammalogy*, 90(6):1318–1323.

Airas, M. (2003). Echolocation in bats. In *Proceedings of spatial sound perception and reproduction. The postgrad seminar course of HUT Acoustics Laboratory*, pages 1–25, Helsinki, Finland.

Akay, B. and Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, 23(4):1001–1014.

Akhtar, S., Ahmad, A. R., and Abdel-Rahman, E. M. (2012). A metaheuristic bat-inspired algorithm for full body human pose estimation. In *Computer and Robot Vision (CRV). 2012 Ninth Conference on*, pages 369–375, Toronto, Canada.

Altringham, J. D., Hammond, L., and McOwat, T. (1996). *Bats: biology and behaviour*. The Oxford University Press.

Amirjanov, A. (2006). The development of a changing range genetic algorithm. *Computer Methods in Applied Mechanics and Engineering*, 195(19):2495–2508.

Antoniou, A. and Lu, W.-S. (2007). *Practical optimization. Algorithms and engineering applications*. Springer (India) Private Limited.

Arita, H. T. and Fenton, M. B. (1997). Flight and echlocation in the ecology and evolution of bats. *Trends in Ecology & Evolution*, 12(2):53–58.

Askarzadeh, A. (2014). Bird mating optimizer: an optimization algorithm inspired by bird mating strategies. *Communication in Nonlinear Science and Numerical Simulation*, 19(4):1213–1228.

Babu, B. V. and Gujarathi, A. M. (2007). Multi-objective differential evolution (mode) algorithm for multi-objective optimization: parametric study on benchmark test problems. *Journal on Future Engineering & Technology*, 3(1):47–59.

Bandyopadhyay, S. and Saha, S. (2013). Some single-and multiobjective optimization techniques. In *Unsupervised Classification*, pages 17–58. Springer.

Banks, A., Vincent, J., and Anyakoha, C. (2007). A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6:467–484.

Becerra, R. L. and Coello, C. A. C. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195(33):4303–4322.

Bingdong, L., Jinlong, L., Ke, T., and Xin, Y. (2015). Many-objective evolutionary algorithms: a survey. *ACM Computing Surveys*, 48(1):1–35.

Binh, T. T. and Korn, U. (1997). Mobes: A multiobjective evolution strategy for constrained optimization problems. In *The Third International Conference on Genetic Algorithms (Mendel 97)*, volume 25, page 27, Brno, Czech Republic.

Bora, T. C., Coelho, L. d. S., and Lebensztajn, L. (2012). Bat-inspired optimization approach for the brushless dc wheel motor problem. *IEEE Transactions on Magnetics*, 48(2):947–950.

Brest, J., Greiner, S., Bošković, B., Mernik, M., and Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657.

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

Campos, M., Krohling, R. A., and Enriquez, I. (2014). Bare bones particle swarm optimization with scale matrix adaptation. *IEEE Transactions on Cybernetics*, 44(9):1567–1578.

Castro-Gutierrez, J., Landa-Silva, D., and Pérez, J. M. (2010). Improved dynamic lexicographic ordering for multi-objective optimisation. In *Parallel Problem Solving from Nature, PPSN XI*, pages 31–40. Springer.

Coelho, L. d. S. and Mariani, V. C. (2008). Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Systems with Applications*, 34(3):1905–1913.

Coello, C. A. C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127.

Coello, C. A. C. (2001). A short tutorial on evolutionary multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer.

Coello, C. A. C. (2006). Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36.

Coello, C. A. C. and Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190.

Coello, C. A. C. and Lechuga, M. S. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Evolutionary Computation (CEC'02), Proceedings of the 2002 Congress on*, volume 2, pages 1051–1056, Hawaii, USA.

Coello, C. A. C. and Mezura-Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203.

Conn, A. R., Scheinberg, K., and Toint, P. L. (1997). Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79(1):397–414.

