THE TITLE OF THE MTP

MTP Report submitted to

Indian Institute of Technology Mandi

for the award of the degree

of

B. Tech

by

VARUN BANSAL

under the guidance of

Siddhartha Sarma

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

June 2018

CERTIFICATE OF APPROVAL

Certified that the thesis entitled **BIO INSPIRED SOLUTIONS TO OPTIMIZATION PROBLEMS**, submitted by **VARUN BANSAL**, to the Indian Institute of Technology Mandi, for the award of the degree of **B. Tech** has been accepted after examination held today.

Date : 30 May, 2018

Mandi, 175001

Faculty Advisor

CERTIFICATE

This is to certify that the thesis titled **BIO INSPIRED SOLUTIONS TO OPTIMIZATION PROBLEMS**, submitted by **VARUN BANSAL**, to the Indian Institute of Technology, Mandi, is a record of bonafide work under my (our) supervision and is worthy of consideration for the award of the degree of **B. Tech** of the Institute.

Date : 30 May, 2018

Mandi, 175001

Supervisor(s)

DECLARATION BY THE STUDENT

This is to certify that the thesis titled **BIO INSPIRED SOLUTIONS TO OPTIMIZATION PROBLEMS**, submitted by me to the Indian Institute of Technology Mandi for the award of the degree of **B. Tech** is a bonafide record of work carried out by me under the supervision of **DR. SIDDHARTHA SARMA**. The contents of this MTP, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Date : 30 May, 2018

Mandi, 175001

Varun Bansal

Acknowledgments

The final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **Dr. Siddhartha Sarma**, for providing me an opportunity to do the project work and giving us all the support and guidance which made me complete the project duly. I am extremely thankful to him for providing his support and guidance.

Also, I would like to thank the committee members for the project **Dr. Varun Dutt** and **Dr. Satinder Kumar Sharma** for their useful feedback throughout the project.

I would also like to thank my project partner **Rajat Mehra** for his role in the project.

Varun Bansal

Abstract

Finding good solutions for NP Hard problems is a highly researched area, as most of these problems have many real life applications. Bio-inspired meta-heuristics like Genetic Algorithm and Ant Colony Optimization for these NP Hard optimization problems like the Travelling Salesman Problem and Vertex Cover have proved to provide much better results in reasonable amount as compared to algorithms based on other techniques.

In this project, we focus mainly on Firefly Algorithm which is one of the less studied bio-inspired algorithm and apply the algorithm on some of these optimization problems and thereby analyze the performance by comparing the results with other well known algorithms. We first try the algorithm on an unconstrained continuous optimization function to gain an understanding of the algorithm and it's parameters, and later move on to the standard TSP (Travelling Salesman Problem) and modify the firefly algorithm to incorporate the discrete nature of the TSP and compare the results with an implementation of the well known Ant Colony Optimization Algorithm. Finally, we study the TDMA (Time Division Multiple Access) Scheduling Problem which can be considered as a modified version of the standard Graph Coloring problem and compare the performance of our implementation of the firefly algorithm with an implementation of the Maximum Independent Set based approach for the problem.

Keywords: *NP Hard Problems, Bio-inspired meta-heuristics, Genetic Algorithm, Ant Colony Optimization, Travelling Salesman Problem, Vertex Cover, Firefly Algorithm, Optimization Problems, TDMA Scheduling, Maximum Independent Set*

Contents

Abbreviations

TSP	- Travelling Salesman Problem
FA	- Firefly Algorithm
ACO	- Ant Colony Optimization
TDMA	- Time Division Multiple Access
EA	- Evolutionary Algorithms
GA	- Genetic Algorithm
PSO	- Particle Swarm Optimization
SI	- Swarm Intelligence

Symbols

α	- Randomization Parameter for firefly
β	- Attractiveness of a firefly
γ	- Absorption coefficient in Firefly Algorithm
ϵ	- Penalty for number of colors in firefly algorithm for graph coloring
δ	- Penalty for number of conflicts in firefly algorithm for graph coloring

List of Tables

List of Figures

Chapter 1

Introduction

Optimization problems are the problems involving maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation. The function allows comparison of the different choices for determining which might be best. The scope of this project is restricted to a specific sub-class of optimization problems that have been a topic of research for the past few decades namely **NP hard problems**. Some common examples of NP hard problems include Vertex cover (finding minimum sized subset of vertices in a graph such that at-least one of the end points of every edge belongs to the subset) and Independent set (finding the maximum sized subset of vertices in a graph such that no edge exists between any 2 vertices in the subset). NP hard problems are an important area of research as many of the real world optimization scenarios can directly be mapped to them, yet no polynomial time algorithm is known to exist to solve these problems. Since, exact optimal solution to these set of problems can not be found in reasonable amount of time, computer science researchers have come up with certain heuristics that can be used to find near optimal answers.

Nature-inspired meta heuristics are currently among the most powerful tools for optimization of many NP-hard combinatorial problems. These methods are based on existing mechanisms of a biological phenomenon of nature. Both simplicity and efficiency have attracted researchers towards these natural phenomenon, resulting in some popular algorithms. The project aims at exploring one of the lesser explored nature inspired algorithm [1], namely the **firefly algorithm** and thereby use it to solve some real world optimization problems.

Several real-world applications require distributed approaches for finding near optimal values. Also, there are problems that can be casted as multi-objective optimization problems. Many of these problems are difficult to solve using traditional optimization techniques. However, bio inspired algorithms can be used to obtain a pragmatic solution within reasonable time limit. In this project, we will explore the efficiency of firefly algorithm over other bio inspired solutions for solving some optimization problems.

1.1 Continuous Function

The first part of the project involves trying out the firefly algorithm on a simple continuous function to get an idea about the performance and parameters involved in the algorithm. The major objective of this sub-task is to gain familiarity with the algorithm and have a basic working program of the firefly algorithm which in future can be used as a black box to solve more complex real life problems later in the project.

1.2 Travelling Salesman Problem

The next part involves the implementation of firefly algorithm for the well known **TSP (Travelling Salesman Problem)** [2] problem and comparing the results with some other bio-inspired algorithm. We use **Ant Colony Optimization** algorithm for the comparison as it is known to produce better results than most other algorithms for this problem.

One of the most widely studied and researched algorithmic optimization problem in the field of computer science and operational research is the Travelling Salesman Problem. The problem is that of a salesman and a given set of cities he needs to visit. Each city is connected to every other city by a link. Each of those links between the cities have some cost attached to it. The cost describes how difficult it is to traverse this edge on the graph, and may be given, for example, by the length of the link. The main aim of the salesman is to keep the total cost of his journey as low as possible.

Formally, given n cities, and an $n*n$ cost matrix C where $\mathbf{C_{ij}}$ denotes the cost between the i-th and the j-th city, find a permutation of the cities, determining a minimum distance circuit passing through each vertex once and only once. Such a circuit is known as a tour or Hamiltonian cycle. i.e., finding a permutation A of i, $1 \leq i \leq n$ such that $\sum_{\mathbf{j=1}}^{\mathbf{n-1}} \mathbf{C_{A[j]A[j+1]}} + \mathbf{C_{A[n]A[1]}}$ is minimized.

As an example if we consider the graph shown in Figure 1.1 as the input instance

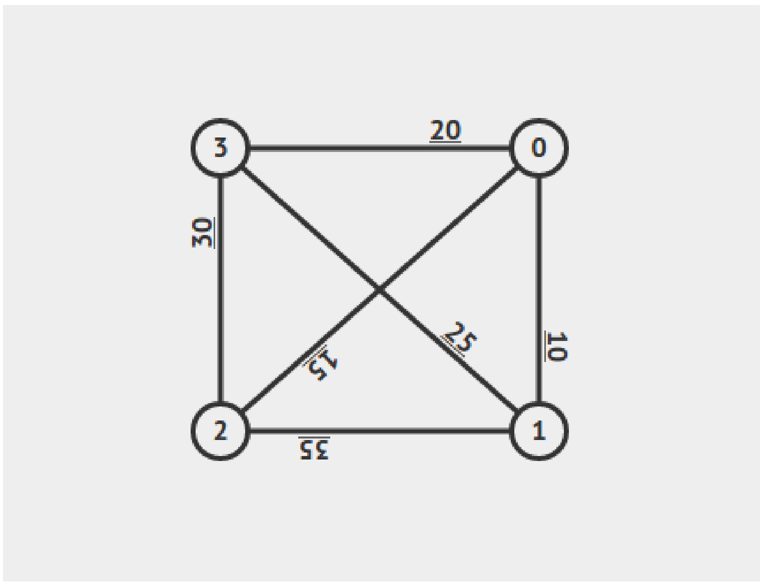

Fig. 1.1: A graph instance for TSP problem

Table 1.1 shows the cost matrix for the input instance. The optimal tour for this input instance is 0-1-3-2-0 with a total cost of 80.

City	0	1	2	3
0	0	10	15	20
1	10	0	35	25
2	15	35	0	30
3	20	25	30	0

Table 1.1: Cost Matrix for graph in Figure 1

1.3 TDMA Scheduling for Sensor Networks

The last part of the project involves solving the problem of TDMA **(Time Division Multiple Access)** scheduling for sensor networks. This is a famous optimization problem in the networking domain. The problem is based on a situation where multiple clients needs to utilize a same channel for their communication.

