EVALUATION OF FIREFLY ALGORITHM USING BENCHMARK FUNCTIONS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Anuroop Kundur

In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

August 2013

Fargo, North Dakota

North Dakota State University

Graduate School

Title

EVALUATION OF FIREFLY ALGORITHM USING

BENCHMARK FUNCTIONS

By

ANUROOP KUNDUR

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair (typed)

Dr. Jun Kong

Dr. Chao You

Approved by Department Chair:

9-5-13
Date

Dr. Brian Slator (Associate Head)
Signature

ABSTRACT

The field of nature inspired computing and optimization techniques have evolved to solve the difficult optimization problems in diverse fields of engineering, science and technology. The Firefly algorithm is one of the several nature inspired algorithms that have been developed in the recent past and is inspired from the flashing behavior of the fireflies. The flashing behavior of the fireflies is to attract other fireflies in the group for mating. The less bright firefly will be attracted by the brighter one. As all the fireflies are assumed to be unisexual, each firefly is attracted to the other. This process is mimicked in the algorithm to find the solution to objective function. In this paper, we evaluate the algorithm using few multi-dimensional benchmark functions. The results of the simulation are satisfactory showing the algorithm to have good performance abilities.

ACKNOWLEDGEMENTS

I would like to take the opportunity to thank my advisor Dr. Simone Ludwig for unprecedented support, shrewd ideas and continuous mentoring throughout my research.

I would also like to thank Dr. Kendall Nygard for having that belief in me that I can do well in computer science given my previous specialization in electronics and his continuous support throughout my masters.

I would also like to thank my committee members for being a part of my research and for their support throughout.

Finally I would like to thank my family members, friends and others who have been a great support throughout my career and always had belief in me that I could do it.

DEDICATION

Dedicated to my parents and my brother

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Interest in optimization has grown after the invention of the digital computers in the late fifties. Formally, optimization or mathematical optimization is the process of formulation and finding the solution to a problem based on some constraints. The problem may be to either minimize or maximize the solution to constrained optimization problems. The problem or the function under consideration is either maximized or minimized by methodically selecting inputs from a set of allowed values [1]. Thus, the word optimum can mean either 'maximum' or 'minimum' based on the circumstances.

Optimization problems are a part of a wide range of fields, to name a few Engineering, Mathematics, Political sciences, Social sciences, Commerce, Economics etc. [2]. In most real life cases solving a problem may lead to innumerable solutions, optimization leads to the best solution of all based on some criterion.

1.1. Formulation

The formulation of the optimization problem is an important part of solving the problem itself. The basic parameters of an optimization problem include the objective function, constraints that pose limits to the objective function and the optimization variables:

Minimize $f(x)$ subject to $g_i(x) \leq b_i$ where $i = 1,\ldots.n$.

In the equation, $f(x)$ is the objective function (fitness function) under consideration that is being minimized, in this case subject to the constraints $g_i(x)$ where, b_i is a constant and the letter i represents the number of constraints on the objective function. The optimization variables or the decision variables are an important part of the optimization process. There are broadly two categories of optimization problems based on the type of decision variables:

i) Continuous optimization: The decision variables are continuous represented by real numbers [3].

ii) Integer optimization: The decision variables are discrete represented by integer numbers [3].

Optimization problems can be as simple as minimizing the area of a rectangle based on one constraint, to as complex as optimizing the air traffic at a busy airport like Chicago. The complexity of an optimization changes with adding more than one objective or fitness function and the number of decision variables. Optimization problems may be both unimodal and multi-modal. Unimodal optimization problems possess one good solution to the function and multi-modal problems imply that they have more than one good solution to the function under consideration; they could all be global maxima, minima or local in many cases too. Most of the real life situations involving optimization are multimodal.

1.2. Examples

Optimization is an indispensable part of many fields. Listed are some of the classic optimization problems. Travelling salesman is perhaps the most known and discussed optimization problem. The goal is to calculate the shortest path between a list of given cities and the distances between those cities.

Bin packing problem is a multi-disciplinary classic combinational optimization problem. It is computationally non-deterministic polynomial-time (NP) hard. In a bin packing problem objects of different volumes must be packed into finite number of bins of same volume.

In electronics, very large scale integrated (VLSI) layout/circuit design is a combinational optimization problem, in which the space on the VLSI chip is optimally used to placed transistors, capacitors etc. to increase the possibility to fabricate a smaller physical structure [4].

The channel assignment problem in wireless communication is a combinatorial optimization problem that is NP hard. The problem can be defined as assigning the least number of radio channels to a set of transceivers to avoid interference between any two given channels. Many approaches have been proposed to solve this problem like graph coloring, genetic algorithms and local search, etc. [5].

Optimization of benchmark functions using nature inspired genetic algorithms is an application in computational optimization. Benchmark functions are used to control and review the performance of the algorithms. The standard benchmark functions used to review the performance of algorithms are unimodal or multimodal with low and high dimensionality. The complexity of solving a NP-hard combinational problem is higher than any other optimization problems with no guarantee of reaching a global optimum.

1.3. Nature-inspired optimization algorithms

Nature provides some of the most efficient ways to solve problems. Algorithms that are inspired by nature and imitate a certain process from nature are nature inspired algorithms. Several nature inspired algorithms have been proposed to solve NP hard optimization problems, for example, nature inspired algorithms for mobile ad hoc routing [6], biologically inspired algorithms for financial modeling, etc [7].

Ant colony optimization, particle swarm optimization, bee algorithm and firefly algorithm are a few examples of nature inspired algorithms.

The firefly algorithm is a meta-heuristic optimization algorithm that follows the flashing behavior of fireflies. The brightness of the fireflies is affiliated with the objective function under consideration. The algorithm has been formulated with three main assumptions [8]:

i) All fireflies are unisexual, eliminating the possibility of attraction based on gender that is, each firefly will be attracted by all other fireflies.

ii) Attraction is dependent on the amount of brightness that is a less brighter firefly is attracted to a brighter one.

iii) The brightness of the firefly is equivalent to the objective function.

In this paper, we are evaluating the firefly algorithm in terms of objective value with regards to the number of generations, and increasing dimensions using five benchmark functions.

The arrangement of the chapters is as follows: Chapter 2 discusses related research. Chapter 3 describes the firefly algorithm and the approach that has been followed. Chapter 4 discusses the results that have been obtained using the evaluation procedure implemented. Chapter 5 concludes this study with ideas for future work.

2. LITERATURE REVIEW

The field of nature inspired computing has produced some ground breaking research results in the past few decades, and is expected to do so in the coming years too. Nature inspired computing mimics the patterns of living phenomena in nature to solve complex problems. The idea of nature inspired computing was set forth in the early 60's and has created a new era of algorithms that solve the most complex problems. Nature inspired algorithms can be broadly classified into two classes namely:

i) Evolutionary algorithms
   a) Genetic algorithm
   b) Memetic algorithm
   c) Neuroevolution

ii) Swarm intelligence
   a) Particle swarm optimization
   b) Ant colony optimization
   c) Firefly algorithm

2.1. Evolutionary algorithms

Evolutionary algorithms are a subset of nature inspired algorithms that apply mechanisms that closely mimic biological evolution. Evolutionary algorithms follow the precept of natural selection given a population of individuals, based on the survival rule "survival of the fittest". The basic steps in the algorithm are selection and mutation to generate candidates for the next generation. For any function to be maximized or minimized a set of candidate solutions in the function domain are generated and a fitness measure based on the function is applied to the candidate solutions to select the fittest of the generation. The fittest in the present generation are

then used to seed the next generation. The basic ways of generating offspring from the current generation is either by recombination or by mutation. This process is repeated for numerous generations until a good candidate solution with sufficient quality or until an agreed limit, i.e., a specific number of generations is reached [9].

2.1.1. Genetic algorithm

Perhaps the most referenced of the evolutionary algorithms is the genetic algorithm. The Algorithms mimics the natural evolution proposed by Charles Darwin. The basic steps of operation in a genetic algorithm are initialization, selection, crossover and mutation, termination. The algorithm is initiated by generating a random population of chromosomes, in the next step the fitness is evaluated for each of the chromosomes based on the function under evaluation. Based on the fitness value the best chromosomes are selected to reproduce the next generation of chromosomes by the crossover and mutation process. The current population is then replaced by the new population that is generated in the previous steps and the process of calculating the fitness function for the new population is repeated. The algorithm is iterated until a predetermined fitness value is reached or for prefixed number of generations or iterations has been reached [10]. The following image depicts the genetic algorithm cycle [11].