Cuevas, E. and Cienfuegos, M. (2014). A new algorithm inspired in the behavior of the social-spider for constrained optimization. *Expert Systems with Applications*, 41(2):412–425.

Cvetkovic, D. and Parmee, I. C. (1998). Evolutionary design and multi–objective optimisation. In *Intelligent Techniques and Soft Computing (EUFIT), Proceedings of the Sixth European Congress on*, pages 397–401, Aachen, Germany.

Damodaram, R. and Valarmathi, M. L. (2012). Experimental study on meta heuristic optimization algorithms for fake website detection. *International Journal of Emerging Technologies in Computational and Applied Sciences*, 2(1):43–53.

De Leon, D. (2012). Unconstrained optimization with several variables, in math 232- mathematical models with technology, spring 2012 lecture notes, pages 1-9, department of mathematics, california state university, fresno, usa. $http://zimmer.csufresno.edu/~doreendl/232.12s/handouts/multivaroptimization.pdf$. [Online], accessed on October 6, 2015.

Deb, K. (2014). *Optimization for engineering design. Algorithms and examples*. PHI Learning Private Limited, 2nd edition.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Pratap, A., and Meyarivan, T. (2001). Constrained test problems for multi-objective evolutionary optimization. In *Evolutionary Multi-Criterion Optimization*, pages 284–298. Springer.

DeNault, L. K. and McFarlane, D. A. (1995). Reciprocal altruism between male vampire bats, desmodus rotundus. *Animal Behaviour*, 49(3):855–856.

Dorigo, M. (1999). Ant colony optimization: A new meta-heuristic. In *Evolutionary Computation (CEC'99), Proceedings of the 1999 Congress on*, pages 1470–1477, Washington, USA.

Eberhart, R. C. and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation (CEC'01), Proceedings of the 2001 Congress on*, volume 1, pages 81–86, Seoul, South Korea.

Edgar, T. F., Himmelblau, D. M., and Lasdon, L. S. (2001). *Optimization of chemical processes*. University of Texas and McGraw Hill, 2nd edition.

Ehrlich, P. R., Dobkin, D. S., and Wheye, D. (1988). How fast and high do birds fly? http://web.stanford.edu/group/stanfordbirds/text/essays/HowFast.html. [Online], accessed on Mac 2, 2015.

Engelbrecht, A. P. (2005). *Fundamentals of computational swarm intelligence*. John Wiley & Sons.

Faritha Banu, A. and Chandrasekar, C. (2013). An optimized approach of modified bat algorithm to record deduplication. *International Journal of Computer Applications*, 62(1):10–15.

Fei, Y., Yu, H., and Xueshou, J. (2010). An improved constrained optimization genetic algorithm. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 2, pages 435–439, Xiamen, China.

Fenton, M., Audet, D., Orbrist, M., and Rydell, J. (1995). Signal strength, timing, and self-deafening: the evolution of echolocation in bats. *Paleobiology*, pages 229–242.

Fenton, M. B. (1997). Science and the conservation of bats. *Journal of Mammalogy*, 78(1):1–14.

Finck, S., Hansen, N., Ros, R., and Auger, A. (2013). Real-parameter black-box optimization benchmarking 2010: presentation of the noiseless functions. subtitle= working paper 2009/20, pp 1-102. http://coco.gforge.inria.fr/doku.php?id=bbob-2013. [Online], accessed on November 10, 2014.

Fister, I. J., Fister, D., and Yang, X. S. (2013). A hybrid bat algorithm. *Electrotehniški Vestnik*, 1-2:1–7.

Gandomi, A. H. and Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17:4831—4845.

Gandomi, A. H., Yang, X.-S., Alavi, A. H., and Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22(6):1239–1255.

Gao, L., Zou, D., Ge, Y., and Jin, W. (2010). Solving pressure vessel design problems by an effective global harmony search algorithm. In *Control and Decision Conference (CCDC), Proceedings of the 2010 Chinese on*, pages 4031–4035, Xuzhou, Chinese.

