TEAM PROJECT 01 – SEARCH ALGORITHMS

CSC14003 – FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE

1. PROJECT OVERVIEW

This project focuses on implementing, analyzing, and comparing swarm intelligence algorithms using only NumPy. Swarm intelligence represents a fascinating branch of artificial intelligence inspired by the collective behavior of decentralized, self-organized systems found in nature. Students will gain hands-on experience with bio-inspired optimization techniques and understand how simple agents following basic rules can solve complex optimization problems.

2. LEARNING OBJECTIVES

By completing this project, students will:

- Understand the theoretical foundations of swarm intelligence algorithms.
- Implement nature-inspired optimization algorithms from scratch using NumPy.
- Develop skills in algorithm visualization and performance analysis.
- Compare swarm intelligence with traditional search algorithms.
- Apply optimization techniques to real-world problem domains.
- Work collaboratively in teams to deliver a comprehensive technical project.

3. PROJECT REQUIREMENTS

Students must implement the following five swarm intelligence algorithms:

Algorithm	Strengths	Weaknesses	Best Applications
Ant Colony Optimization (ACO) [1]	Effective for combinatorial problems and	Computationally intensive and requires fine-tuning	Routing problems, scheduling, and resource allocation

	handles complex discrete spaces well		
Particle Swarm Optimization (PSO) [2]	Good for continuous optimization and simple and easy to implement	Can converge to local optima and is less effective for discrete problems	Hyperparameter tuning, engineering design, financial modeling
Artificial Bee Colony (ABC) [3]	Adaptable to large, dynamic problems and balanced exploration and exploitation	Computationally intensive and requires careful parameter tuning	Telecommunications, large-scale optimization, and high-dimensional spaces
Firefly Algorithm (FA) [4]	Excels in multimodal optimization and has strong global search ability	Sensitive to parameter settings and slower convergence	Image processing, engineering design, and multimodal optimization
Cuckoo Search (CS) [5]	Efficient for solving optimization problems and has strong exploration capabilities	May converge prematurely and performance depends on tuning	Scheduling, feature selection, and engineering applications

Implementation requirements:

- All algorithms must be implemented using **NumPy only** (no scikit-learn, scipy.optimize, or other high-level libraries)

- Code should be modular, well-documented, and follow Python best practices
- Each algorithm should have configurable parameters (population size, iterations, etc.)
- Implementations should handle both continuous and discrete optimization problems. Maybe you can choose suitable problem for each algorithm.

Students are required to create visualizations that demonstrate *convergence ability*, *comparative performance*, *parameter sensitivity analysis*, (advanced) 3D surface plots to show the objective function landscape (for continuous optimization problems). Recommended libraries for visualization: Matplotlib, and Seaborn.

Furthermore, students are **required to compare** swarm intelligence algorithms against **at least three** traditional search algorithms:

- Hill Climbing (steepest ascent).
- Simulated Annealing.
- Genetic Algorithm.
- Breadth-First Search (for discrete problems).
- Depth-First Search (for discrete problems).
- A* Search (for path-finding problems).

The recommended metrics for comparison are:

- Convergence speed.
- Computational complexity (time and space).
- Robustness (performance across multiple runs).
- Scalability (performance with problem size).

For choosing test problems, students should report both on continuous and discrete problems. You can choose at least one for each type of problem as shown in the table below:

Type of problem	Continuous Optimization	Discrete Optimization
	Sphere function	Traveling Salesman Problem (TSP)
	Rastrigin function	Knapsack Problem (KP)
	Rosenbrock function	Graph coloring (GC)
	Ackley function	

The report must fully give the following sections:

- Member information (student ID, full name, ...)
- Work assignment table, which includes detailed information on each task assigned to team members, along with the completion rate of each member compared to the assigned tasks
- Self-evaluation of the completion rate at each level of the project and other requirements.
- Detailed algorithm description for each level. Although a higher level may encompass the previous ones, presenting the information by levels will enhance clarity for the assessment and commentary process. Illustrative images are encouraged.
- Describe the test cases and the results when run on each of those test cases.
- The report needs to be **well-formatted and exported to PDF**. Note that for editors like Jupyter notebook, the team needs to find a way to format it well before exporting it to PDF.
- If there are figures cut off by the page break, etc., points will be deducted.
- References (if any) with APA format.
- Language usage: **Vietnamese**

For submission:

- Your report, source code and test cases must be contributed to the form of a compressed file (**must be .zip**) and named according to the format **<Group_ID>.zip**, for example: *Group_01.zip*.
- Demo videos (recording the running process of your program for each test case) should be uploaded to YouTube or Google Drive, and the public URLs are included in the report.
- If the compressed file is larger than **25MB**, prioritize compressing the report and source code. Test cases may be uploaded to Google Drive and shared via a link.

4. ASSESSMENT

No	Description	Details	Scores
1	Technical Report	1.1 Algorithm descriptions with mathematical formulations 1.2 Implementation details and design decisions 1.3 Experimental methodology and test configurations 1.4 Results analysis with tables and charts 1.5 Comparative analysis and discussion 1.6 Conclusions and insights learned 1.7 Minimum 25 pages	40%
2	Source code	2.1 Well-structured Python implementation of all swarm algorithms 2.2 Implementation of traditional search algorithms for comparison 2.3 README with setup instructions and usage examples. And uploaded it on Github.	40%

3	Demo	Short video demo uploaded on Youtube at least 5 minutes.	20%

5. NOTICES

Please pay attention to the following notices:

- This is a GROUP assignment. Each group has about 3 to 4 members.
- Duration: **about 4 weeks**.
- Any plagiarism, any tricks, or any lie will have a 0 point for the course grade.

6. REFERENCES

[1] Dorigo, M., Birattari, M., & Stutzle, T. (2007). Ant colony optimization. *IEEE computational intelligence magazine*, *1*(4), 28-39.

[2] Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft computing*, *22*(2), 387-408.

[3] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, *39*(3), 459-471.

[4] Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. International journal of swarm intelligence, 1(1), 36-50.

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