TDMA allows for shared transmission medium based on time slots. In TDMA multiple transmitters can send packets at the same time provided their transmission does not cause collision of packets. [3]

The TDMA scheduling problem is to determine the minimum number of time slots so that all the nodes can transmit their packets without causing any loss of packets. The nodes that conflict with each other are determined by the construction of a conflict which is different from the network graph. [4]

1.3.1 3-Coloring

A sub-part to the work on TDMA Scheduling involves the implementation of the firefly algorithm for the 3-coloring problem [5] which is a specific case for the well known k-coloring problem [6]. The problem involves coloring each node of an input graph with one of three colors in such a way that there is no edge in the graph with both the end points colored with the same color.

Formally, given an undirected, unweighted graph $G(V, E)$, assign each $v \in V$ an element of the set $\{0, 1, 2\}$, such that there is no $e(a, b) \in E$ such that a and b are assigned the same element.

For Example, if we consider the graph shown in Figure 1.2, one of the possible color assignment can be as shown in Table 1.2.

To convert this problem into an optimization problem, we try to minimize the number of conflicts in the above mentioned assignment wherein a conflict is defined as an edge $e(a, b) \in E$ with a and b having same color assignment.

Node	Color
0	0
1	1
2	2
3	1
4	0

Table 1.2: Color Assignment for graph in Figure 1.2

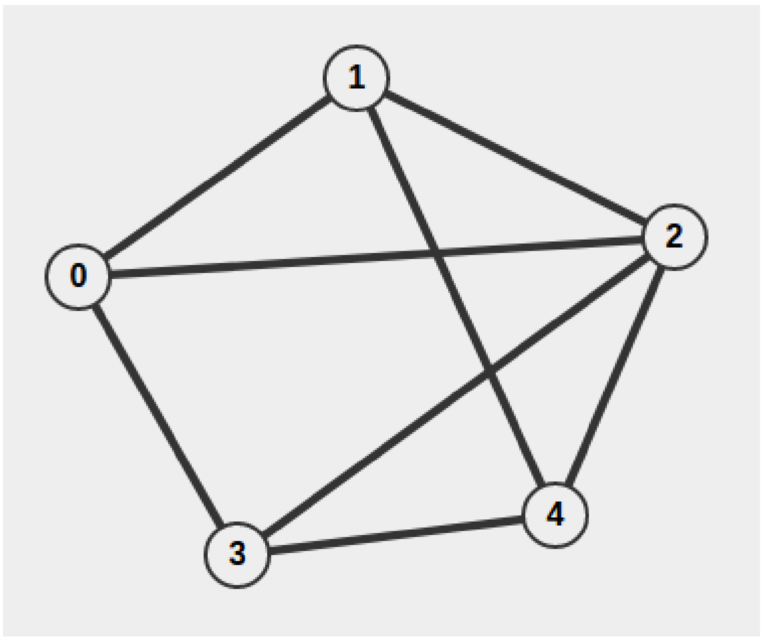

Fig. 1.2: A graph instance for 3-coloring problem

1.3.2 General Graph Coloring Problem

The next sub-part to the work of TDMA Scheduling involves implementing the firefly algorithm for the general graph coloring problem [7] which involves finding the optimal coloring in an undirected graph. Just like in the case of 3-coloring, colors have to be assigned to nodes in such a manner that there is no edge in the graph whose end points share the same color. In this case, rather than checking whether there exists a 3-coloring in the graph, the problem involves finding the minimum value of k such that a k-coloring exists in the graph.

Formally, given an undirected, unweighted graph $G(V, E)$, assign each $v \in V$, a color, such that there is no $e(a, b) \in E$ such that a and b are assigned the same element and the total number of colors used are minimized.

The best possible color assignment for the graph shown in Figure 1.3 is shown in the Table 1.3. As is evident from the table, the minimum value of k which there exists a k-coloring is 3.

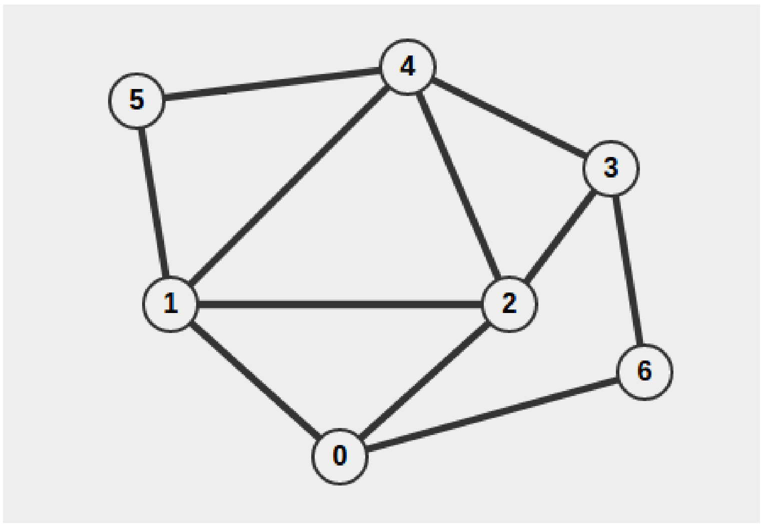

Fig. 1.3: A graph instance for graph coloring problem

Node	Color
0	1
1	2
2	0
3	2
4	1
5	0
6	0

Table 1.3: Color Assignment for graph in Figure 1.3

Chapter 2

Background and Related Work

2.1 Background

The goal of an optimization problem is either to minimize or maximize a particular objective function. The use of randomized algorithms has been propelled by the fact that most of the optimization problems are very complex.Most of the randomized algorithms are based on some heuristic information that is extracted from nature.Some of the most common examples of bio inspired algorithms include Ant Colony Optimization(ACO), Evolutionary Algorithms(EA) etc. These algorithms have been widely used in optimization problems pertaining to different domains like computational biology, engineering and telecommunications. These algorithms have achieved a lot of success when applied to these optimization problems.

Most of the bio inspired algorithms makes use of the behavior of swarm intelligence. **Swarm intelligence** [8] is based on the interaction between the organisms despite having any central structure governing their movement.These organisms use simple rules dictating their behaviour. Their behavior is not completely deterministic and also includes a random component. Despite such simple structure, these interactions lead to the emergence of an intelligent global behaviour which is not known to any individual organism.

Some examples of swarm intelligent system in nature include ant colonies, birds flocking, bacterial growth etc. Various algorithms have been developed that use swarm intelligence. **Example** - Ant Colony Optimization [9], Bee Optimization [10], Particle Swarm

Optimization [11].

Among the various swarm intelligent systems, firefly algorithm [12] has been known to perform better than most others. The firefly algorithm uses the phenomenon of difference in the light intensities emanated by the fireflies. This helps the firefly to move towards brighter and more attractive locations in order to obtain optimal solutions. To avoid convergence at local optimal values in the solution space, a random component is also added to the movement of the fireflies. The combined effect of both these components namely exploitation and exploration have been known to produce optimal results reasonably quickly.

From our general knowledge of physics we all know that light intensity, I is inversely proportional to the squared distance of the distance between the source and the point of observation. i.e. $I \propto 1/r^2$ [13]. Also, air absorbs some amount of light as light passes passes through it. Due to these factors, fireflies can be seen only till a certain distance.

For simplicity, the following simplifying assumptions are made while using the firefly algorithm: [12]

1. There is no concept of sex within this population of fireflies which allows for every firefly to get attracted towards every other firefly.

2. The brighter a particular firefly is, more will the other fireflies get attracted to it. The level of attraction of a firefly towards another firefly reduces with increase in the distance between them. A firefly will in random directions if there is no firefly more attractive than this firefly.

3. The value of the objective function is what governs the brightness of a firefly. If the problem is a maximization function, the value of the objective function can itself be considered as brightness or a value proportional to it.

2.2 Related Work

Graph Coloring Problem which is an NP Hard Problem is a highly studied problem, and many researchers have tried to find good solutions for the problem using various classes of algorithms including constructive heuristics [14], local search heuristics [15], and meta-heuristics although most studied among them are the metaheuristics based approaches, since they are believed to produce the best results.

Previously, Ehsan Salari and Kourosh Eshghi [16], in their paper on Graph coloring in the year 2008 have provided an Ant Colony Optimization based algorithm for finding the graph coloring. Also, Vincenzo Cutello, Giuseppe Nicosia, and Mario Pavone [17] have provided a Hybrid Immune Algorithm with Information Gain for the graph coloring problem. Another similar research is done by M.Chams, A.Hertz, and D.de Werra [18] in which they have experimented with simulated annealing based approach for finding solutions to the problem.

Sinem Coleri Ergen and Pravin Varaiya in two different papers [19] [20] have studied TDMA scheduling algorithms for Sensor Networks. They have proposed two algorithms for the problem both of which are based on heuristics.

Chapter 3

Work Done

As mentioned in the previous sections, we have worked on three different optimization problems which included an unconstrained continuous function, Travelling Salesman Problem and the TDMA scheduling problem and implemented the firefly algorithm for each of them and compared the results with other algorithms. In the following sections we describe in detail about the work done specific to these problems.

3.1 Continuous Function

We first implemented the basic version of firefly algorithm for a trivial continuous function and gaining familiarity with the algorithm and the different parameters involved and how these parameters can be tweaked to produce optimal results in less number of iterations. For this purpose, $f(x_1, x_2, x_2) = -(x_1^2 + x_2^2 + x_3^2)$ was chosen as the objective function. The reason for the choice of such a function was the known global optima point, because of which it becomes easy to get an estimate of the efficiency of the algorithm.

To check the convergence of fireflies, the program was run for different parameters and the number of fireflies that ended up in the vicinity of the optimal point were recorded. The results of these experiments can be found in chapter 4, section 4.1.