Garg, H. (2014). Solving structural engineering design optimization problems using an artificial bee colony algorithm. *Journal of Industrial and Management Optimization*, 10(3):777–794.

Ghasemi, A. (2013). A fuzzified multi objective interactive honey bee mating optimization for environment/economic power dispatch with valve point effect. *Electrical Power and Energy Systems*, 49:308–321.

Ghose, K., Horiuchi, T. K., Krishnaprasad, P. S., and Moss, C. F. (2006). Echolocating bats use a nearly time-optimal strategy to intercept prey. *PLoS Biology*, 4(5):865–873.

Gong, W., Cai, Z., and Liang, D. (2014). Engineering optimization by means of an improved constrained differential evolution. *Computer Methods in Applied Mechanics and Engineering*, 268:884–904.

Hamida, S. B. and Schoenauer, M. (2002). Aschea: new results using adaptive segregational constraint handling. In *Evolutionary Computation (CEC'02), Proceedings of the 2002 Congress on*, volume 1, pages 884–889, Honolulu, USA.

Hashmi, A., Goel, N., Goel, S., and Gupta, D. (2013). Firefly algorithm for unconstrained optimization. *IOSR Journal of Computer Engineering*, 11(1):75–78.

Havens, T. C., Spain, C. J., Salmon, N. G., and Keller, J. M. (2008). Roach infestation optimization. In *Swarm Intelligence Symposium (SIS 2008), Proceedings of the IEEE*, pages 1–7, St. Louis, USA.

He, Q. and Wang, L. (2007a). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20(1):89–99.

He, Q. and Wang, L. (2007b). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation*, 186(2):1407–1422.

Hofmeyr, S. A. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473.

Hsieh, T.-J. (2014). A bacterial gene recombination algorithm for solving constrained optimization problems. *Applied Mathematics and Computation*, 231:187–204.

Huang, Z., Ma, M., and Wang, C. (2008). An archived differential evolution algorithm for constrained global optimization. In *Smart Manufacturing Application (ICSMA 2008, International Conference on*, pages 255–260, Gyeonggi-do, South Korea.

Hughes, E. J. (2005). Evolutionary many-objective optimisation: many once or one many? In *Evolutionary Computation, Proceedings of the 2005 IEEE Congress on*, pages 222–227, Edinburgh, Scotland.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimisation: a short review. In *Evolutionary Computation, Proceedings of the 2008 IEEE Congress on*, pages 2424–2431, Hong Kong, China.

Jensen, M. E., Moss, C. F., and Surlykke, A. (2005). Echolocating bats can use acoustic landmarks for spatial orientation. *Journal of Experimental Biology*, 208(23):4399–4410.

Jiao, L., Li, L., Shang, R., Liu, F., and Stolkin, R. (2013). A novel selection evolutionary strategy for constrained optimization. *Information Sciences*, 239:122–141.

Jones, D. F., Mirrazavi, S. K., and Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1–9.

Kang, F., Li, J., and Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 181(16):3508–3531.

Kao, C.-C., Chuang, C.-W., and Fung, R.-F. (2006). The self-tuning pid control in a slider–crank mechanism system by applying particle swarm optimization approach. *Mechatronics*, 16(8):513–522.

Karaboga, D. and Basturk, B. (2007a). Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *Foundations of Fuzzy Logic and Soft Computing, Lecture Notes in Computer Science*, pages 789–798. Springer.

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

Karpat, Y. and Özel, T. (2007). Multi-objective optimization for turning processes using neural network modeling and dynamic-neighborhood particle swarm optimization. *The International Journal of Advanced Manufacturing Technology*, 35(3-4):234–247.

Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Evolutionary Computation (CEC'99), Proceedings of the 1999 Congress on*, volume 3, pages 1931–1938, Indianapolis, USA.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *International Conference on Neural Network IV, Proceedings of the IEEE*, pages 1942–1948, Perth, Australia.