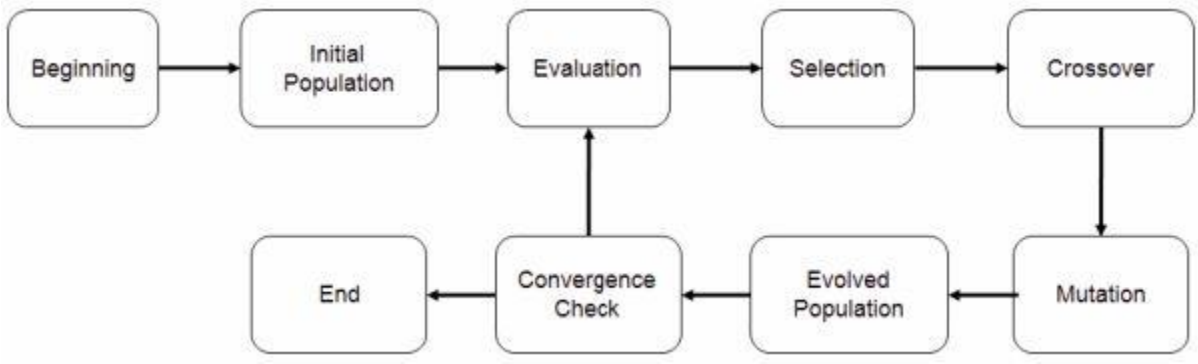

Figure 1: Genetic algorithm cycle [11]

Repeated calculation of the fitness function over many generations make genetic algorithms a costly way of answering real world problems.

Few of the prominent applications of genetic algorithms in real world problems are:

i) In automotive design for optimum design of vehicle suspensions and performance optimization of high end sport cars [12].

ii) For optimizing the telecommunications routing, which is still under development.

iii) For optimizing traffic and shipment routing to reduce congestions on roads [13].

2.1.2. Neural networks

Neural networks are a part of Neuroevolution that mimics the pattern of communication in the nervous system of the body. They are also referred to as artificial neural networks or neural nets [14]. The nervous system and its communication process is mimicked by using directed graphs in artificial neural networks, in which the directed connectors show the flow from input to the output performing function evaluations at different stages. The directed graphs usually consist of one input layer, one or more hidden layers and an output layer. Each node in the directed graph performs a simple computation and the connectors convey the signal from one node to the other. Each connector is assigned a weight and the weight is the amplification factor of the signal that is being propagated through. The weights are at first chosen randomly and are updated by a learning process given that the task to be accomplished by the neural network is known. Neural networks are constrained to only a few applications as of now and are to be researched much further to make it more diverse. The following figure depicts “neural networks” [15].

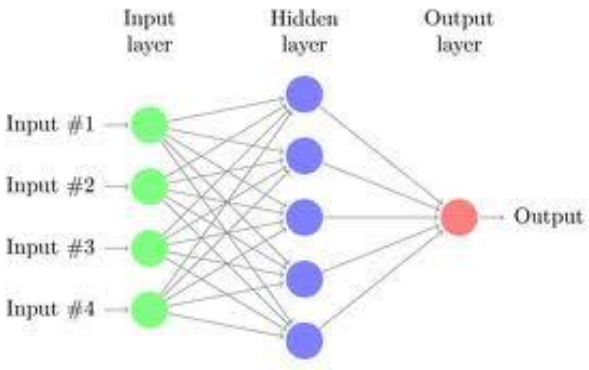

Figure 2: Example neural networks [15]

Artificial neural networks are mostly used in the fields of:

i) Robotics

ii) Control engineering

iii) Data processing

2.2. Swarm intelligence

Swarm intelligence is a collated behavior of disintegrated systems. The term was coined in 1989 by Beni and Wang. Swarm intelligence is mostly motivated from insects, birds and fish that follow a certain pattern when moving around in swarms or flocks. The members of the swarm must be able to communicate with each other about the environmental surroundings, for example, ants leave a trail of a certain chemical when going in search of food. Increase in the intensity of the chemical tells the ants on the trail that the food is closer. The swarms are also expected to acclimate to the conditions of the surroundings and vary decisions accordingly. A tight coupling exists between the swarm and its members. The members collectively control the behavior of the swarm. The members of the swarm interact with others in the swarm to solve a global problem.

2.2.1. Particle swarm optimization

Particle Swarm optimization is a stochastic optimization algorithm introduced in the first half of 90's by Eberhart and Kennedy. PSO is inspired by birds flocking. The intent of the development was to simulate graphically the pattern of a flock of birds flying in tandem and execute swift direction changes. The initialization in PSO is similar to that of genetic algorithm; a random population is generated to represent the members of the swarm. The search for optimum goes on for generations by updating the swarm throughout the generations [16]. The particles fly through the solution based on their own best value and the swarm's best value.

PSO has been applied to a wide variety of problems including multi-objective problems in which more than one objective function is optimized simultaneously.

2.2.2. Ant colony optimization

Another famous member of the swarm intelligence algorithms family is the Ant colony optimization algorithm. Ant colony optimization is a population based algorithm developed for combinatorial problems. It is inspired from the trail setting technique used by the ants when moving around in the search for food. The chemical deposit increases as the distance to the food decreases. This is mimicked by using weights in computational environment. Software agents called artificial ants search for optimal solutions in a weighted graph. Local minimums are also avoided as the chemical left by the ants on their trail tend to evaporate quickly [16].

Ant colony optimization applications include scheduling, assignment and routing problems, thus, making it more specific to routing and scheduling problems.

2.3. Firefly algorithm

Most of the nature inspired algorithms are metaheuristic. Heuristics are problem specific and are not efficient at solving others, whereas metaheuristics are indented to solve a wide range

of problems. Meta heuristics might not be as efficient as problem specific heuristics; they can easily replace heuristics when they are not readily available.

Every algorithm defined thus far has its own way of working out solutions to diverse problems. There might be some performance differences but all algorithms work towards establishing the goal. Every algorithm specializes at solving a certain issue, for example Ant Colony Optimization specializes at scheduling, assignment and routing problems and may lack the performance advantage when applied to solving multi-objective function, which is Particle Swarm Optimizations forte.

The firefly algorithm is a metaheuristic algorithm, which is inspired by the flashing behavior used by fireflies to attract each other in the mating process. It has been first proposed Yang in 2007 [8]. The brightness of the firefly is the key point of the algorithm, and is equivalent to the objective function under consideration. Three main assumptions were made when proposing the algorithm:

i) All fireflies are unisexual that is one firefly will be attracted by all others.

ii) Attraction is dependent on the amount of brightness, that is a less brighter firefly is attracted to a brighter one.

iii) The brightness of the firefly is equivalent to the objective function.

The attractiveness is dependent on the distance between the two fireflies as the intensity of light decreases as the distance between the two fireflies increases. Therefore, the closer the fireflies the more attractive they seem to each other. Multiple variants of firefly algorithm are being developed.

The firefly algorithm has been proved to be efficient at solving optimization tasks and can be more efficient than other meta-heuristic algorithms when applied to continuous

constrained optimization tasks [17], stochastic functions [18], multi-modal functions and even in the field of digital image processing. Given the wide range of applications, less complexity and more efficiency than other meta-heuristic make the firefly algorithm a good subject for continuing research.

3. FIREFLY ALGORITHM

Flashing behavior of the fireflies is unique to the kind of species they belong to and varies from one type of species to the other. The aim of such flashing behavior can be either of attracting mates or for attracting prey. The flashing light can be formulated to mimic the objective function under consideration and formulate a new optimization algorithm.

3.1. Formulation

The attractiveness of different fireflies leads to the movement of firefly towards the other. This is the basic idea of the formulation of the algorithm. Luminescence or light intensity is defined as the amount of light energy transmitted and it varies with the distance. The attractiveness of the fireflies varies with the brightness which is in turn related to the objective function in the mathematical domain. The intensity decreases with the increase in distance, and hence, a given firefly will be attracted to a firefly that is close to it even though it is less bright than a farther but brighter firefly. The intensity of light is known to vary inversely with the square of increasing distance or radius given by [18]:

$$I(r) \propto \frac{1}{r^2}$$

Where *I(r)* represent the light intensity as a function of distance and *r* is the radius. This can be converted to equality by adding a constant *I(s)*, which is intensity at the source. In real life conditions, every different form of energy is affected by atmospheric components, for example, sound traveling in a medium can be affected by the direction of wind and the amount of natural noise generated in the surrounding environment. Light is no exception to that and hence the intensity of light can also decrease depending on several environmental constraints. The intensity of light in real situations also depends on a factor called the absorption coefficient, and therefore, the inclusion of the absorption coefficient (γ) changes the equation to:

$$I(r) = I_0 e^{-\gamma r^2}$$

Attractiveness of the firefly is dependent and is directly proportional to the intensity of light. Hence the intensity equation can be transformed to represent the attractiveness as follows:

$$\beta = \frac{\beta_0}{1 + \gamma r^2}$$

Here β represents the attractiveness of the firefly and β_0 is the attractiveness at a radius 0; for practicality of implementation the attractiveness can vary as any power of radius rather than the square root [18]. The mapping of the parameters and corresponding notation used in the algorithm is shown in Table 1.