3.2 Travelling Salesman Problem

This part of the project involved implementing the firefly for the Traveling Salesman Problem and comparing the results with the implementation of Ant Colony Optimization. For this task, it was decided that I will implement the firefly algorithm for TSP and my teammate, Rajat was given the responsibility to implement the ACO for the same and then the results of both were compared.

The mapping of TSP to a problem instance for firefly algorithm caused the following challenges:

- **Representation of the state and intensity measure of fireflies:** For this, we decided to represent the state of a firefly by a permutation of cities, which uniquely determines a valid solution for the problem instance and the intensity measure of a firefly was decided to be the inverse of the total cost incurred by the tour corresponding to that firefly. The motivation behind this was to convert the minimizing problem of TSP into a maximizing function as the firefly algorithm is designed to achieve states which correspond to maximum intensity.

- **The notion of distance between two fireflies:** For this, it was decided to take the hamming distance between the permutations corresponding to the fireflies as the distance measure. The hamming distance between two permutations is defined as the number of positions at which the corresponding values are different. In other words, it measures the minimum number of substitutions required to change one permutation into the other. For example, the hamming distance between the permutations 3**142** and 3**241** is 2 and the numbers marked in bold contribute to the hamming distance.

- **Movement of a firefly towards another firefly:** For the case when we need to move the firefly A towards another firefly B, we keep the cities which are in the same position in the representation of both the fireflies fixed. Further, we generate a random number x between 2 and the calculated hamming distance and replaced x number of cities from A by the cities at corresponding positions in B and the remaining cities were placed randomly.

To further improve the exploration of the search space by the algorithm, in each iteration we generate m different fireflies corresponding to each movement of firefly. In this manner, we generate $k * m$ fireflies after each iteration if we have k fireflies at the start of the iteration and then we pick the best k out of these fireflies. This introduces a new parameter m into the algorithm.

After the implementation phase of the algorithm, various parameters involved in the program namely the number of iterations, number of fireflies, m (number of fireflies to generate for each movement) and Gamma (light absorption coefficient) were tweaked to get some idea about the optimal values of the parameters and the results were recorded for mainly two data-sets from TSPLIB [21]. Further, the results obtained from both the algorithms, Ant Colony Optimization [9] and Firefly Algorithm were compared. For this both the programs were for a number of different combinations of parameter values and best set of values were recorded and compared. These results and the corresponding plots can be found in Chapter 4.

3.3 TDMA Scheduling Problem

Since the problem instance of TDMA scheduling can be easily mapped to an instance of graph coloring problem, we first worked on implementing the algorithm for a very restricted version of graph coloring problem and later generalized for the general coloring problem.

3.3.1 3-Coloring Problem

For this purpose we first define a set of terminologies to be used. We define the set of fireflies as X where each $X_i \in X$ represents a candidate solution i.e. each X_i is an n dimensional vector where the j-th element, X_{ij} represents the color assigned to the j-th node in the graph and takes a value from the set $\{0,1,2\}$ each of the numbers 0, 1 and 2 representing a different color. For the purpose of evaluation of a candidate solution X_k, we use a fitness measure as follows [22]:

$$Fitness(X_k) = 1 - \frac{conflict(X_k)}{m}$$

where m is the total number of edges in the graph and $conflict(X_k)$ measures the total number of conflicts in the candidate solution and is mathematically defined as follows:

$$conflict(X_k) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} conflict_{ij}$$

Here $conflict_{ij}$ is an indicator variable for each pair of vertices (i, j) which denote whether or not the pair of vertices (i, j) cause a conflict. Formally,

$$conflict_{ij} = \begin{cases} 1 & color_i = color_j \text{ and } (i, j) \in E \\ 0 & otherwise \end{cases}$$

Since the coloring problem is ultimately to be used to solve an instance of the TDMA problem, we first convert the input graph into an instance of conflict graph and then find the coloring assignment on the conflict graph. The conflict graph represents which pair of edges in the initial graph cannot be used simultaneously, which basically means the pair of edges which share a common end-point vertex. Formally, corresponding to every edge in the initial graph, there is a vertex in the conflict graph and corresponding to each pair of edges $e1$ and $e2$ which share a end-point vertex in the initial graph, there is an edge between the vertices corresponding to $e1$ and $e2$ in the conflict graph.

For example, if we consider the input graph shown in Figure 12, the conflict graph will look something like the one depicted in figure 13. The numbers written along the edges in the input graph represent the edge number. If we consider edge number 5 in the input graph, edges which have a common end-point with it are 0, 2, 3 and 4. Therefore, the vertex numbered 5 in the conflict graph has edges with vertices numbered 0, 2, 3 and 4.

Further, A measure of similarity is used for the purpose of difference between any two fireflies X_i and X_j and is defined as follows: [22]

$$Similarity_{ij} = 1 - \frac{H(X_i, X_j)}{n}$$

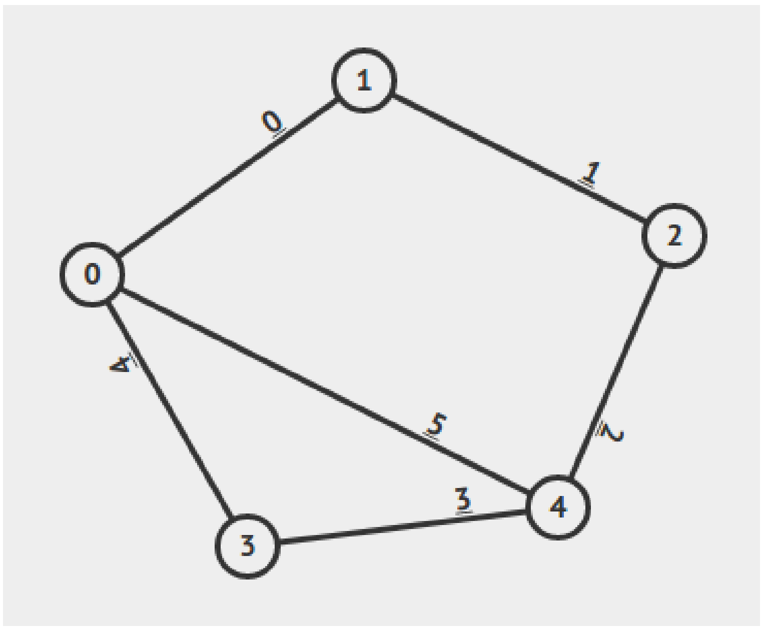

Fig. 3.1: Input instance for conversion to conflict graph

Here $H(X_i, X_j)$ is the hamming distance between the vectors X_i and X_j and n is the number of vertices in the graph. Further, the attractiveness between fireflies X_i and X_j is defined as:

$$\beta(X_i, X_j) = \beta_0 e^{-\gamma Similarity_{ij}^2}$$

where β_0 and γ are constant parameters. With the help of these similarity and attractiveness measures, the function for movement of a firefly X_i towards X_j becomes as follows: [22]

Algorithm 1 Move function for firefly algorithm

```
Function Move($X_i$ , $X_j$):
    calculate the Similarity($X_i$ , $X_j$)
    calculate the attractiveness beta
    for k in [1..n]:
        generate a random number r in range [0,1]
        if $r \leq$ beta:
            replace $X_i[k]$ with $X_j[k]$
        End if
    End or
```

The performance of the firefly algorithm were then compared to those of a greedy algorithm. I worked on the implementation of the firefly variant whereas my group mate, Rajat

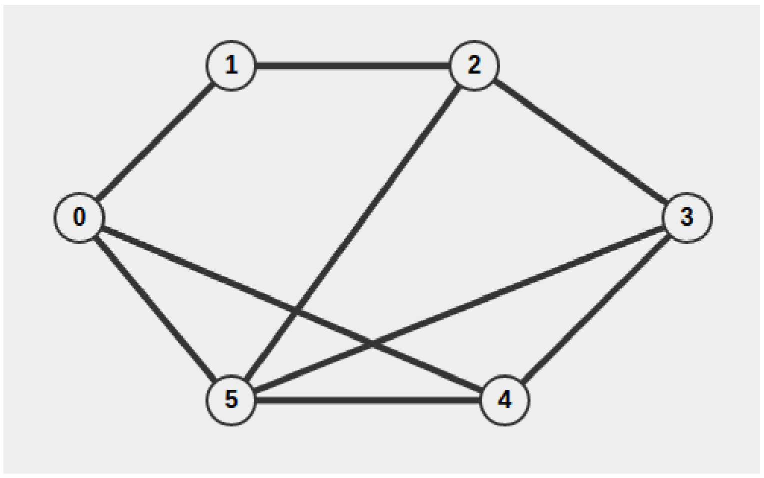

Fig. 3.2: Conflict graph for input graph in Figure 3.1

worked on the implementation of greedy algorithm for the same. The results of performance comparison can be found in the chapter 4.

3.3.2 General Graph Coloring Problem

This part involved converting an instance of TDMA scheduling problem into an instance of graph coloring and then implementing the firefly algorithm to come up with a solution for this graph coloring problem instance. In the general graph coloring problem, we optimize the number of colors required to color the graph unlike the case of 3-coloring problem where we were optimizing on the number of conflicting edges in the color assignment. Further, unlike the case of 3-coloring, the candidate solutions can have any number of colors, due to which the representation of a candidate solution has to be made more general. Also, the move function defined for 3-coloring solution can no longer be used in the same way, as now different candidate solutions can have different number of colors and moving a candidate solution in the direction of another would mean that the number of colors in the solution also changes.