Kennedy, J., Kennedy, J. F., Eberhart, R. C., and Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann.

Kennedy, J. and Mendes, R. (2002). Population structure and particle swarm performance. In *Evolutionary Computation (CEC'02), Proceedings of the 2002 Congress on*, volume 2, pages 1671–1676, Honolulu, USA.

Khan, K., Nikov, A., and Sahai, A. (2011). A fuzzy bat clustering method for ergonomic screening of office workplaces. In *Third International Conference on Software, Services and Semantic Technologies S3T 2011, Proceedings of the*, pages 59–66, Berlin, Germany.

Khan, K. and Sahai, A. (2012). A comparison of ba, ga, pso, bp and lm for training feed forward neural networks in e-learning context. *International Journal of Intelligent Systems and Applications*, 4(7):23–29.

Khan, K., Sahai, A., and Campus, A. (2012). A fuzzy c-means bi-sonar-based metaheuristic optomization algorithm. *International Journal of Artificial Intelligence and Interactive Multimedia*, 1(7):26–32.

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

Komarasamy, G. and Wahi, A. (2012). An optimized k-means clustering technique using bat algorithm. *European Journal of Scientific Research*, 84(2):26–273.

Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007.

Koziel, S. and Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44.

Kumar, R. (2014). Directed bee colony optimization algorithm. *Swarm and Evolutionary Computation*, 17:60–73.

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. In *Parallel Problem Solving from Nature*, pages 193–197. Springer.

Lee, K. S. and Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36):3902–3933.

Lemma, T. A. and Hashim, F. M. (2011). Use of fuzzy systems and bat algorithm for exergy modeling in a gas turbine generator. In *Humanities, Science and Engineering (CHUSER), Proceedings of the 2011 IEEE Colloquium on*, pages 305–310, Penang, Malaysia.

Li, Z., Shang, Z., Liang, J., and Niu, B. (2012). An improved differential evolution for constrained optimization with dynamic constraint-handling mechanism. In *Evolutionary Computation (CEC'12), Proceedings of the IEEE Congress on*, pages 1–6, Brisbane, Australia.

Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C. A. C., and Deb, K. (2006). Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, School of EEE, Nanyang Technological University, Singapore.

Liang, X., Li, W., Liu, P. P., Zhang, Y., and Agbo, A. A. (2015). Social network-based swarm optimization algorithm. In *12th International Conference on Networking, Sensing and Control. Proceedings of the 2015 IEEE*, pages 360–365, Taipei, Taiwan.

Lin, J.-H., Chou, C.-W., Yang, C.-H., and Tsai, H.-L. (2012a). A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *Computer and Information Technology*, 2(2):56–63.

Lin, J.-H., Chou, C.-W., Yang, C.-H., Tsai, H.-L., and Lee, I.-H. (2012b). A bio-inspired optimization algorithm for modeling the dynamics of biological systems. In *Innovations in Bio-Inspired Computing and Applications (IBICA), Proceedings of the 2012 Third International Conference on*, pages 206–211, Kaohsiung, Taiwan.

Liu, H., Cai, Z., and Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, 10(2):629–640.

Lovász, L. (2010). Discrete and continuous: two sides of the same? In *Visions in Mathematics*, pages 359–382. Springer.

Marichelvam, M. and Prabaharam, T. (2012). A bat algorithm for realistic hybrid flowshop scheduling problems to minimize makespan and mean flow time. *ICTACT Journal on Soft Computing*, 3(1):428–433.

Merriam-Webster (2015). Merriam-webster dictionary. http://www.merriam-webster.com/dictionary/optimization. [Online], accessed on October 23, 2015.

Messac, A., Sundararaj, G. J., Tappeta, R. V., and Renaud, J. E. (2000). Ability of objective functions to generate points on nonconvex pareto frontiers. *AIAA Journal*, 38(6):1084–1091.