Table 1: Parameters and definitions

Parameter	Notation in Algorithm
Brightness	Objective function
Beta (β)	Attractiveness
Alpha (α)	Randomization parameter
Gamma (γ)	Absorption coefficient
Number of generations	Iterations
Number of fireflies	Population
Dimension	Problem dimension
R	Radius, time interval etc. (depends on application

3.2. Algorithm

In the firefly algorithm, the optimization process depends on the brightness of the fireflies and the movement of fireflies towards their brighter counterparts. Every firefly is attracted to the other depending on brightness because the fireflies are all unisexual according to the first assumption about artificial fireflies. The following section describes the pseudo code [21].

Define an initialize benchmark function $f(x)$, $x = (x_1, ..., x_d)$

Generate initial population of fireflies x_i (i = 1, 2, ..., n)

Determine light intensity for x_i by calculating $f(x_i)$

Define light absorption coefficient γ

While t < MaximumGeneration

Make a copy of the generated firefly population for move function

For i = 1 : n all n fireflies

For j = 1 : i all n fireflies

If ($I_j > I_i$),

Move fireflies i and j according to attractiveness

Evaluating new solutions and updating light intensity for next iteration

End if

End for j

End for i

Sorting the fireflies to find the present best

End while

Begin post process on best results obtained

The firefly algorithm starts by initializing a population of fireflies and each firefly is different from the other in the swarm. The differentiation is based on the brightness of the firefly. The brightness of the firefly is what determines the internal movement of the fireflies.

During the iterative process, the brightness of one firefly is compared with the others in the swarm and the difference in the brightness triggers the movement. The distance travelled depends on the attractiveness between the fireflies. During the iterative process the best solution thus far is continuously updated and the process goes on until certain stopping conditions are satisfied. After the iterative process comes to a halt the best solution of the evaluation is determined and the post process is initiated to obtain the results. The flowchart diagram is shown in Figure 3.

We modified an already existing Firefly algorithm code [19] to evaluate the performance of the algorithm by varying the input parameters like the number of iterations and the problem dimension. The measurement metrics added to the code allows to automatically generate the best, worst, average and standard deviation over several runs. The estimated running time for each run was also kept track of. The code was also automated to write the generated results to an excel file. The results obtained are discussed in detail in the next session.

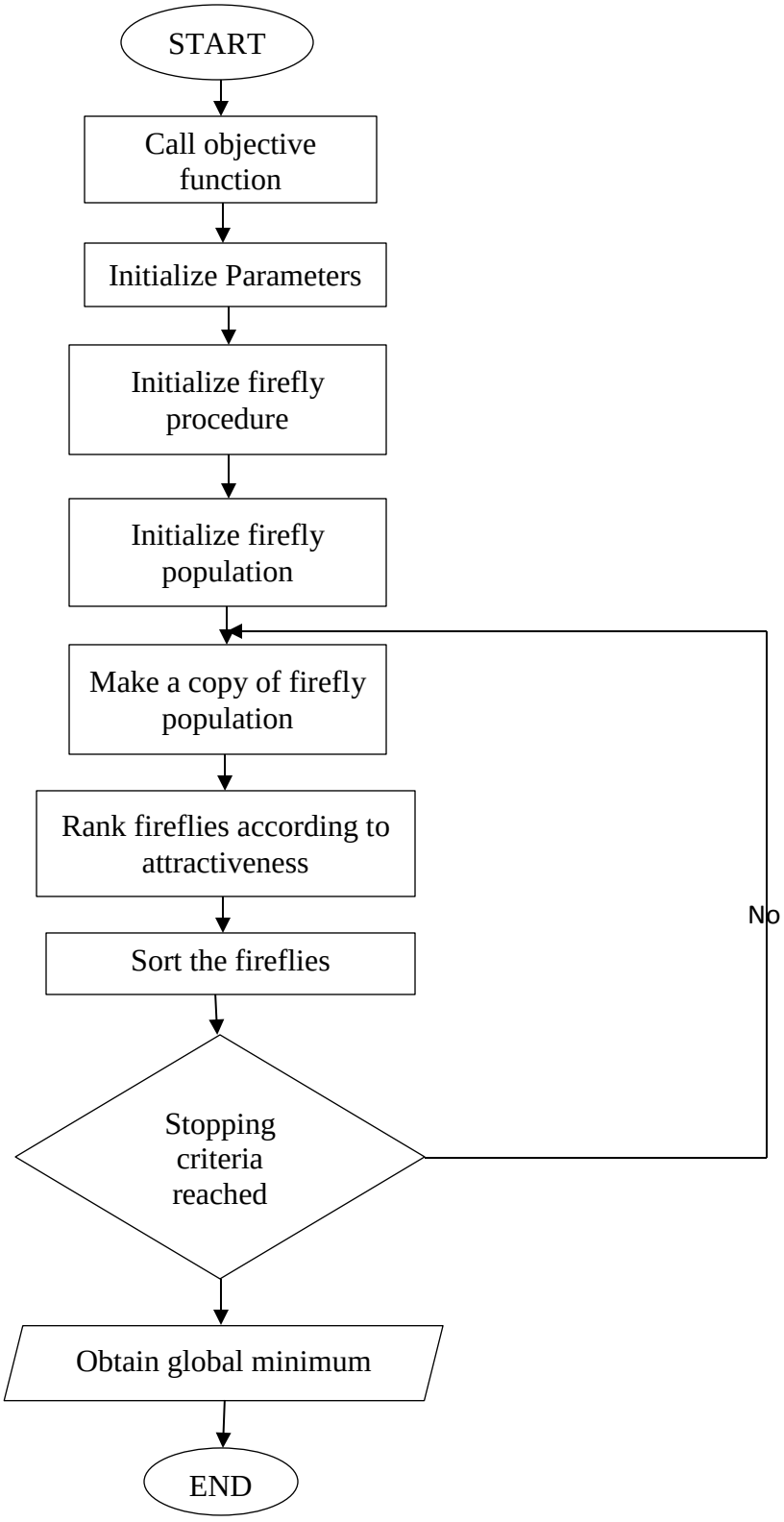

Figure 3: Flowchart illustrating the algorithm

4. EXPERIMENTAL SETUP AND RESULTS

4.1. Introduction

Global optimization techniques are required to be validated by a published set of benchmark problems. Benchmark functions are a way to test the optimization techniques.

4.2. Benchmark functions

There are four different types of benchmark functions designed for testing the global optimization problem [20]. All the classes discussed below are continuous:

a) Unimodal, convex, multidimensional: The set of functions designed for this class are easier to solve and have one global optimum. These functions can result in poor convergence to global optimum.

b) Multimodal and two dimensional functions with a few local optima: The set of functions under this class are of medium complexity and are serviceable to functions that have a few local optima and one global optimum.

c) Multimodal and two dimensional with many local optima: The set of functions under this class are more complex than the previous one as they are applicable to functions with large number of local optima.

d) Multimodal and multidimensional with many local optima: The functions under this set are more complex compared to the previous ones and are applied to intelligent optimization algorithms in combination with the previous class. Two dimensional problems are very rare in real world situations and hence multidimensional multimodal benchmark problems are prevalent in such situations.

Simple introductions to the functions that have been used to test the firefly algorithm have been defined below with their mathematical definition listed.

a) De Jong function: It is a simple and a continuous benchmark problem and belongs to the first class defined above. It is mathematically defined as:

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

b) Michalewicz function: Michalewicz function is a mutimodal test function with n! local optima and the parameter m defines the difficulty of the functions, as the m goes higher the difficulty of the search increases. It is defined mathematically as:

$$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$$

c) Rosenbrock function: It is a unimodal function and is also known as the second function of De Jong. It is defined as follows:

$$f(x) = [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

d) Schwefel function: Schwefel function is a multimodal separable function. Global optimum for the function is distant from the next best local optima and hence the function is tricky and can drive the algorithm in a different search direction. It is defined as:

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

e) Ackley function: Ackley is a multimodal non-separable function. It is widely used for validation and testing purposes of algorithms. It is mathematically defined as,

$$f(x) = -a.\exp\left(-b.\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n}\cos(cx_i)}{n}\right) + a + \exp(1)$$

Table 2: Benchmark problems

Function Name	Search Space
De Jong	-5.12 to +5.12
Michalewicz	0 to PI
Rosenbrock	-2.048 to +2.048
Schwefel	-500 to +500
Ackley	-32.768 to +32.768

4.3. Experimental setup

The benchmark functions used for running the simulations have been defined in the previous section. The simulations have been made with the following parameter setup for all functions.

