****

**INDEPENDENT UNIVERSITY,**

**BANGLADESH**

**Artificial Intelligence**

**Report**

**Team Project**

**Topic: Hill Climbing and Swarm Intelligence Algorithms**

**Context: Solving the Travelling Salesman Problem**

**Faculty: Amin Ahsan Ali, Associate Professor, Dept of CSE, IUB**

**Email:** [aminali@iub.edu.bd](mailto:aminali@iub.edu.bd)

**Prepared by  
Fahim Shahriar Eram (2022523), Syed Niaz Mohtasim (2021607)**

**Date of Submission: 16th August 2023**

**Semester: Summer 2023**

# 

# The Travelling Salesman Problem and its applications

Travelling Salesman Problem (TSP) is a classic optimization problem in computer science and mathematics. It asks for the shortest possible route that visits each city exactly once and returns to the starting city.

TSP has many applications in different fields, such as:

* Planning and logistics, such as vehicle routing, aircraft route arrangement, and job sequencing.
* Manufacturing, such as computer wiring and cutting wallpaper.
* Biology, such as DNA sequencing.
* Artificial intelligence, such as using intelligent algorithms to solve TSP.

# Objectives of this project

The main goal of this project is to solve TSP by using two different algorithms, the Hill climbing algorithm and a swarm intelligence algorithm of our choice. We chose to go with the Bat Algorithm. The Bat Algorithm strikes a balance between exploration (searching the solution space for diverse solutions) and exploitation (refining promising solutions).

# Description of the Algorithms used

**Hill climbing algorithm** is an optimization algorithm that tries to find the best solution for a given problem by moving in the direction of increasing value until it reaches a peak or a local maximum. It is a type of local search algorithm that is often used in artificial intelligence and to solve TSP.

On the other hand, the **Bat algorithm** is a metaheuristic optimization algorithm that mimics the echolocation behavior of Bats when they search for their prey. It is based on the idea that Bats can adjust their pulse emission rate and loudness according to their distance to the target. The Bat algorithm can be applied to solve the TSP by using a discrete representation of the solution, where each Bat represents a possible tour of the cities.

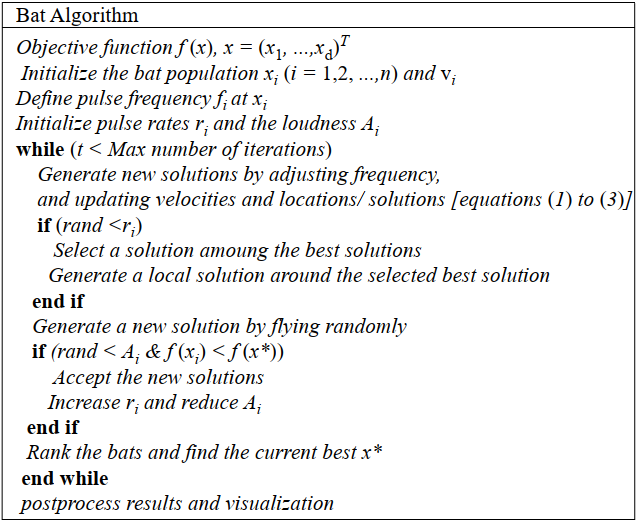


Figure 1 Bat Algorithm Pseudocode [1]

# Experimental Setup

We used Python as the programming language to implement these algorithms and the Visual Studio Code IDE as our code editor. We used libraries such pandas and numpy for data reading and processing. We modelled TSP as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's weight. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph.

For each algorithm, we ran it a certain 100 number of times (100 runs) with the same set of problem instances. We ran both algorithms on 3 different instances of symmetric TSP (distance from city A to city B is same as city B to city A) which we obtained from the TSPLIB library. The 3 file instances are berlin52, att48 and pr76.

The maximum iteration was set to 1000 for both the algorithms. While bat algorithm had another variable, population size. We varied it between 50,100,250 and 500.

The performance metric we used to evaluate the algorithms is the total distance travelled by the tour, we chose the shortest distance from the 1000 iterations and chose it as our best distance and compared it to the optimal distance, which we got from TSPLIB as well. We used these two distances to calculate an error percentage for both algorithms.

Iterations = 1000, runs for each iteration = 100

# Results

Figure 2 Time Bar Chart (att48)

Figure 3 Time Bar Chart (berlin52)

Figure 4 Time Bar Chart (pr76)

The simplicity of hill climbing is highlighted by these charts as it takes less than a minute to run for all three cities. The Bat algorithm's adaptability to different population sizes showcased its versatility and potential for optimization across various problem complexities although increasing population increases its runtime significantly.

Figure 5 Line Chart comparing distances

**Error percentages**

**For att48**

Hill Climbing - 135.06%

Bat Algo (50) - 105.12%

Bat Algo (100) - 103.93%

Bat Algo (250) - 69.54%

Bat Algo (500) – 68.83%

**For berlin52**

Hill Climbing - 135.56%

Bat Algo (50) - 133.52%

Bat Algo (100) - 132.42%

Bat Algo (250) - 111.98%

Bat Algo (500) – 125.07%

**For pr76**

Hill Climbing - 231.46%

Bat Algo (50) - 165.58%

Bat Algo (100) - 160.05%

Bat Algo (250) - 158.39%

Bat Algo (500) - 150.32%

Figure 6 Error vs Population (Bat Algorithm)

Even though hill climbing is a simple and easy to implement with a low runtime, bat algorithm is more versatile due to having the ability of various population size. And as the graphs show that on increasing the population size, error percentage goes down. Even at the low population size of 50, it is performing better than hill climbing. But bat algorithm has more random elements involved so it may not always bring the best results on population increase. There's a point beyond which increasing the population size may not lead to significant improvements in solution quality.

# Conclusion

In conclusion, this project explored two distinct approaches to solving the Traveling Salesman Problem, shedding light on their strengths and trade-offs. Hill Climbing, a simple yet efficient local search algorithm, exhibited low runtimes and ease of implementation. However, it struggled to achieve competitive results, with error percentages exceeding those of the Bat Algorithm.

The Bat Algorithm, on the other hand, displayed remarkable versatility and adaptability. By varying the population size, the algorithm demonstrated the potential for optimization across diverse problem complexities. As population size increased, error percentages decreased, highlighting its effectiveness in finding near-optimal solutions.

While Hill Climbing is an accessible choice for straightforward cases, the Bat Algorithm emerges as a robust alternative for tackling more complex instances of TSP. It's important to note that the Bat Algorithm introduces random elements, and beyond a certain population size, improvements in solution quality may diminish.

In summary, this project provides valuable insights into the performance of Hill Climbing and the Bat Algorithm in solving TSP. The choice of algorithm should depend on the problem's complexity and the balance between runtime constraints and solution quality requirements.

# Contributions

In the completion of this project, our team, consisting of Syed Niaz Mohtasim and Fahim Shahriar Eram collaborated closely to successfully implement and evaluate two prominent optimization algorithms, namely Hill Climbing and the Bat Algorithm, for solving the Traveling Salesman Problem (TSP). The project was divided into distinct roles, where each member contributed their expertise to different facets of the project but also were aware at all times of what each member were doing and both have a good understanding of what the other did.

Fahim worked on the algorithm implementations and coding, wrote the section of report on TSP and its applications.

Syed worked on graph generation and the rest of the report.

Update – Syed worked on the modified code for bat algorithm and the modified report.