Mezura-Montes, E. and Coello, C. A. C. (2005a). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17.

Mezura-Montes, E. and Coello, C. A. C. (2005b). Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *MICAI 2005: Advances in Artificial Intelligence*, pages 652–662. Springer.

Mishra, S., Shaw, K., and Mishra, D. (2012). A new meta-heuristic bat inspired classification approach for microarray data. *Procedia Technology*, 4:802–806.

Molga, M. and Smutnicki, C. (2005). Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf. [Online], accessed on July 17, 2013.

Moore, J. and Chapman, R. (1999). Application of particle swarm to multiobjective optimization. Technical report, Department of Computer Science and Software Engineering, Auburn University.

Murata, T., Ishibuchi, H., and Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957–968.

Musikapun, P. and Pongcharoen, P. (2012). Solving multi-stage multi-machine multi-product scheduling problem using bat algorithm. In *2nd International Conference on Management and Artificial Intelligence, Proceedings of the*, volume 35, pages 98–102, Singapore.

Nakamura, R. Y., Pereira, L. A., Costa, K., Rodrigues, D., Papa, J. P., and Yang, X.-S. (2012). Bba: A binary bat algorithm for feature selection. In *Graphics, Patterns and Images (SIBGRAPI), Proceedings of the 2012 25th SIBGRAPI Conference on*, volume 35, pages 291–297, Ouro Preto, Brazil.

Nebro, A. J., Durillo, J., Garcia-Nieto, J., Coello, C. A. C., Luna, F., and Alba, E. (2009). Smpso: A new pso-based metaheuristic for multi-objective optimization. In *Computational Intelligence in Multi-Criteria Decision-Making (MCDM'09), Proceedings of the IEEE Symposium on*, pages 66–73, Tennessee, USA.

Ngatchou, P., Zarei, A., and El-Sharkawi, M. A. (2005). Pareto multi objective optimization. In *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*, pages 84–91, Virginia, USA.

Novick, A. (1971). Echolocation in bats: Some aspects of pulse design: During insect pursuits, landings, and obstacle evasions, bats alter the design of their orientation pulses in ways which help us uncover the nature of their sonar. *American Scientist*, 59(2):198–209.

Osyczka, A. and Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10(2):94–99.

Parsopoulos, K. E. and Vrahatis, M. N. (2002). Particle swarm optimization method in multiobjective problems. In *ACM Symposium on Applied Computing, Proceedings of the 2002*, pages 603–607, Madrid, Spain.

Parsopoulos, K. E. and Vrahatis, M. N. (2005). Unified particle swarm optimization for solving constrained engineering optimization problems. In *Advances in Natural Computation*, pages 582–591. Springer.

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

Peer, E. S., van den Bergh, F., and Engelbrecht, A. P. (2003). Using neighbourhoods with the guaranteed convergence pso. In *Swarm Intelligence Symposium (SIS'03), Proceedings of the 2003 IEEE*, pages 235–242, Indianapolis, USA.

Pye, J. D. (1960). A theory of echolocation by bats. *The Journal of Laryngology & Otology*, 74(10):718–729.

Ramesh, B., Mohan, V. C. J., and Reddy, V. V. (2013). Application of bat algorithm for combined economic load and emission dispatch. *International Journal of Electrical and Electronic Engineering & Telecommunications*, 2(1):1–9.

Rao, R. V., Savsani, V. J., and Vakharia, D. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315.

Rao, S. S. (2009). *Engineering optimization: theory and practice*. John Wiley & Sons, 4th edition.

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

Reddy, V. U. and Manoj, A. (2012). Optimal capacitor placement for loss reduction in distribution systems using bat algorithm. *IOSR Journal of Engineering*, 2(10):23–27.

Rivers, N. M., Butlin, R. K., and Altringham, J. D. (2006). Autumn swarming behaviour of natterer's bats in the uk: population size, catchment area and dispersal. *Biological Conservation*, 127(2):215–226.