1. Population = 80
2. Number of dimensions = 6

The number of iterations has been kept constant at 8,000 unless stated otherwise. The parameters alpha, beta, gamma and epsilon have been set at 0.25, 0.20, 1.0 and 0.001 respectively. The values of all the parameters remain constant unless stated otherwise.

Independent simulations have been run for each of the five functions defined in the previous section and the following results have been collected.

The outputs for 25 independent runs are:

1. Best value

2. Worst value
3. Mean Value
4. Standard deviation of mean value
5. Time elapsed

4.4. Results

The firefly algorithm has been tested using five different benchmark functions via 4 different test cases. In the first test case, the best objective value has been calculated for all functions over a fixed number of iterations. The second test case is used to evaluate the variation of the mean of 30 independent runs versus varying dimensions. The third test case is used to evaluate the variation of the mean of 30 independent runs versus varying iterations. The fourth test case is used to evaluate the variation of the best objective value versus varying iterations.

4.4.1. Test case 1: Best objective value versus fixed number of iterations

Table 3 shows the results of 25 independent runs on the benchmark function Ackley. The table includes all results discussed in the Section 4.3.1. The best objective value was found at 0.00072, while the worst was 9.35 and the mean of the 25 independent runs was 2.25507. The time elapsed for performing 25 independent runs and to collect the results was 470 seconds.

Table 3: Ackley benchmark function outputs

Best	Worst	Mean	Standard Deviation	Time (in sec.)
0.00072	9.35306	2.25507	2.27691	470

Table 4 shows the results of 25 independent runs on the benchmark function De Jong. The table includes all the results discussed in the Section 4.3.1. The best objective value was found at 0, while the worst was 0.00004 and the mean of the 25 independent runs was 0.00001. The variation between the best and the worst value indicates that the standard deviation of the

function is very low. The time elapsed for performing 25 independent runs and to collect the results was 178 seconds.

Table 4: De Jong benchmark function outputs

Best	Worst	Mean	Standard Deviation	Time (in sec.)
0	0.00004	0.00001	0.00001	178

Table 5 shows the results of 25 independent runs on the benchmark function Michalewicz. The table includes all the results discussed in the Section 4.3.1. The best objective value was found at -2.65692, while the worst was -4.68698 and the mean of the 25 independent runs was -4.10826. The time elapsed for performing 25 independent runs and to collect the results was 186 seconds.

Table 5: Michalewicz benchmark function results

Best	Worst	Mean	Standard Deviation	Time (in sec.)
-2.65692	-4.68698	-4.10826	0.55496	186

Table 6 shows the results of 25 independent runs on the benchmark function Rosenbrock. The table includes all the results discussed in the Section 4.3.1. The best objective value was found at 0.00774, while the worst was 3.99466 and the mean of the 25 independent runs was 0.18517. The time elapsed for performing 25 independent runs and to collect the results was 180 seconds.

Table 6: Rosenbrock benchmark function results

Best	Worst	Mean	Standard Deviation	Time (in sec.)
0.00774	3.99466	0.18517	0.77768	180

Table 7 shows the results of 25 independent runs on the benchmark function Schwefel. The table includes all the results discussed in the Section 4.3.1. The best objective value was found at 0.00003, while the worst was 217.1397 and the mean of the 25 independent runs was 91.69742. The time elapsed for performing 25 independent runs and to collect the results was 102 seconds.

Table 7: Schwefel benchmark function results

Best	Worst	Mean	Standard Deviation	Time
0.00003	217.1397	91.69742	69.49743	102

On evaluating the results, De Jong performed well in terms of standard deviation with a very low value of 1.0 E-05, and the performance of Schwefel was poor in terms of standard deviation with a very high value of 69.49743. Schwefel took the least time of 102 sec. to evaluate all the results. The rest of the functions performed moderately in terms of time complexity.

4.4.2. Test case 2: Simulations performed by varying the problem dimension

The following section discusses the results obtained by varying the problem dimension over a constant number of iterations. 30 independent repetitions have been performed for five functions and the mean of all the thirty independent runs has been calculated. Table 8 shows the collected results.

Presented below the table are the individual graphs for each function displaying the variation of mean value over 30 independent runs with increase in problem dimension.

Table 8: Average results for function on varying dimension

Function Name	Average Result				
	Dim = 2	**Dim = 4**	**Dim = 6**	**Dim = 8**	**Dim = 10**
Ackley	10.82313	13.29198	16.17964	17.8368	18.63649
Dejong	4.84E-07	5.38E-06	1.95E-05	4.04E-05	7.36E-05
Michalewicz	0.801303	2.527208	4.287659	6.095856	7.941764
Rosenbrock	5.63E-05	0.149998	0.164495	0.174542	0.343287
Schwefel	51.99992	735.7153	1360.356	1890.222	2630.118

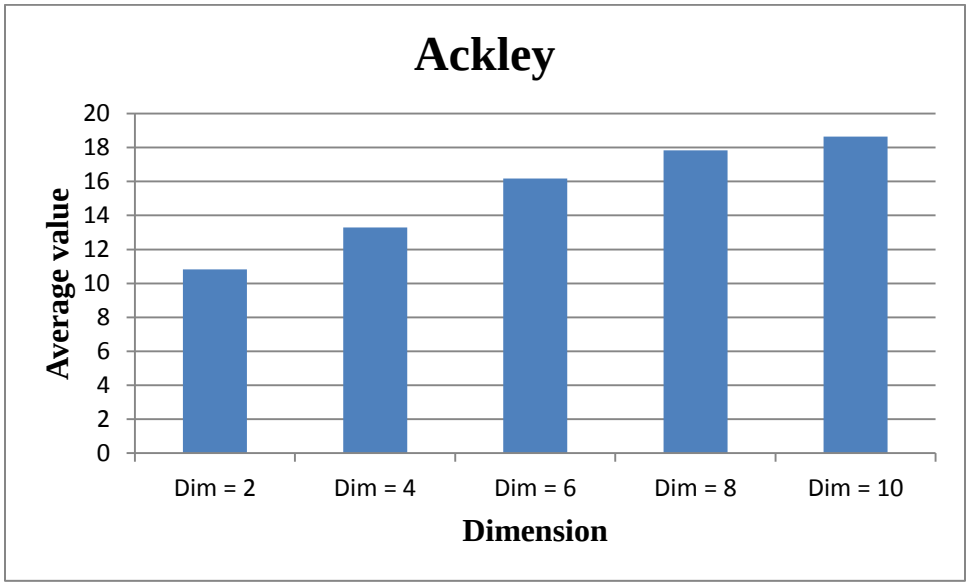

Figure 4: Mean values for Ackley against dimension

The variation of the mean versus augmenting dimension value for Ackley is shown in Figure 4. The lowest mean value 10.823 was for dimension 2, and the highest mean value 18.636 was obtained for dimension 10.

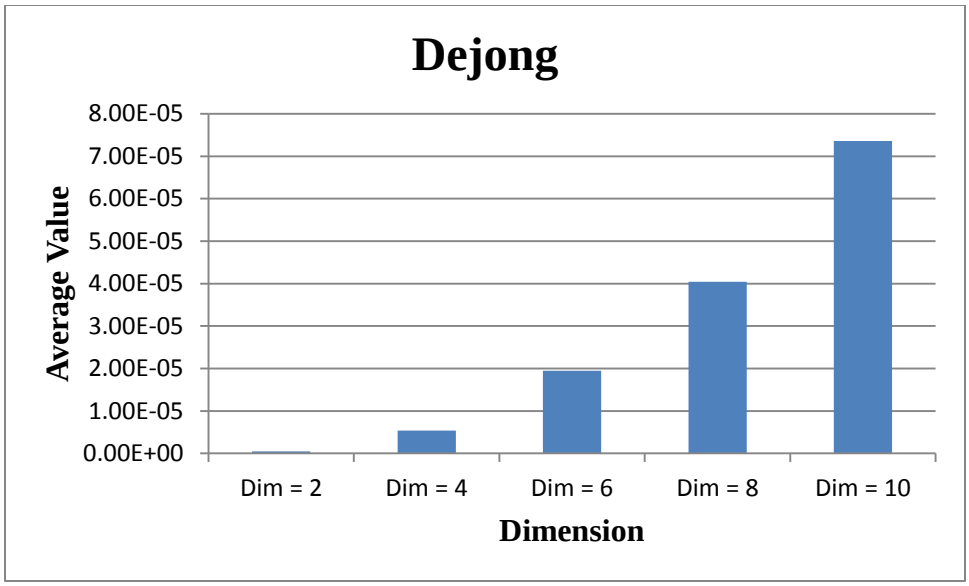

Figure 5: Mean values for De Jong against dimension

The variation of the mean versus augmenting dimension value for De Jong is shown in Figure 5. The lowest mean value 4.84E-07 was achieved for dimension 2, and the highest mean value 7.36E-05 was obtained for dimension 10.