To incorporate these requirements, we made several modifications to the algorithm to make it suitable for this problem

1. Since the fireflies can now have any number of colors, the initial population was generated in such a manner that different individuals of the population have different

number of colors. For this purpose, the number of colors of each candidate was decided by drawing a number from a uniform distribution in the range 2 to n (number of vertices in the graph).

2. Another major change was made in the fitness function as now the fitness function had to incorporate both, the number of colors in the solution and the number of conflicting edges. The new fitness function was taken as the weighted sum of the fitness measures corresponding to the number of colors the number of conflicts. Formally, the Fitness of a candidate solution X_k is defined as follows:

$$Fitness(X_k) = \epsilon * Color\_Fitness(X_k) + \delta * Conflict\_Fitness(X_k)$$

where ϵ and δ are parameters and Color_Fitness and Conflict_Fitness are defined as follows:

$$Color\_Fitness(X_k) = 1 - \frac{Colors(X_k)}{n}$$

$$Conflict\_Fitness(X_k) = 1 - \frac{conflict(X_k)}{m}$$

Here, Colors(X_k) denotes the number of colors in the candidate solution X_k, Conflict(X_k) denotes the number of conflicts in the solution as defined in the previous section, n denotes the number of vertices in the conflict graph and m denotes the number of edges in the conflict graph.

3. Both Alpha step and Beta step also had to be modified to incorporate different number of colors in different solutions. The movement steps were modified so as to try and reduce the number of conflicts by modifying those vertices which are causing a conflict.

I worked on the implementation of the beta step whereas Rajat worked on the alpha step. Algorithm 2 shows the pseudo code for the beta step which tries to move one firefly in the direction of other firefly and the extent of the movement is decided by the value of attractiveness between them. The value of attractiveness between the fireflies is defined by the following formula:

$$Attractiveness(X_i, X_j) = \beta_0 * \exp^{\gamma * similarity(X_i, X_j)^2}$$

Where, Similarity between the fireflies is defined as follows:

$$Similarity(X_i, X_j) = 1 - \frac{dist(X_i, X_j)}{n}$$

Here n is the number of vertices and dist is the hamming distance between the fireflies.

Algorithm 2 **Beta Step for movement one firefly in the direction of other**

```
function BETA_STEP(firefly_to_move, firefly_brigther, $\beta$0, gamma)
    attractiveness ← Attractiveness($X_i$,$X_j$)
    conflicts = $\emptyset$
    for e ∈ E do
        u ← e.first
        v ← e.second
        if u.color == v.color then
            conflict.append(e)
    for e ∈ conflict do
        r ← random no. between 0 and 1.
        if r ≤ attractiveness then
            node ← one end point of e
            firefly_to_move[node] ← firefly_brigther[node]
    Return firefly_to_move
```

Chapter 4

Exerimental Studies and Results

The results of various experiments performed for different sub-tasks during the project have been presented in the following sections.

4.1 Continuous Function

For the continuous function case, since we know the exact optimal value of the optimization function, we measure how many fireflies actually end up in the vicinity of the optimal value. The following graph shown in the figure 4.1 plots fraction of fireflies that end up within a distance of 0.1 units from the optimal value are plotted against the natural logarithm of the number of iterations performed for two different combinations of the parameters. In the first case, the value of the parameter alpha, which represents the randomization coefficient is kept to be 0.2, beta, which represents the attractiveness coefficient is kept at 0.5 and the number of fireflies is kept 100. In the other case, alpha is kept unchanged, beta is kept 0.3 and the number of fireflies is kept 50. As is evident from the graph, with the right choice of parameters, more than 90% of the fireflies converge near the optimal value within 100 iterations of the program. This gives a good understanding of how quickly can the firefly algorithm move towards optimal solutions

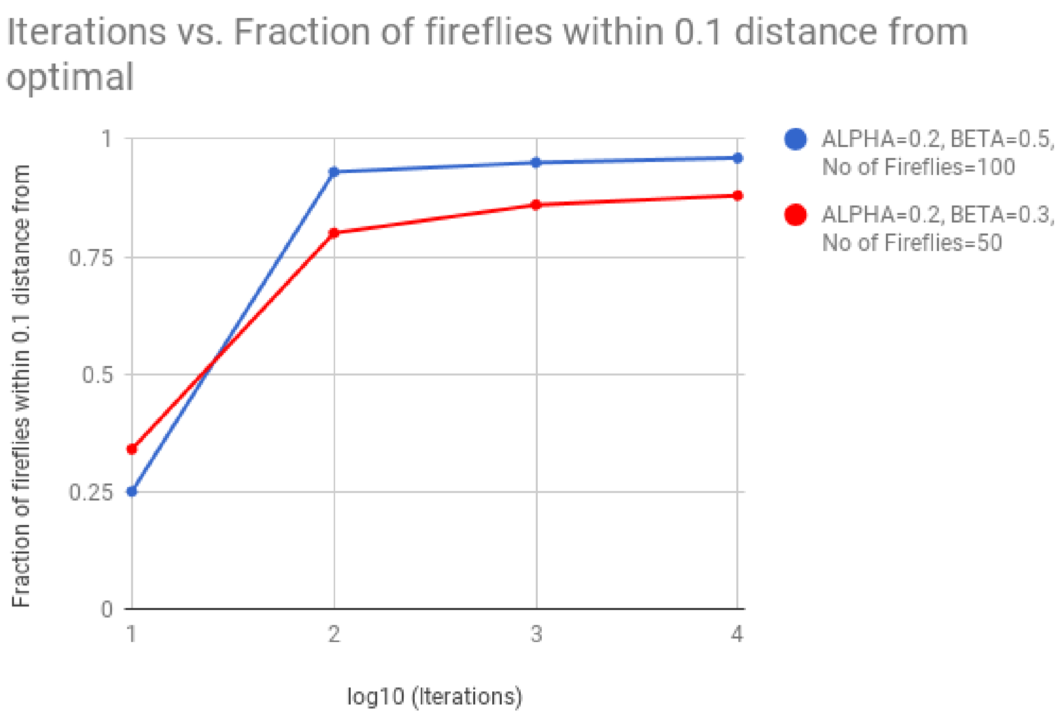

Fig. 4.1: log(No. of iterations) vs Fraction of fireflies that ended within a distance of 0.1 units from optima

4.2 Travelling Salesman Problem

For the purpose of testing the performance of the firefly algorithm on the Travelling Salesman Problem, we chose two standard data-sets namely tsp225 and tsp442 which are random graphs containing 225 and 442 vertices respectively. The graphs shown in Figure 4.2 to Figure 4.9 show the values achieved by our implementation of firefly algorithm by varying the different parameters involved in the algorithm namely, number of iterations, number of fireflies, number of neighbours(m) and gamma.

It is evident from these graphs that the optimal value achieved does not monotonically increase or decrease with any of the parameters, rather the best values are obtained with parameter values kept somewhere in between the range.