Rizk-Allah, R. M., Zaki, Elsayed, M., and El-Sawy, A. A. (2013). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Applied Mathematics and Computation*, 224:473–483.

Roeva, O., Fidanova, S., and Paprzycki, M. (2013). Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Computer Science and Information Systems (FedCSIS), Proceedings of the 2013 Federated Conference on*, pages 371–376, Krakow, Poland.

Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

Runarsson, T. P. and Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):233–243.

Sadollah, A., Bahreininejad, A., Eskandar, H., and Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing*, 13(5):2592–2612.

Sardiñas, R. Q., Santana, M. R., and Brindis, E. A. (2006). Genetic algorithm-based multi-objective optimization of cutting parameters in turning processes. *Engineering Applications of Artificial Intelligence*, 19:127–133.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary Computation (CEC'98).World Congress on Computational Intelligence, Proceedings of the 1998 IEEE International Conference on*, pages 69–73, Alaska, USA.

Sierra, M. R. and Coello, C. A. C. (2005). Improving pso-based multi-objective optimization using crowding, mutation and ε-dominance. In *Evolutionary Multi-Criterion Optimization*, pages 505–519. Springer.

Sierra, M. R. and Coello, C. A. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308.

Simmons, J. A., Howell, D. J., and Suga, N. (1975). Information content of bat sonar echoes: recent research on echolocation in bats identifies some of the kinds of information conveyed by echoes of their sonar sounds. *American Scientist*, 63(2):204–215.

Simmons, J. A., Saillant, P. A., Wotton, J. M., Haresign, T., Ferragamo, M. J., and Moss, C. F. (1995). Composition of biosonar images for target recognition by echolocating bats. *Neural Networks*, 8(7-8):1239–1261.

Stanimirovic, I. P. (2012). Compendious lexicographic method for multi-objective optimization. *Series Mathematics Informatics*, 27:55–66.

Statnikov, R., Matusov, J., and Statnikov, A. (2012). Multicriteria engineering optimization problems: statement, solution and applications. *Journal of Optimization Theory and Applications*, 155(2):355–375.

Stebbings, R. E., Yalden, D. W., and Herman, J. (2007). *Which bat is it?: a guide to bat identification in Great Britain and Ireland*. The Mammal Society, 3rd edition.

Suga, N. (1990). Biosonar and neural computation in bats. *Scientific American*, 262(6):60–68.

Surlykke, A., Futtrup, V., and Tougaard, J. (2003). Prey-capture success revealed by echolocation signals in pipistrelle bats (pipistrellus pygmaeus). *Journal of Experimental Biology*, 206(1):93–104.

Takahama, T. and Sakai, S. (2005). Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. *IEEE Transactions on Evolutionary Computation*, 9(5):437–451.

Tawfeeq, M. A. (2012). Intelligent algorithm for optimum solutions based on the principles of bat sonar. *International Journal of Computer Science and Information Security*, 10(10):11–19.

Tessema, B. and Yen, G. G. (2006). A self adaptive penalty function based algorithm for constrained optimization. In *Evolutionary Computation (CEC'06), Proceedings of the 2006 IEEE Congress on*, pages 246–253, Vancouver, Canada.

Than, K. (2011). Highest flying bird found; can scale himalaya, national geographic news, june 10, 2011. http://news.nationalgeographic.com/news/2011/06/110610-highest-flying-birds-geese-himalaya-mountains-animals/. [Online], accessed on April 18, 2015.

Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., and Istanda, V. (2012). Bat algorithm inspired algorithm for solving numerical optimization problems. *Applied Mechanics and Materials*, 148:134–137.

Tuttle, M. D. (2006). Bats, artificial roosts, and mosquito control. https://www.batcon.org/pdfs/bathouses/MosquitoControl.pdf. [Online], accessed on April 16, 2015.