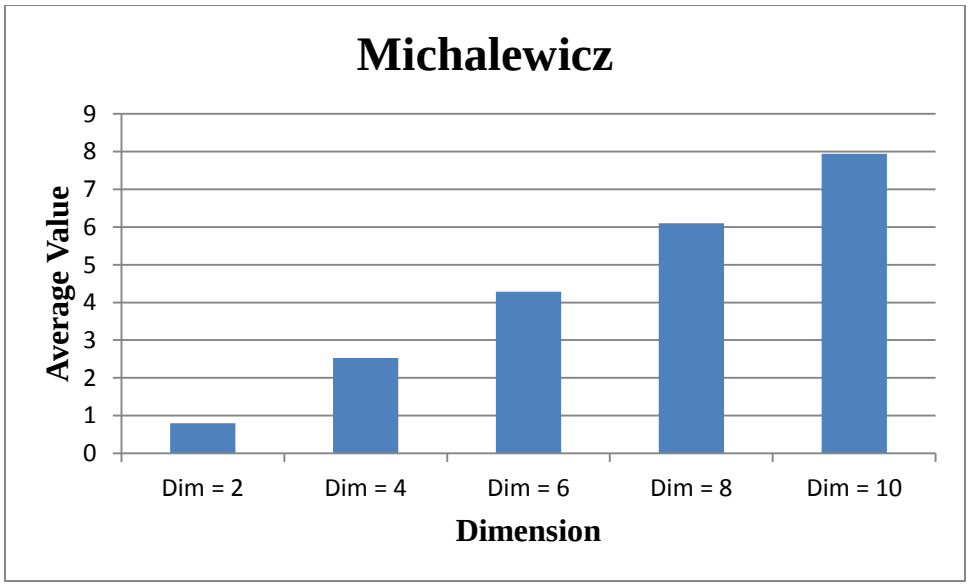

Figure 6: Mean values for Michalewicz against dimension

The variation of the mean versus augmenting dimension value for Michalewicz is shown in Figure 6. The lowest mean value 0.8013 was achieved for dimension 2, and the highest mean value 7.941 was obtained for dimension 10.

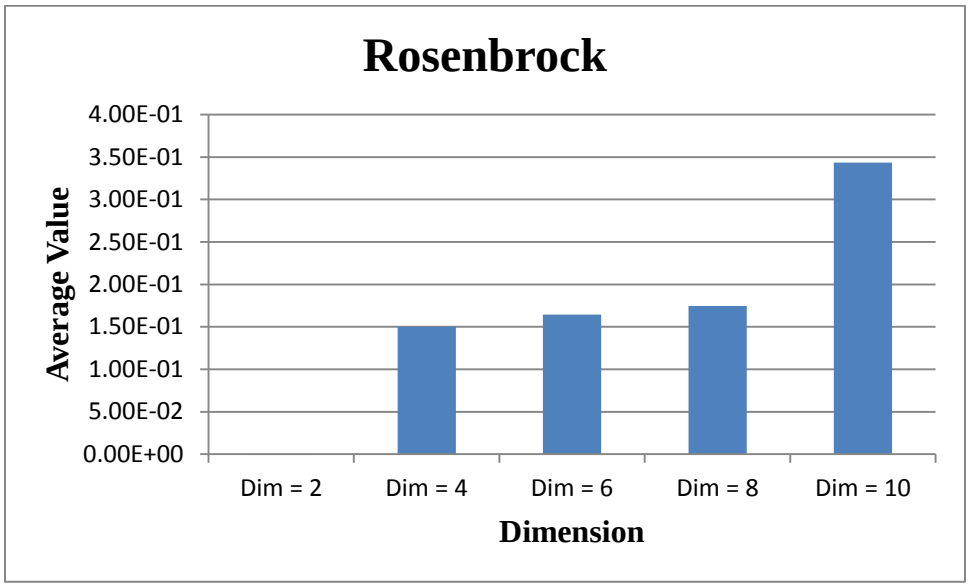

Figure 7: Mean values for Rosenbrock against dimension

The variation of the mean versus augmenting dimension value for Rosenbrock is shown in Figure 7. The lowest mean value 5.63E-05 was achieved for dimension 2, and the highest mean value 3.43E-01 was obtained for dimension 10.

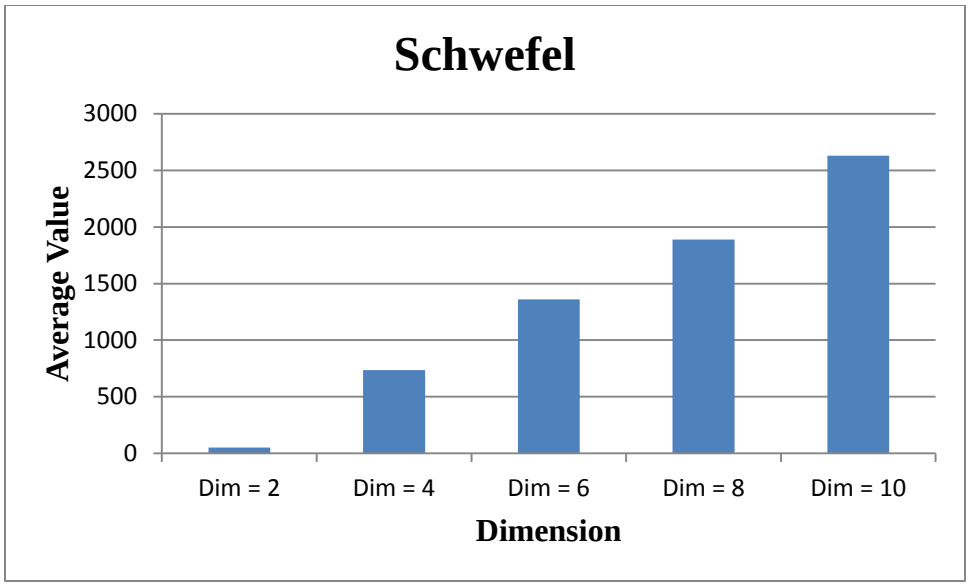

Figure 8: Mean values for Schwefel against dimension

The variation of the mean versus augmenting dimension value for Schwefel is shown in Figure 8. The lowest mean value 51.999 was achieved for dimension 2, and the highest mean value 2630.118 was obtained for dimension 10.

Tabular values and the graphs presented illustrate that the lower the dimension, the better the performance of the algorithm. This is a well-known behavior of optimization algorithms in general.

4.4.3. Test case 3: Simulations performed by varying number of generations

The following section describes the results obtained by varying the number of iterations and recording the best value for one single independent run. The simulations have been performed on five different functions and the results obtained are illustrated in the figures below.

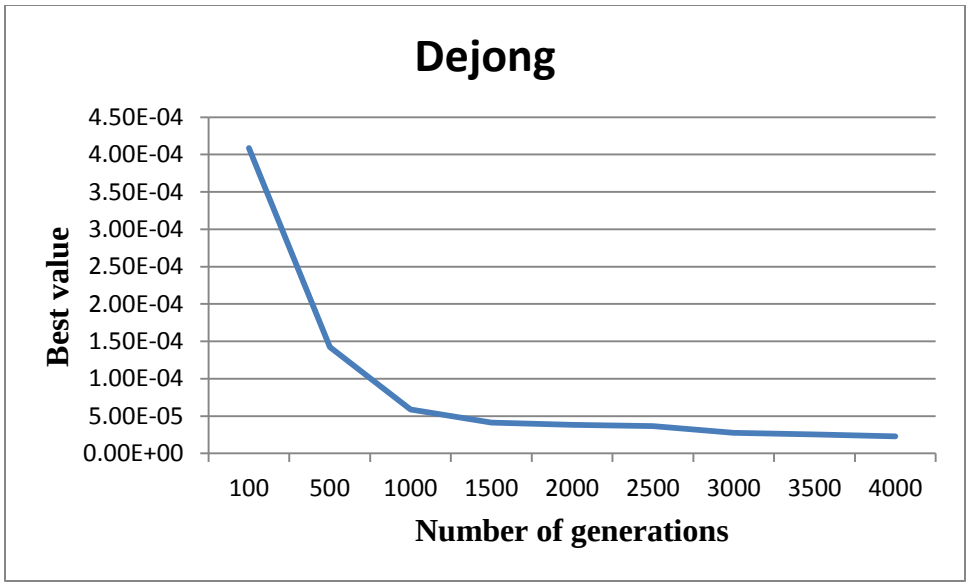

Figure 9: Variation of best value with number of generations for De Jong

The variation of best objective value versus number of iterations for De Jong function is illustrated in Figure 9. The number of iterations has been varied between 100 and 4,000. The best objective value varied from 4.1E-04 to 2.28E-05.

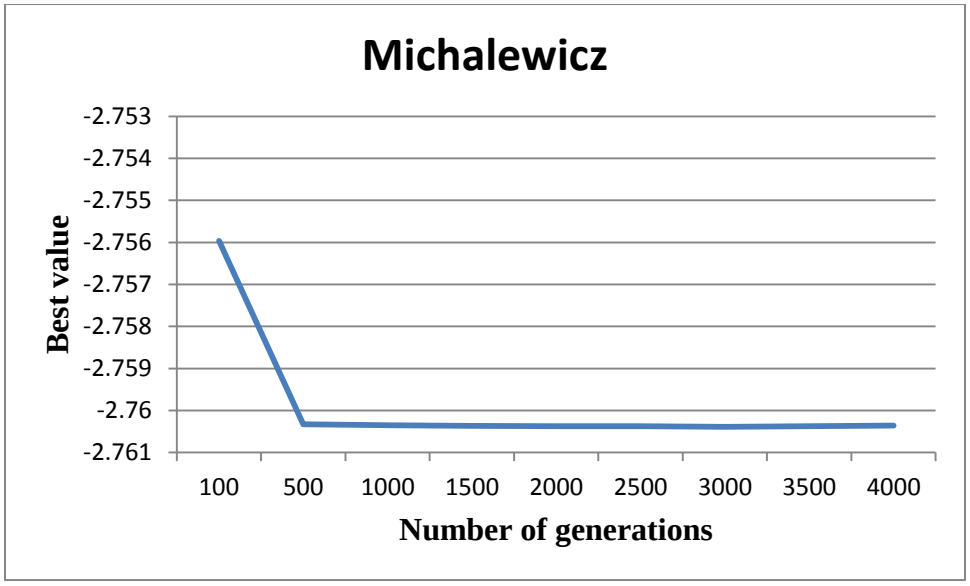

Figure 10: Variation of best value against number of generations for Michalewicz

The variation of best objective value versus number of iterations for Michalewicz function is illustrated in Figure 10. The number of iterations has been varied between 100 and 4,000. The best objective value varied from -2.756 to -2.7604.

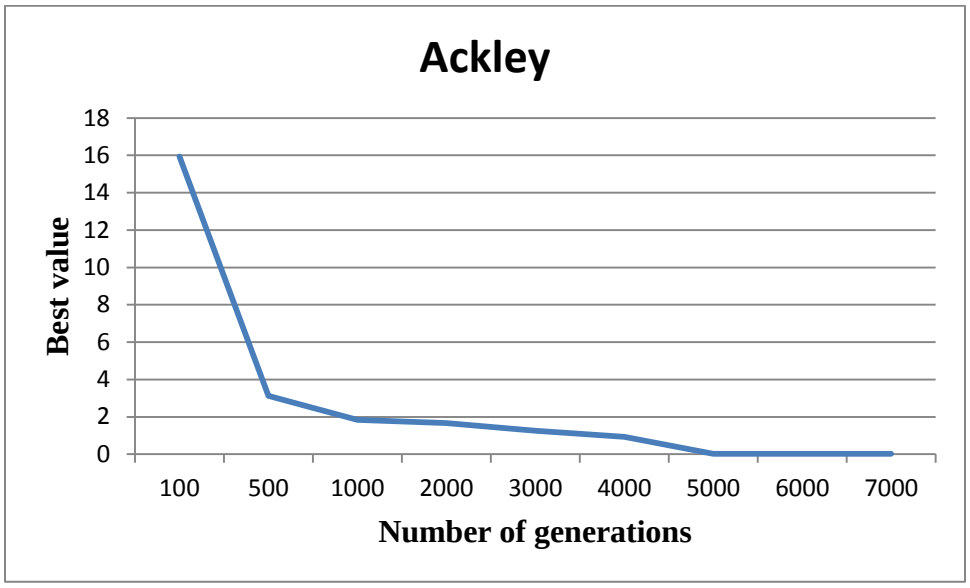

Figure 11: Variation of best value with number of generations for Ackley

The variation of best objective value versus number of iterations for Ackley function is illustrated in Figure 11. The number of iterations has been varied between 100 and 7,000. The best objective value varied from 15.9379 to 0.01741.

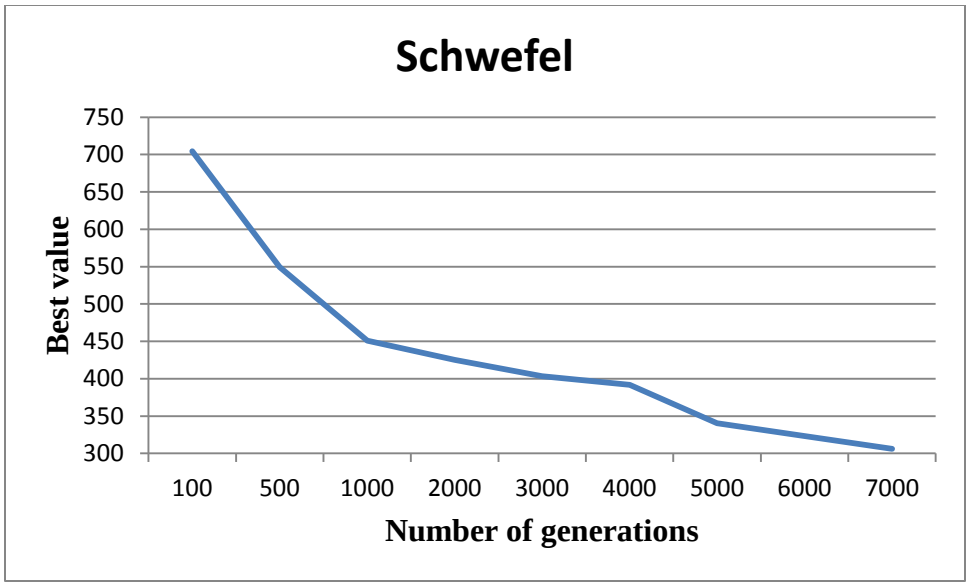

Figure 12: Variation of best value with number of generations for Schwefel

The variation of best objective value versus number of iterations for Schwefel function is illustrated in Figure 12. The number of iterations has been varied between 100 and 7,000. The best objective value varied from 704.3218 to 305.9326.

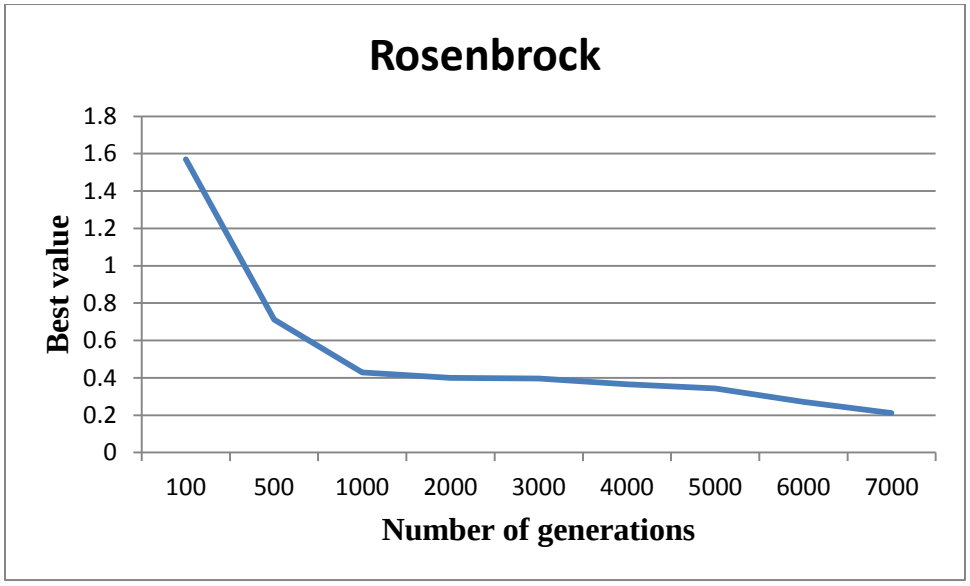

Figure 13: Variation of best value with number of generations for Rosenbrock

The variation of best objective value versus number of iterations for Rosenbrock function is illustrated in Figure 13. The number of iterations has been varied between 100 and 7,000. The best objective value varied from 1.56967 to 0.21194.

The plots clearly show that the best value decreases, that is, the performance of the algorithm increases with increasing number of iterations.

4.4.4. Test case 4: Simulations performed by varying number of generations against the average value

Five functions were evaluated by the firefly algorithm on a constant number of dimensions, by varying the number of iterations. Thirty independent runs for each function have been made, the average over the 30 runs has been calculated and the results have been tabulated.

Table 9: Varying number of iterations and calculating average value

Function Name	Average Result			
	Generations =1,000	Generations =10,000	Generations =100,000	Generations =1,000,000
Ackley	5.090657	4.266571	4.096618	4.055079
Rosenbrock	0.726725	0.420028	0.330758	0.238947
Michalewicz	-2.73496	-2.74938	-2.73324	-2.73326
Dejong	0.000163	4.91E-05	1.68E-05	4.69E-06
Schwefel	724.5558	560.4997	339.9028	183.3132

The results tabulated have been plotted and have shown the following behavior:

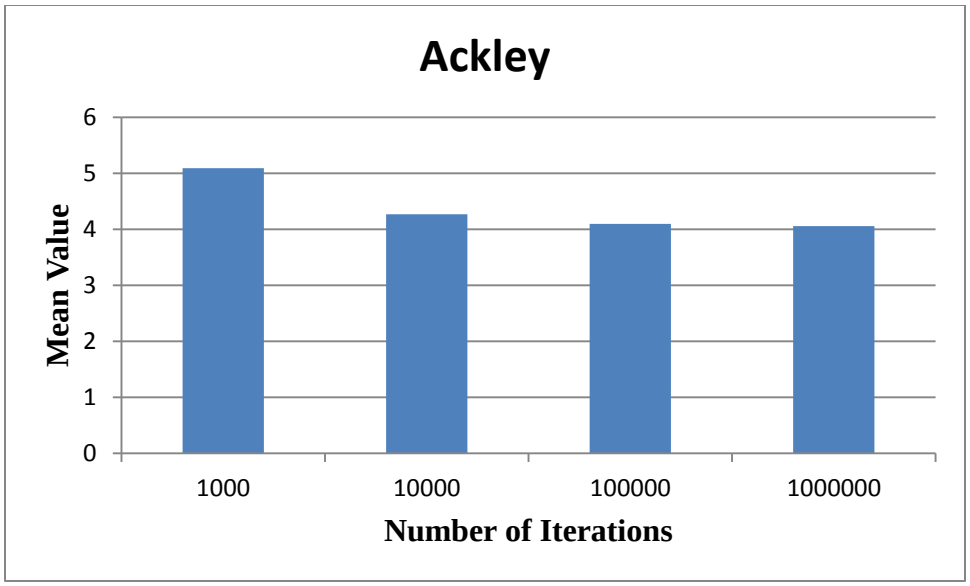

Figure 14: Variation of mean value for Ackley against increasing iterations

The variation of mean value versus the number of iterations for Ackley function is illustrated in Figure 14. The mean value has varied between 5.09 and 4.05; it has almost remained constant for 100,000 and 1,000,000 iterations.

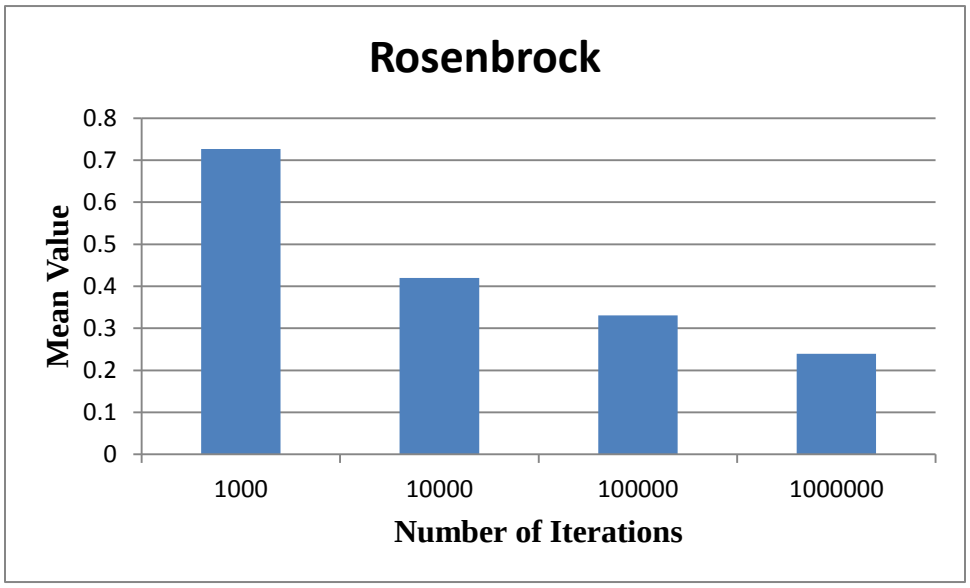

Figure 15: Variation of mean value for Rosenbrock against increasing iterations

The variation of mean value versus the number of iterations for Rosenbrock function is illustrated in Figure 15. The mean value has been calculated at 4 different points between 1,000 and 1,000,000. The mean value has varied between 0.7267 and 0.2389.

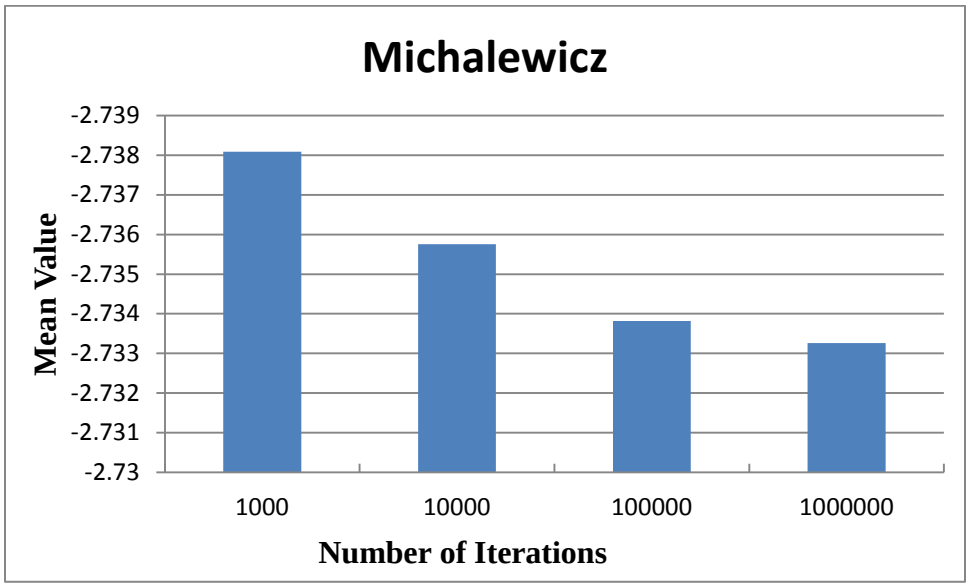

Figure 16: Variation of mean value for Michalewicz against increasing iterations

The variation of mean value versus the number of iterations for Michalewicz function is illustrated in Figure 16. The mean value has been calculated at 4 different points between 1,000 and 1,000,000. The mean value has varied between -2.7349 and -2.7332.

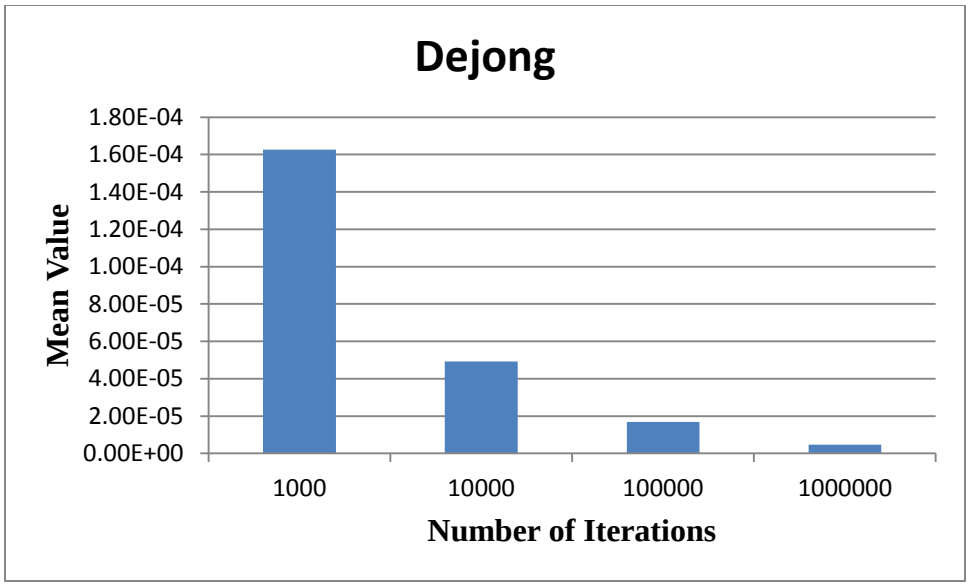

Figure 17: Variation of mean value for De Jong against increasing iterations

The variation of mean value versus the number of iterations for De Jong function is illustrated in Figure 17. The mean value has been calculated at 4 different points between 1,000 and 1,000,000. The mean value has varied between 1.6E-04 and 4.96E-06.

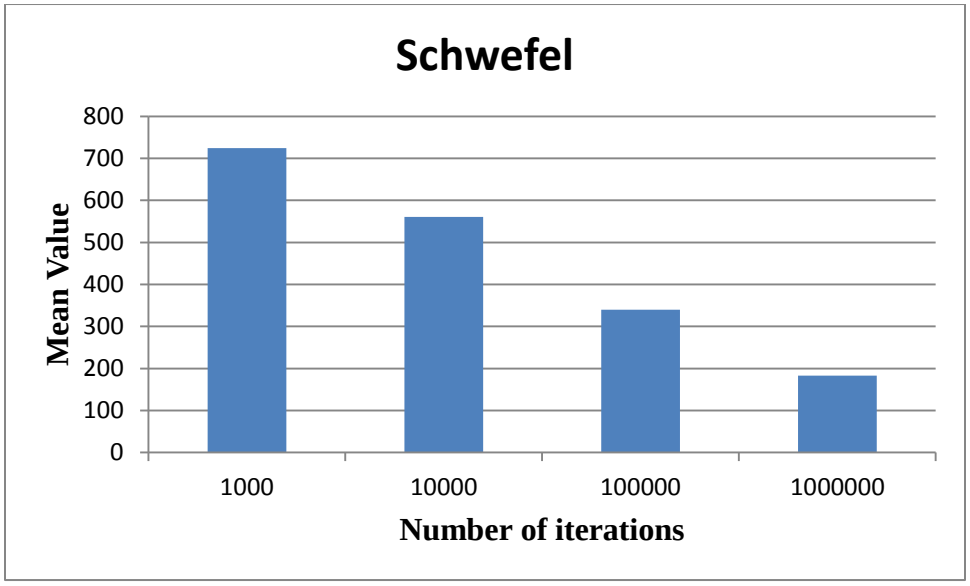

Figure 18: Variation of mean value for Schwefel against increasing iterations

The variation of mean value versus the number of iterations for Schwefel function is illustrated in Figure 18. The mean value has been calculated at 4 different points between 1,000 and 1,000,000. The mean value has varied between 724.5558 and 183.3132.

There is an improvement in performance of the algorithm with the increase in the number of iterations. The increased performance is larger in some functions than in other functions.

5. CONCLUSIONS AND FUTURE WORK

The firefly algorithm uses the process of attraction based on the brightness in fireflies to optimize an objective function. Prior research has shown that the algorithm can solve both continuous and discrete optimization problems [22]. Given the algorithms capability to be useful in both continuous and discrete domain it can be extended to solve real world optimization problems.

In this paper, the firefly algorithm has been tested using five different benchmark functions both unimodal and multimodal in class. Three different test cases have been designed to test the algorithm using the functions.

In test case 1, the problem dimension has been varied and the average of the best values over 25 independent runs has been calculated. The variation of the mean value over varying dimension has also been calculated. It was observed that the algorithm performed better at lower dimension than higher dimension.

In test case 2, the number of iterations have been varied and the best value for one single independent run have been noted and it was observed that the algorithm performed better with increasing iterations.

In test case 3, the number of iterations has been varied and the mean of best values over 30 independent runs was calculated. The algorithm performed better with increasing number of iterations, which is the expected behavior.

The results obtained from the evaluation of the algorithm insinuated that the algorithm is not as efficient as opposed to particle swarm optimization.

In the future, we plan to incorporate the functionality of other algorithms to generate a hybrid firefly algorithm to improve the performance, as well as test the hybrid by changing

various input parameters and also extend its application to various real world optimization problems.

6. REFERENCES

[1] Dantzig, G. B. (1998). *Linear programming and extensions*. Princeton university press.

[2] Fletcher, R. (1987). *Practical methods of optimization*. John Wiley & Sons.

[3] Banga, J. R. (2008). Optimization in computational systems biology. *BMC systems biology*, *2*(1), 47.

[4] Held, S., Korte, B., Rautenbach, D., & Vygen, J. (2011). Combinatorial Optimization in VLSI Design.

[5] Kendall, G., & Mohamad, M. (2004, November). Channel assignment in cellular communication using a great deluge hyper-heuristic. In *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on* (Vol. 2, pp. 769-773). IEEE.

[6] Di Caro, G., Ducatelle, F., & Gambardella, L. M. (2005). AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks.*European Transactions on Telecommunications*, *16*(5), 443-455.

[7] Brabazon, A., & O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Berlin: Springer.

[8] Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.

[9] Eiben, A. E., & Smith, J. E. (2010). *Introduction to evolutionary computing* (Vol. 2). Berlin: Springer.

[10] Mitchell, M. (1998). An introduction to genetic algorithms (complex adaptive systems).

[11] Oliveira, F. R., & Neto, F. B. L. (2010). Flexible Dialogues in Decision Support Systems.

[12] Wloch, K., & Bentley, P. J. (2004, January). Optimising the performance of a formula one car using a genetic algorithm. In *Parallel Problem Solving from Nature-PPSN VIII* (pp. 702-711). Springer Berlin Heidelberg.

[13] Sadek, A. W., Smith, B. L., & Demetsky, M. J. (1997). Dynamic traffic assignment: Genetic algorithms approach. *Transportation Research Record: Journal of the Transportation Research Board*, *1588*(1), 95-103.

[14] Mehrotra, K., Mohan, C. K., & Ranka, S. (1997). *Elements of artificial neural networks*. the MIT Press.

[15] Fauske, K. (2006, December 7). Example: Neural network. Texample.net. Retrieved July 2013 from http://www.texample.net/tikz/examples/neural-network/.

[16] Sivanandam, S. N., Sumathi, S., & Deepa, S. N. (2007). *Introduction to fuzzy logic using MATLAB*. Springer.

[17] Łukasik, S., & Żak, S. (2009). Firefly algorithm for continuous constrained optimization tasks. In *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems* (pp. 97-106). Springer Berlin Heidelberg.

[18] Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications* (pp. 169-178). Springer Berlin Heidelberg.

[19] Mancuso, N. (2010, July 26). Hybrid firefly algorithm with simulated annealing schedule for continuous optimization problems. Code.google.com. Retrieved July 2013, from https://code.google.com/p/csc6810project/source/browse/#svn%2Ftrunk%2FFireFly.

[20] Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*.

[21] Khadwilard, A., Chansombat, S., Thepphakorn, T., Chainate, W., & Pongcharoen, P. (2012). Application of Firefly Algorithm and its parameter setting for job shop scheduling. *ยินดี ต้อนรับ สู่ วารสาร ฯ*, *8*(1), 49-58.

[22] Sayadi, M. K., Hafezalkotob, A., & Naini, S. G. J. (2012). Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. *Journal of Manufacturing Systems*.