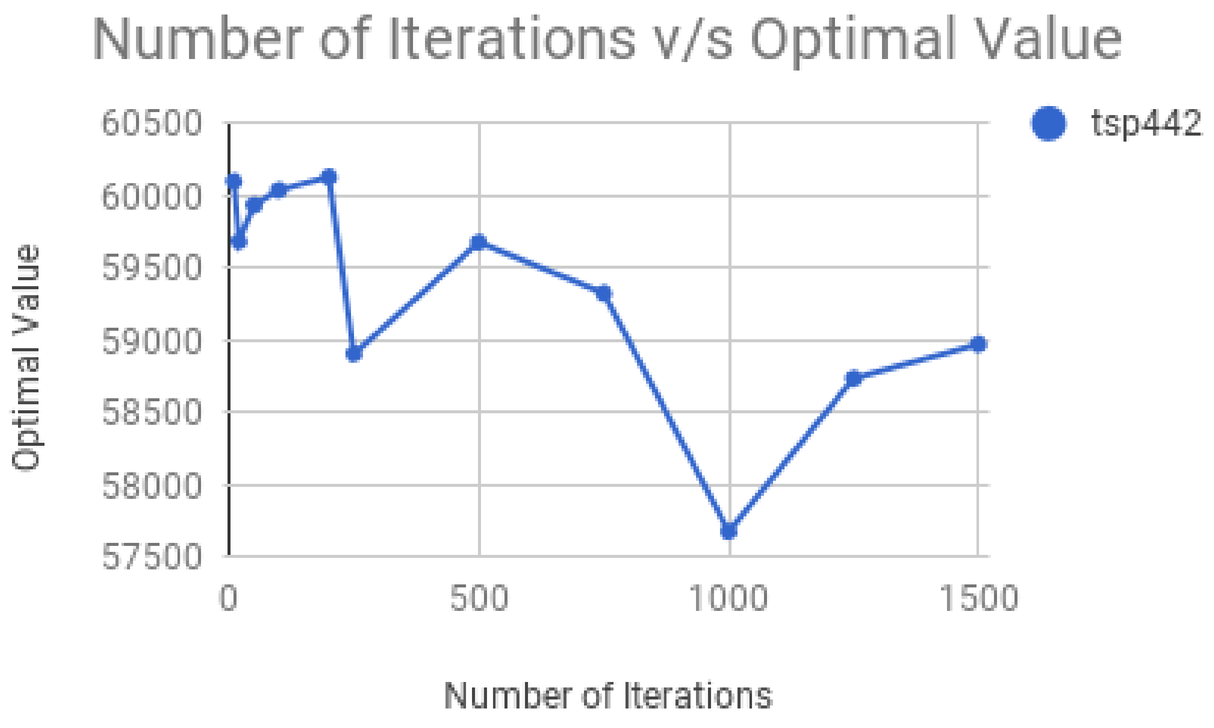

Fig. 4.2: Plot of Number of iterations vs Optimal value for tsp442 data-set

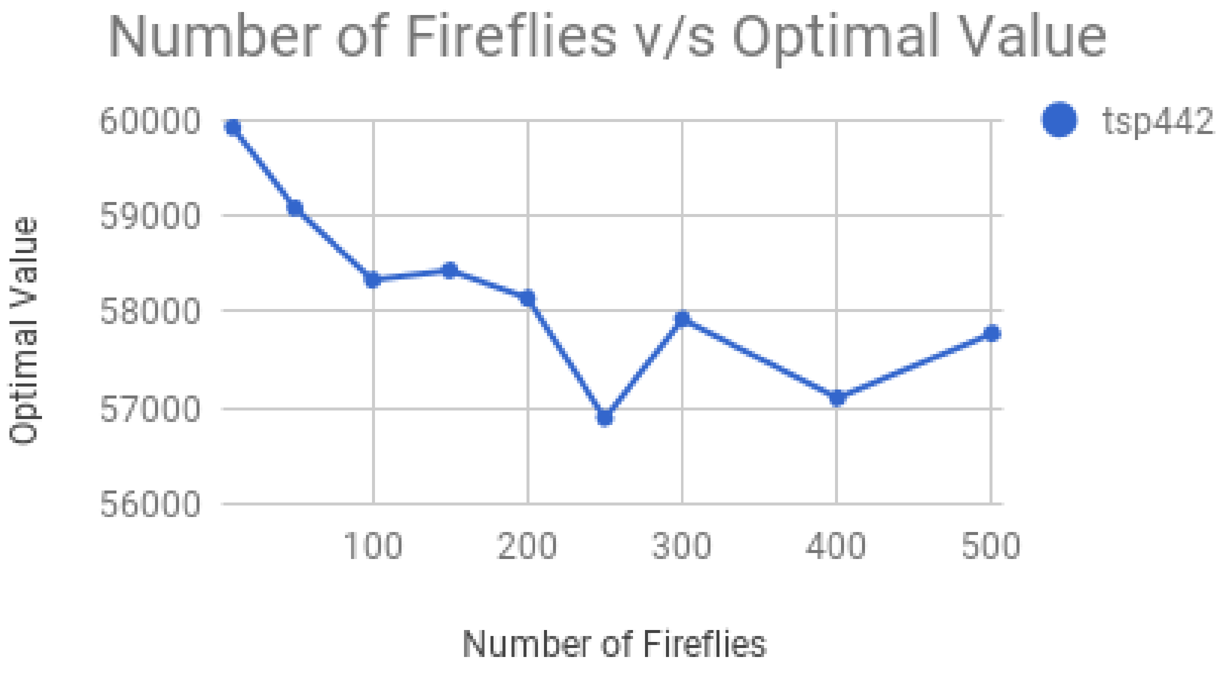

Fig. 4.3: Plot of Number of fireflies vs Optimal value for tsp442 data-set

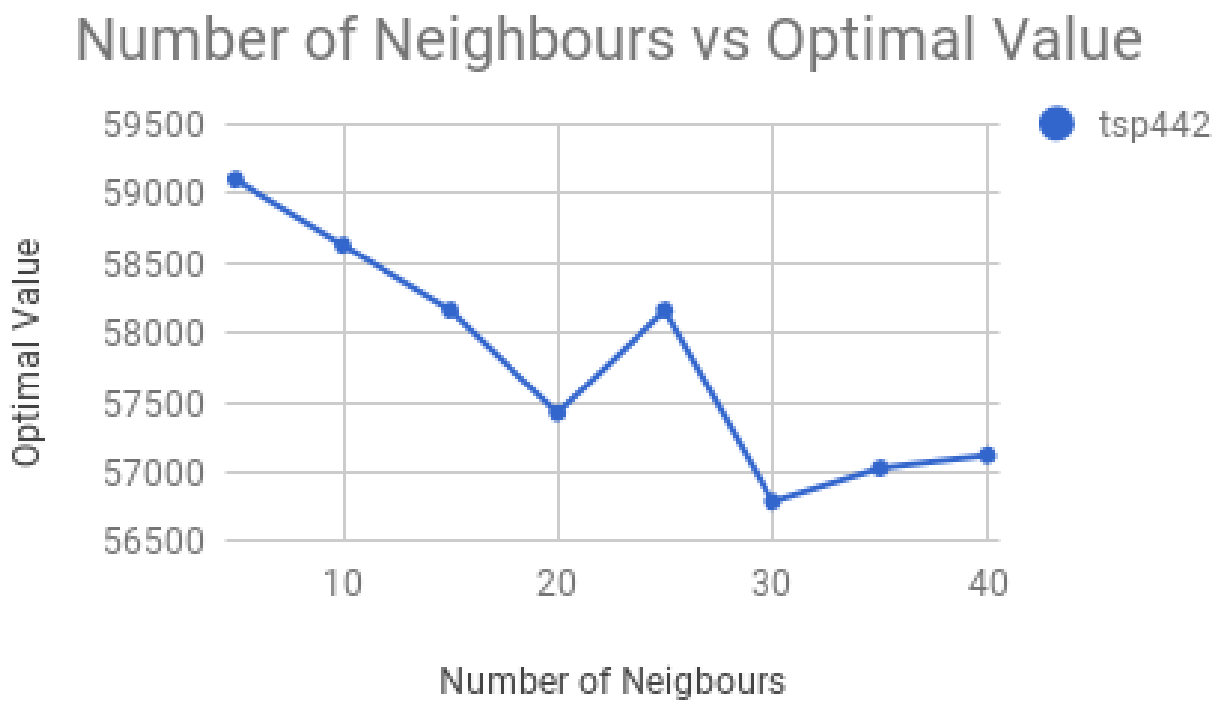

Fig. 4.4: Plot of Number of neighbours vs Optimal value for tsp442 data-set

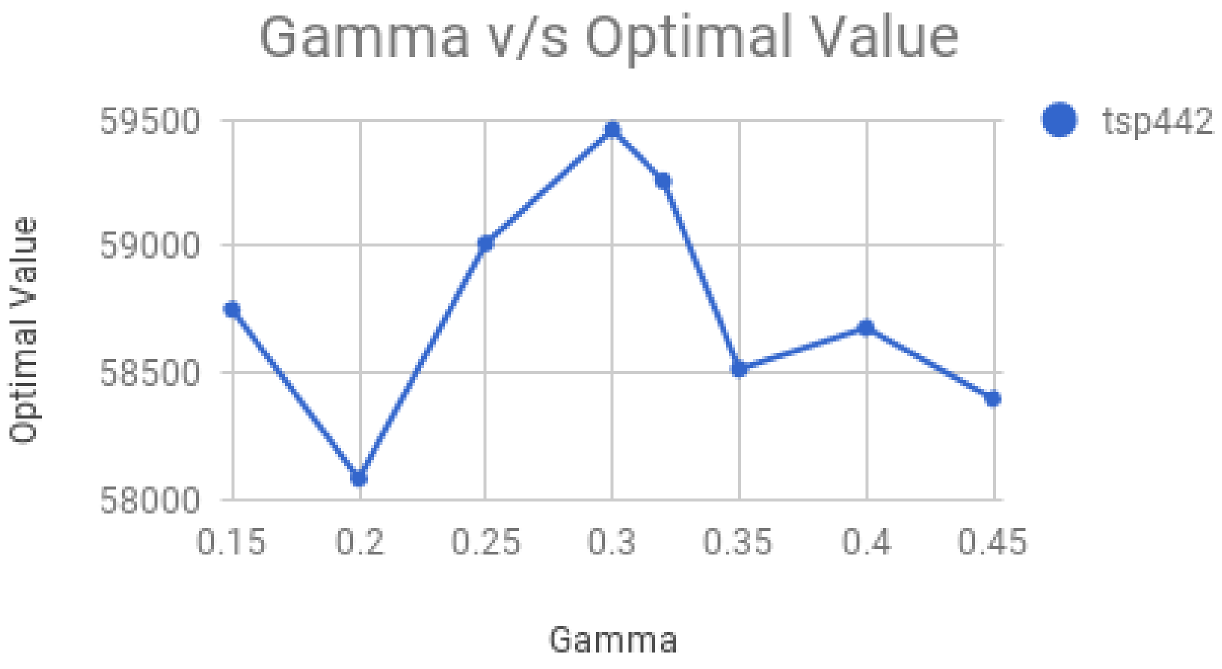

Fig. 4.5: Plot of Gamma vs Optimal value for tsp442 data-set

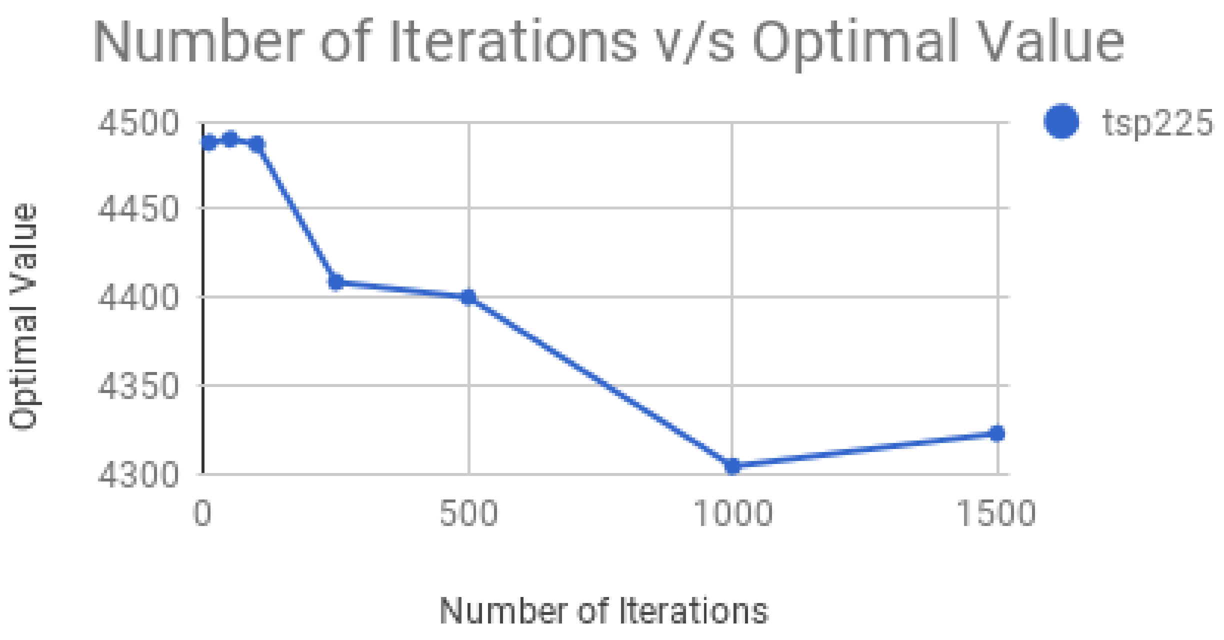

Fig. 4.6: Plot of Number of iterations vs Optimal value for tsp225 data-set

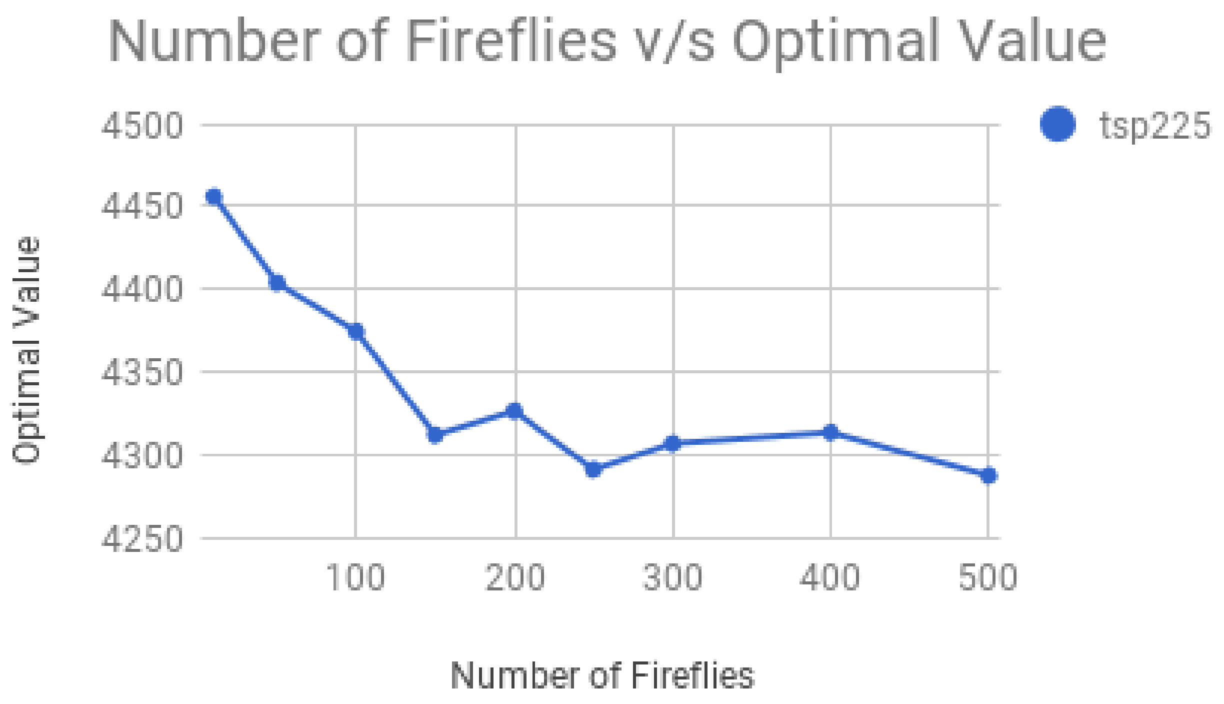

Fig. 4.7: Plot of Number of fireflies vs Optimal value for tsp225 data-set

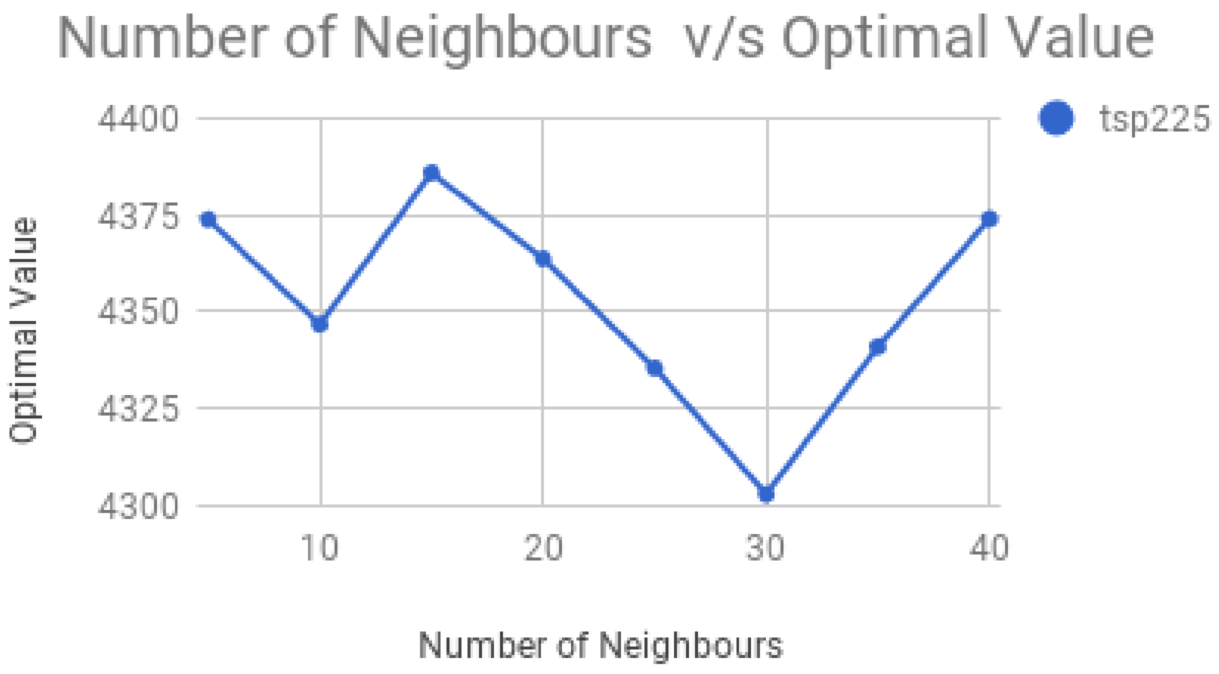

Fig. 4.8: Plot of Number of neighbours vs Optimal value for tsp225 data-set

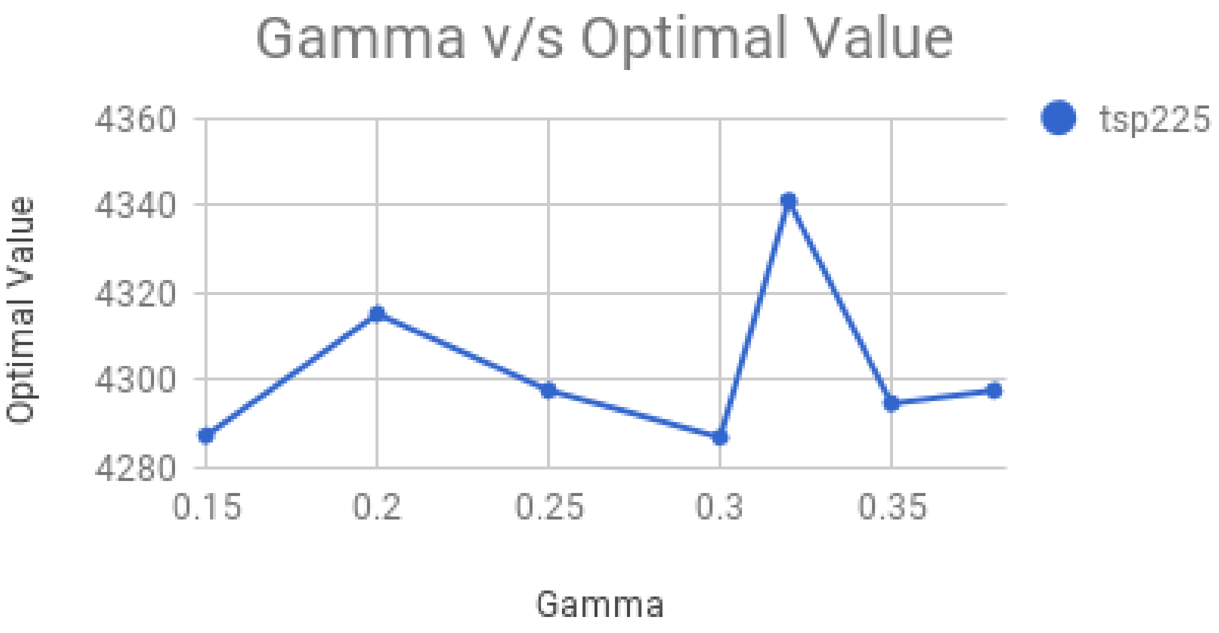

Fig. 4.9: Plot of Gamma vs Optimal value for tsp225 data-set

Further, Table 4.1 summarizes the results of the comparison between the firefly algorithm and the . The best value obtained for both the data-sets are marked in green.

It is evident from these results that firefly algorithm produced better results in comparison to Ant Colony Optimization, although the difference between the best values is not much.

Dataset	Firefly					ACO			
TSP442	Fireflies	Iterations	Gamma	m	Optimal Value	Alpha	Beta	Rho	Optimal Value
	250	1000	0.25	20	57010.1	4	4	0.3	57579
	250	1000	0.2	20	56897.1	4	7	0.4	57616.7
	250	1000	0.35	20	57151.1	4	7	0.42	57295.6
	500	1000	0.35	10	57254.0	4	5	0.3	57646.3
TSP225	Fireflies	Iterations	Gamma	m	Optimal Value	Alpha	Beta	Rho	Optimal Value
	250	500	0.3	15	4317.9	4	7	0.3	4440.9
	100	1000	0.35	20	4287.2	4	8	0.3	4327.3
	250	1000	0.2	20	4286.7	4	7	0.3	4440.9
	100	500	0.2	20	4321.7	2	4	0.3	4306.9

Table 4.1: Comparison of results of Firefly Algorithm and Ant Colony Optimization

4.3 3-Coloring

For the purpose of comparing the results we used an implementation of the greedy algorithm for the problem. The results were compared for various combinations of the number of vertices and the number of edges in the graph. The same were recorded multiple times and the minimum, maximum and the average value obtained are reported. As can be seen from table 4.2 that the implementation of the firefly algorithm performed better than the implementation of the greedy approach. Figure 4.10 to Figure 4.13 pictorially represent the same comparative analysis.

Nodes	Edges	Greedy Algorithm			Firefly Algorithm		
		Min	Max	Average	Min	Max	Average
10	5	0.864	1	**1**	1	1	**1**
	10	0.822	0.856	**0.841**	0.826	1	**0.852**
	15	0.754	0.757	**0.756**	0.762	0.765	**0.764**
	20	0.712	0.718	**0.714**	0.719	0.723	**0.720**
	25	0.692	0.695	**0.694**	0.696	0.699	**0.698**
50	50	0.864	1	**0.919**	0.864	1	**0.924**
	100	0.822	0.856	**0.842**	0.826	0.866	**0.853**
	250	0.754	0.757	**0.756**	0.762	0.765	**0.764**
	500	0.712	0.718	**0.714**	0.719	0.723	**0.720**
	1000	0.692	0.695	**0.694**	0.696	0.699	**0.698**
100	100	0.888	0.945	**0.915**	0.895	0.945	**0.924**
	250	0.802	0.821	**0.810**	0.811	0.825	**0.816**
	500	0.741	0.755	**0.747**	0.748	0.759	**0.752**
	1000	0.710	0.714	**0.712**	0.714	0.716	**0.715**
	2000	0.692	0.693	**0.692**	0.694	0.695	**0.694**
500	500	0.903	0.927	**0.916**	0.907	0.927	**0.917**
	1000	0.831	0.848	**0.836**	0.832	0.848	**0.838**
	2500	0.744	0.750	**0.748**	0.746	0.750	**0.749**
	5000	0.711	0.713	**0.712**	0.712	0.714	**0.713**
	10000	0.692	0.693	**0.693**	0.693	0.694	**0.693**

Table 4.2: Comparison of results of greedy and firefly for 3-coloring problem

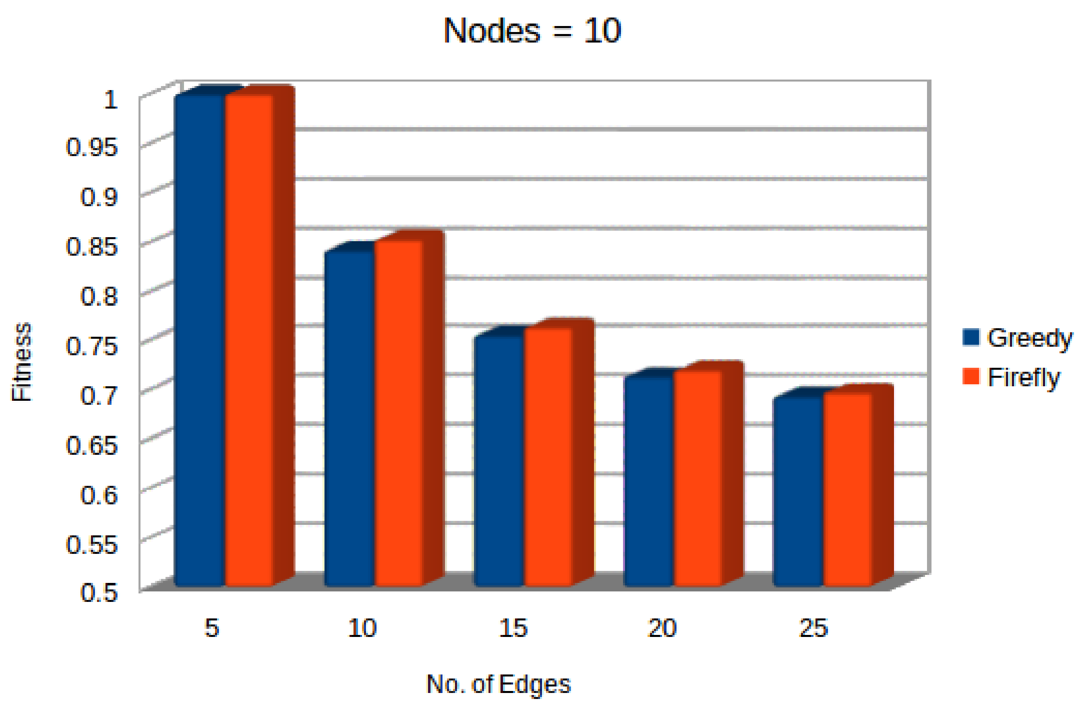

Fig. 4.10: Comparison of greedy and firefly on input graph of 10 nodes

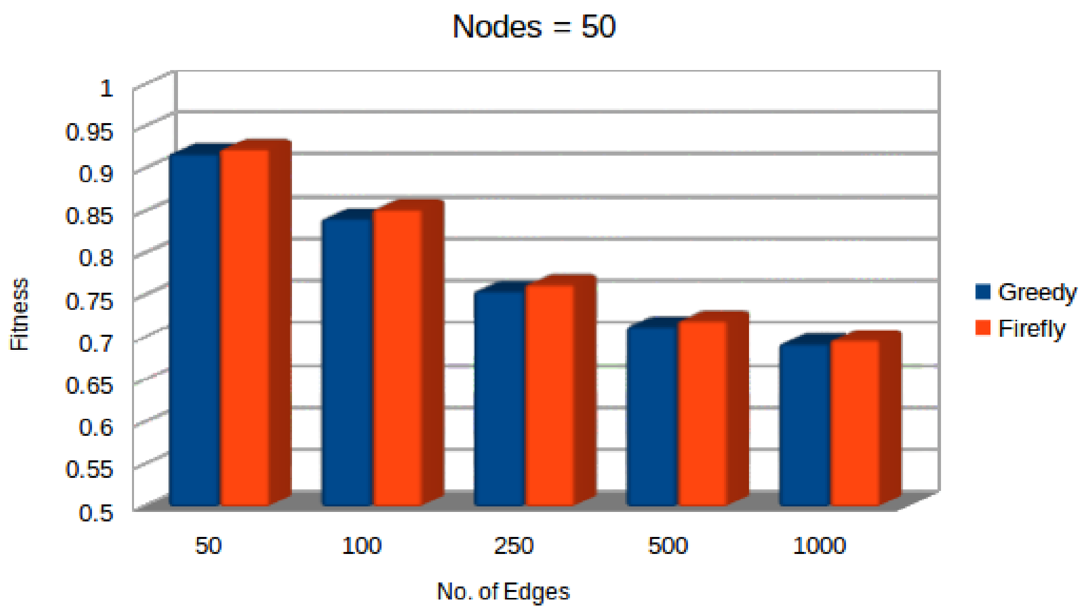

Fig. 4.11: Comparison of greedy and firefly on input graph of 50 nodes

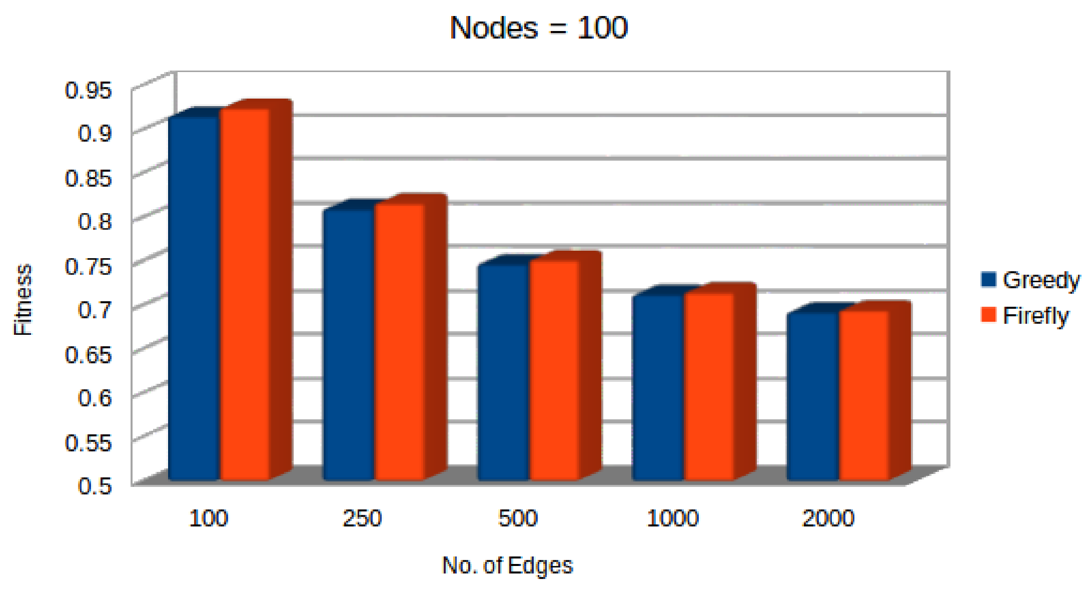

Fig. 4.12: Comparison of greedy and firefly on input graph of 100 nodes

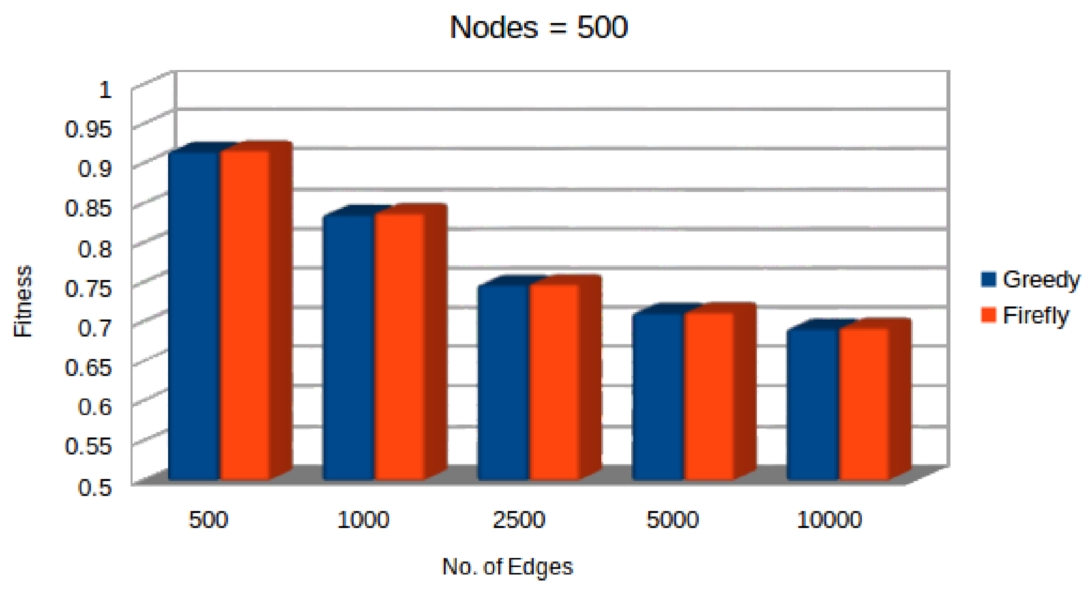

Fig. 4.13: Comparison of greedy and firefly on input graph of 500 nodes

4.4 TDMA Scheduling

For the TDMA scheduling problem, We generated the input graph randomly by specifying the number of vertices and the clustering coefficient which is a measure of the ratio of the the number of edges in the graph to the maximum number of edges that can exist in a graph with given number of vertices. This gives a measure of how dense the graph is. The clustering coefficient becomes the probability of adding an edge between any pair of vertices. For example, if take an 10 as the number of vertices. The maximum number of edges possible is $\binom{10}{2}$ which is 45. suppose we want nearly 10 edges in the graph. Then the clustering index becomes 10/45. Now for each pair of vertices, we generate a uniformly distributed random between 0 and 1. If the generated number turns out to be less than or equal to the clustering index, we add an edge between those pair of vertices.

For the purpose of comparing the results, we use an implementation of the Maximum Independent Set approach, wherein we first try to find a set of vertices of the maximum possible size in which no two vertices share an edge and remove these vertices from the graph. We keep repeating this process until there are no more vertices left in the graph. The number of iterations required to complete the process is also the number of colors required to color the graph as the vertices removed in one particular iteration can be colored with the same color.

The comparison was done for 3 different values of the number of nodes in the original graph and the value of the Clustering Coefficient and multiple results for the same combination of inputs were recorded. The results of the same have been summarized in Table 4.3. It is evident from the table table that firefly algorithm produced better results than the MIS approach.

Input Graph		Conflict Graph		Number of Colors	
Nodes	**Clustering Coefficient**	**Nodes**	**Edges**	**Firefly**	**MIS**
50	0.408	516	6992	**23**	**24**
		492	6410	**22**	**24**
		505	6867	**24**	**28**
	0.816	1002	26282	**41**	**41**
		967	24522	**39**	**40**
		990	25538	**41**	**43**
100	0.202	986	12764	**27**	**31**
		1002	13157	**23**	**26**
		971	12133	**24**	**27**
	0.404	1946	50218	**45**	**48**
		1995	52371	**45**	**47**
		2033	54621	**43**	**43**
250	0.080	2550	34354	**27**	**28**
		2524	34179	**26**	**27**
		2474	32643	**29**	**30**
	0.161	4916	128448	**52**	**55**
		5000	133397	**51**	**51**
		5015	133799	**50**	**51**

Table 4.3: Performance of Firefly vs Maximum Independent Set for graph coloring problem

Chapter 5

Conclusion and Future Work

5.1 Conclusion

As can be seen from the results of various problems in Chapter 4, the firefly algorithm produces better results than it's counterparts. It has produced good results for problems involving both continuous as well as discrete optimization functions. The quick convergence and ease of implementation makes it a good alternative to other bio-inspired meta-heuristics.

5.2 Future Work

The movement functions for the firefly algorithm used in the TDMA scheduling problem can be further optimized to give better results.

Also, the program for graph coloring can be converted into a full fledged package which can take any graph and find the number of colors and the color assignment to the vertices.

References

[1] R. Chiong, *Nature-inspired algorithms for optimisation*, vol. 193. Springer, 2009.

[2] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical computer science*, vol. 4, no. 3, pp. 237–244, 1977.

[3] J. Mao, Z. Wu, and X. Wu, "A tdma scheduling scheme for many-to-one communications in wireless sensor networks," *Computer Communications*, vol. 30, no. 4, pp. 863–872, 2007.

[4] S. C. Ergen and P. Varaiya, "tdma scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2009.

[5] R. Beigel and D. Eppstein, "3-coloring in time o (1.3289 n)," *Journal of Algorithms*, vol. 54, no. 2, pp. 168–204, 2005.

[6] P. Galinier, A. Hertz, and N. Zufferey, "An adaptive memory algorithm for the k-coloring problem," *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 267–279, 2008.

[7] P. Galinier and J.-K. Hao, "Hybrid evolutionary algorithms for graph coloring," *Journal of combinatorial optimization*, vol. 3, no. 4, pp. 379–397, 1999.

[8] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.

[9] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *biosystems*, vol. 43, no. 2, pp. 73–81, 1997.

[10] D. Karaboga and B. Basturk, "Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems," in *International fuzzy systems association world congress*, pp. 789–798, Springer, 2007.

[11] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*, pp. 760–766, Springer, 2011.

[12] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, pp. 169–178, Springer, 2009.

[13] I. Fister, I. Fister Jr, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[14] D. Costa and A. Hertz, "Ants can colour graphs," *Journal of the operational research society*, vol. 48, no. 3, pp. 295–305, 1997.

[15] P. Galinier and A. Hertz, "A survey of local search methods for graph coloring," *Computers & Operations Research*, vol. 33, no. 9, pp. 2547–2562, 2006.

[16] E. Salari and K. Eshghi, "An aco algorithm for the graph coloring problem," *Int. J. Contemp. Math. Sciences*, vol. 3, no. 6, pp. 293–304, 2008.

[17] V. Cutello, G. Nicosia, and M. Pavone, "A hybrid immune algorithm with information gain for the graph coloring problem," in *Genetic and Evolutionary Computation Conference*, pp. 171–182, Springer, 2003.

[18] M. Chams, A. Hertz, and D. De Werra, "Some experiments with simulated annealing for coloring graphs," *European Journal of Operational Research*, vol. 32, no. 2, pp. 260–266, 1987.

[19] S. C. Ergen and P. Varaiya, "Tdma scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[20] S. C. Ergen and P. Varaiya, "Tdma scheduling algorithms for sensor networks," *Berkeley University*, 2005.

[21] TSPLIB, "List of datasets for TSP."

[22] K. Chen and H. Kanoh, "A discrete firefly algorithm based on similarity for graph coloring problems," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017 18th IEEE/ACIS International Conference on*, pp. 65–70, IEEE, 2017.

Curriculum Vitae

Name: Varun Bansal

Date of birth: 30 December, 1996

Education qualifications:

- Bachelor of Technology in CSE from IIT Mandi (In Progress): 8.55 (C.G.P.A till 7th Semester)
- AISSCE (12th): 92.2% : C.B.S.E. Board
- AISSE (10th): 10.0 (C.G.P.A.) : C.B.S.E. Board

Permanent address: A-152, Xu-III, Greater Noida, Uttar Pradesh, Pin:201308