Vogler, B. and Neuweiler, G. (1983). Echolocation in the noctule (nyctalus noctula) and horseshoe bat (rhinolophus ferrumequinum). *Journal of Comparative Physiology*, 152(3):421–432.

Voigt-Heucke, S. L., Taborsky, M., and Dechmann, D. K. (2010). A dual function of echolocation: bats use echolocation calls to identify familiar and unfamiliar individuals. *Animal Behaviour*, 80(1):59–67.

Wang, G. and Guo, L. (2013). A novel hybrid bat algorithm with harmony search for global numerical optimization. *Journal of Applied Mathematics*, 2013:1–21.

Wang, G., Guo, L., Duan, H., Liu, L., and Wang, H. (2012). A bat algorithm with mutation for ucav path planning. *The Scientific World Journal*, 2012:1–15.

Wang, H., Jiao, L., and Yao, X. (2015). Two_ arch2 : an improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):524–541.

Wang, L. and Li, L. P. (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization*, 41(6):947–963.

Wang, Y. and Cai, Z. (2012). Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 16(1):117–134.

Wang, Y., Cai, Z., Zhou, Y., and Fan, Z. (2009). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413.

Waters, D. A. and Warren, R. (2003). *Bats*. The Mammal Society.

Wilkinson, G. S. (1988). Reciprocal altruism in bats and other mammals. *Ethology and Sociobiology*, 9(2):85–100.

Wong, E. Y. C., Lau, H. Y. K., and Mak, K. L. (2010). Immunity-based evolutionary algorithm for optimal global container repositioning in liner shipping. *OR Spectrum*, 32(3):739–763.

Xie, J., Zhou, Y., and Chen, H. (2013). A novel bat algorithm based on differential operator and lévy flights trajectory. *Computational Intelligence and Neuroscience*, 2013:1–13.

Yang, B., Chen, Y., and Zhao, Z. (2007). A hybrid evolutionary algorithm by combination of pso and ga for unconstrained and constrained optimization problems. In *Control and Automation, Proceedings of the 2007 IEEE International Conference on*, pages 166–170, Guangzhou, China.

Yang, B., Chen, Y., Zhao, Z., and Han, Q. (2006). A master-slave particle swarm optimization algorithm for solving constrained optimization problems. In *Intelligent Control and Automation (WCICA), Proceedings of the 2006 Sixth World Congress on*, volume 1, pages 3208–3212, Dalian, China.

Yang, X.-S. (2005). Engineering optimizations via nature-inspired virtual bee algorithms. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, pages 317–323. Springer.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications*, pages 169–178. Springer.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer.

Yang, X.-S. (2011). Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274.

Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing (NaBIC), Proceedings of the 2009 World Congress on*, pages 210–214, Coimbatore, India.

Yang, X.-S. and Deb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications*, 24(1):169–174.

Yang, X.-S. and Hossein, G. A. (2012). Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, 29(5):464–483.

Yang, X.-S., Karamanoglu, M., and Fong, S. (2012). Bat algorithm for topology optimization in microelectronic applications. In *Future Generation Communication Technology (FGCT), Proceedings of the 2012 International Conference on*, pages 150–155, London, United Kingdom.

Yıldız, A. R. (2009). A novel particle swarm optimization approach for product design and manufacturing. *International Journal of Advanced Manufacturing Technology*, 40(5-6):617–628.

Zahara, E. and Kao, Y.-T. (2009). Hybrid nelder–mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2):3880–3886.

Zeidler, E. (1995). Applied functional analysis. In *Applied Mathematical Sciences*, volume 108. Springer.

Zhang, C., Lin, Q., Gao, L., and Li, X. (2015). Backtracking search algorithm with three constraint handling methods for constrained optimisation problems. *Expert Systems with Applications*, 42:7831–7845.

Zhang, J. W. and Wang, G. G. (2012). Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 203:88–93.

Zhang, M., Luo, W., and Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, 178(15):3043–3074.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for Multiobjective Optimisation*, pages 3–37. Springer.