### Traveling Salesman Problem Solver Project Overview

Introduction:

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and operations research. It involves finding the shortest possible route that visits each city exactly once and returns to the original city. Despite its seemingly simple description, the problem is NP-hard, meaning that there is no known polynomial-time algorithm that solves it optimally.

This project aims to implement two metaheuristic algorithms, namely Particle Swarm Optimization (PSO) and Simulated Annealing (SA), to tackle the TSP. These algorithms provide approximate solutions to the TSP that are often very close to the optimal solution for small to moderate-sized instances.

Main Objectives:

1. \*\*Algorithm Implementation\*\*: Implement PSO and SA algorithms enhanced with a nearest neighbor initialization approach. These algorithms iteratively improve candidate solutions to converge to a near-optimal solution for the TSP.

2. \*\*Solution Visualization\*\*: Provide a graphical interface to visualize the solutions obtained from the PSO and SA algorithms. This allows users to intuitively understand the routes chosen by the algorithms.

3. \*\*User Interaction\*\*: Develop a user-friendly interface using Tkinter, a Python GUI toolkit, to enable users to interact with the solver easily. Users can select TSP instances stored in CSV files and observe the solutions generated by the algorithms.

Workflow Overview

1. \*\*Data Preparation\*\*: The project starts by preparing the TSP instances. Cities' coordinates are stored in CSV files, which are then read by the solver to create a representation of the problem.

2. \*\*Algorithm Execution\*\*: PSO and SA algorithms are executed on the given TSP instances. These algorithms iteratively explore the solution space, gradually improving the candidate solutions until convergence.

3. \*\*Solution Visualization\*\*: The solutions obtained from both algorithms are visualized using matplotlib, a Python plotting library. Users can observe the routes chosen by the algorithms and compare their performance.

4. \*\*User Interaction\*\*: A graphical interface is provided using Tkinter, allowing users to browse and select TSP instances stored in CSV files. After selecting a file, users can trigger the execution of the algorithms and visualize the solutions.

Expected Results

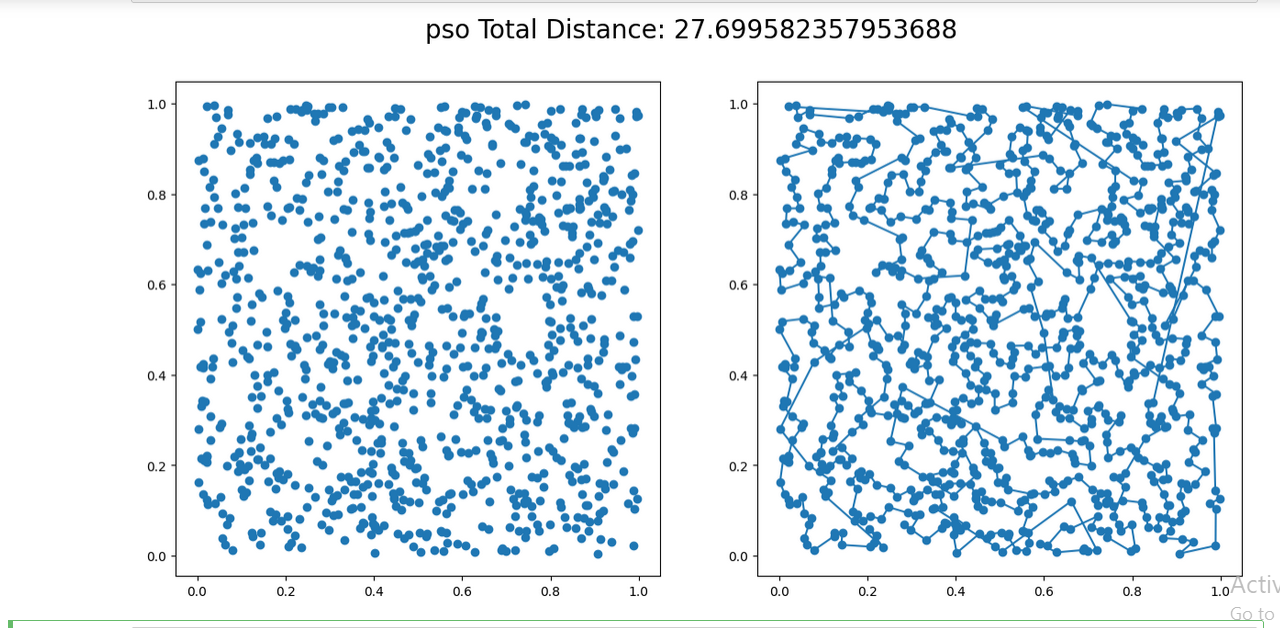
- The PSO and SA algorithms should provide near-optimal solutions to the TSP for small to moderate-sized instances.

- The graphical interface should provide an intuitive way for users to interact with the solver and observe the solutions generated by the algorithms.

- Users should gain insights into the effectiveness of PSO and SA algorithms for solving the TSP and understand their performance characteristics.

# Comparing the distance between particle swarm optimization and simulated\_annealing :

Pso total\_distance:



simulated\_annealing Total Distance:

