Semester Project

Name: M Rehan Mehdi

Sap ID: 55189

Program: Computer Science

Semester/Section: 4/1

Course: Analysis of Algorithm

Instructor: Mr. Muhammad Usman Sharif

Project Title: Firefly Algorithm

Submission Date: May 11, 2025

Table of Contents:

1. Introduction

The Firefly Algorithm (FA) is a metaheuristic inspired by the flashing behavior of fireflies, developed by Xin-She Yang in 2008. It provides a robust solution for solving NP-Hard, non-convex, and multimodal optimization problems. This project presents the theoretical foundation, implementation in C++, time and space complexity analysis, real-world applications, and its limitations in line with the Seoul Accord standards and the Course Learning Outcomes (CLOs).

2. Methodology

2.1 Algorithm Overview & Pseudocode:

The Firefly Algorithm is based on the attractiveness behavior of fireflies, where brighter fireflies attract others. Brightness is determined by the fitness value of the firefly's position in the search space.

2.2 General Firefly Algorithm Pseudocode:

```
1. Initialize parameters (population size, β0, γ, α, max generations)
2. Generate initial firefly population
3. Evaluate fitness of each firefly
4. for t = 1 to MaxGenerations:
  a. for i = 1 to n fireflies:
    i. for j = 1 to n fireflies:
      - If firefly_j is brighter than firefly_i:
        * Move firefly_i towards firefly_j using attractiveness and randomness
    ii. Evaluate new position
5. Update the global best solution
```

Parameter Definitions:

- **α (alpha):** Randomness scaling factor
- **β_0 (beta):** Base attractiveness
- **γ (gamma):** Light absorption coefficient

- n: Number of fireflies

- MaxGen: Maximum generations

3. Applications: Real-World Scenario & Ethical Considerations:

Real-World Scenario: Exam Scheduling

The Firefly Algorithm is applied to the exam timetabling problem, where exams must be scheduled into a limited number of timeslots without causing conflicts for students. This is a known NP-Hard problem, commonly faced by educational institutions.

3.1 Implement algorithms and solve real-world problems

We applied the Firefly Algorithm to solve an exam scheduling problem, where:

- Students must not have two exams at the same time (minimize conflicts).
- Only a limited number of timeslots is available (resource constraint).

Each firefly represents a possible schedule. The algorithm:

- Initializes random schedules.
- Moves less optimal fireflies toward better ones.
- Evaluates solutions based on the number of exam conflicts.
- The algorithm is tested over multiple runs to measure:
- Conflicts per run
- Time taken per run

3.2 C++ Implementation:

I used C++ to implement the Firefly Algorithm for solving the exam scheduling problem. Each firefly represents a possible exam schedule. The algorithm improves the schedules by moving fireflies toward better ones. It also checks the time taken and memory used during the process.

3.3 Pseudocode for Exam Scheduling Problem

1. Set algorithm parameters (firefly count, β0, γ, α, max generations)

2. Generate initial firefly population (each firefly = exam schedule)

```
3. For each firefly, calculate fitness = -number of student exam conflicts
4. For each generation (t = 1 to MaxGenerations):
  a. For each firefly i:
    i. For each firefly j:
      - If firefly_j has fewer conflicts than firefly_i:
        * Move firefly_i toward firefly_j (adjust exam timeslots)
    ii. Recalculate fitness of firefly_i
5. Return the firefly with the lowest number of conflicts (best schedule)
```

4. Complexity Analysis

4.1 Time Complexity:

1. Initialization: $O(n \times m)$
2. Main loop: $O(n^2 \times m \times G)$
   Where:
3. n = number of fireflies
4. m = number of exams
5. G = generation

4.2 Space Complexity:

Each firefly stores a vector of size m → Total: $O(n \times m)$

- **Worst-case:** $O(n^2 \times m \times G)$ – full nested iterations
- **Best-case:** $\Omega(n \times m)$ – fast convergence
- **Typical:** $\Theta(n^2 \times m \times G)$

This shows how the algorithm behaves in different scenarios, and how firefly interactions affect complexity.

5. Algorithm Design

- The Firefly Algorithm was adapted creatively:
- Instead of using it on standard numeric problems, we used it on a discrete scheduling problem.
- The movement of fireflies is customized to change exam slots intelligently.
- The algorithm handles an ill-defined search space where no clear optimal schedule is known.

- This demonstrates original problem-solving and creative use of a metaheuristic technique.

5.1 Key Challenges:

- Students are enrolled in overlapping courses.
- Timeslots and resources (rooms/invigilators) are limited.
- Manual scheduling is time-consuming and error-prone.

5.2 Firefly Algorithm Impact:

- Quickly generates near-optimal exam schedules.
- Reduces the number of student exam clashes.

5.3 Ethical Considerations:

1. **Fairness:** Minimizing student exam conflicts promotes academic equity and mental well-being.
2. **Efficiency:** Automation saves administrative effort and reduces scheduling mistakes.

6. Limitations

Why the Firefly Algorithm Can Fail in Some Casses:

Premature Convergence:

Fireflies (solutions) can get stuck too early, focusing on a suboptimal solution. This happens when they stop exploring different possibilities too soon.

Scalability Issues:

As the number of fireflies or generations increases, the algorithm becomes slower. For very large problems, it may take too much time to find a solution.

Not Ideal for All Problems:

Firefly Algorithm was made for continuous problems (like optimization in math), but applying it to things like scheduling requires adjustments. If these adjustments are wrong, it can affect how well the algorithm works.

7. CLO & Seoul Accord Mapping:

This *Table 1* outlines how the project meets the Course Learning Outcomes (CLOs) and the Seoul Accord graduate attributes as per the course rubric.

CLO	**Bloom's Level**	**What Was Done in the Project**	**Seoul Accord Attribute(s)**
3.1	Apply	Implemented the Firefly Algorithm in C++ to solve an exam scheduling problem; tested performance	#1 Conflicting Requirements, #6 Stakeholder Diversity
4.1	Analyze	Analyzed time and space complexity of the algorithm in various cases	#2 Depth of Analysis, #4 Unfamiliar Issues
4.2	Analyze	Discussed asymptotic notations (Big-O, Θ, Ω); explained behavior across input sizes	#3 Depth of Knowledge, #8 Interdependence
5.1	Evaluate	Applied the algorithm to a real-world case; discussed fairness, efficiency, and ethical impact	#5 Beyond Standard Practice, #7 Societal Consequences
6.1	Create	Adapted the algorithm for a discrete, ill-defined scheduling problem; fine-tuned solution design	#8 Interdependence, #9 Ill-Defined Requirements

Table 1

8. GitHub Repository Link:

https://github.com/MRehanMehdi/Firefly-Algorithm-Exam-Scheduling.git