A logo with a symbol and text

Description automatically generated with medium confidence A logo with blue text

Description automatically generated

Helwan University | Faculty of Computers & Artificial Intelligence AI Dept. Module: Evolutionary Algorithms [AI420] SPRING 2024

* Project Idea: Particle Swarm Optimization (PSO) and Simulated Annealing for the Traveling Salesperson Problem (TSP)
* Team Discussion Time: (15:20 PM)
* Team ID: 19

|  |  |
| --- | --- |
| **Name** | **ID** |
| **محمد احمد حنفي حسين** | **20210729** |
| **عبدالله حمدي احمد مصطفى** | **20210542** |
| **عمر عادل احمد عبداللطيف** | **20210607** |
| **محمد هيثم عبدالمنعم السيد** | **20210846** |
| **تسنيم احمد طه محمد** | **20210237** |
| **ياسمين السيد سعد احمد** | **20211045** |

**1-Traveling Salesperson Problem:**

The Traveling Salesman Problem (TSP) stands as one of the most famous and studied conundrums in the realm of computer science and optimization. At its core, the TSP seeks the most efficient route fora salesman to visit a set of cities exactly once and return to the starting point, with the objective of minimizing the total distance traveled. Despite its seemingly simple premise, the TSP is a quintessential example of an NP-hard problem, meaning its solution's complexity grows exponentially with the number of cities. This challenge has sparked intense interest across various disciplines, from mathematics to logistics and beyond. The quest to develop efficient algorithms to solve the TSP has led to innovations in optimization techniques, ranging from exact algorithms like branch and bound to heuristic approaches such as genetic algorithms and simulated annealing. Beyond its theoretical intrigue, the TSP finds practical applications in diverse fields, including transportation planning, circuit design, and even DNA sequencing, making it a cornerstone problem in the intersection of mathematics and real-world problem-solving.

TSP is an **Optimization problem**, at its core, the TSP seeks to find the Shortest possible route that visits a given set of cites and returns to the original city, with the constraint that each city must be visited exactly once. The optimization aspect comes from the desire to minimize the total distance traveled.

In its basic form, TSP is a **constrained optimization** problem because it comes with a set of constraints: each city must be visited exactly once, and the salesman must return to the starting city. These constraints narrow down the space of possible solutions.

**2- Main functionalities:**

**a- PSO algorithm :**

\* This code iteratively improves the solution by updating the positions of particles based on their personal best and global best solutions, aiming to find the shortest route that visits each city exactly once and returns to the starting point\*

1. **Distance method**: a method which calculates the distance between two cities.
2. **Nearest neighbor initial route method**: a method which finds the nearest city to a specific city. We use this method to help us set some smart routes for the particles instead of a completely random one.
3. **total distance method:** a method which calculates the total distance for a route in particles which works as our **fitness function.**

5- **Particle Class**: Represents a particle in the PSO algorithm. Each particle has a route (sequence of cities), velocity, cost (total distance of the route), and personal best (pbest) route and cost.

6- **initialize particles method:** a method which generates random particles to be used for our PSO algorithm (which we initialized to be **20 particles** )

1. **Update velocity method**: a method which updates the velocity for each particle after each iteration using the PSO equation which updates it, which is   
   *vij*​(*t*+1)=*w*⋅*vij*​(*t*)+*c*1​⋅*r*1​⋅(*pij*​−*xij*​(*t*))+*c*2​⋅*r*2​⋅(*pgj*​−*xij*​(*t*))

5-**Update position method**: a method which updates the position for each particle after each iteration which update the position after each iteration using the PSO equation which updates it, which is *xij*​(*t*+1)=*xij*​(*t*)+*vij*​(*t*+1)

1. **crowding penalty method**: a method that calculates a penalty factor for each pair of particles in the population based on their proximity to each other in the solution space. This penalty factor is then used to adjust the fitness of the particles ( which works as **preserving diversity)**

7- **PSO tsp method**: a method which applies the PSO algorithm to our particles (updating particles’ velocities and positions after each iteration) and setting up number for iterations which we will update the Pbest and Gbest for our routes in, which we initialized to be 100 iterations.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**b- simulated annealing:**

**\*** These functionalities together form a solution to the Traveling Salesman Problem (TSP) using the simulated annealing optimization technique\*

1- **Loading Dataset**: The code loads a dataset of cities with their coordinates from a CSV file.

2- **Distance Calculation**: It defines a function to calculate the Euclidean distance between two cities using their coordinates.

3- **Nearest Neighbor Initial Route**: This function implements the nearest neighbor algorithm to generate an initial route for the traveling salesman problem. It starts from a random city and iteratively selects the nearest unvisited city until all cities are visited, returning to the starting city.

4- **Total Distance Calculation**: This function calculates the total distance of a given route, including returning to the starting city.

5- **Simulated Annealing Algorithm**: The code implements the simulated annealing algorithm to find an optimized route for the traveling salesman problem. It iteratively swaps two cities randomly and accepts the new route based on the Metropolis criterion. The algorithm aims to minimize the total distance traveled.

1. Initialize the temperature parameter 𝑇T and the initial solution 𝑥current current.

2. Repeat until stopping criteria are met: a. Generate a new solution 𝑥new new by perturbing the current solution. b. Calculate the change in energy: Δ𝐸=𝐸(𝑥new)−𝐸(𝑥current)ΔE=E(xnew)−E(xcurrent). c. If Δ𝐸<0ΔE<0, accept the new solution: 𝑥current=𝑥newxcurrent=xnew. d. If Δ𝐸≥0ΔE≥0, accept the new solution with probability 𝑃(Δ𝐸,𝑇)=𝑒−Δ𝐸𝑇P(ΔE,T)=e−TΔE. e. Update the temperature schedule.

3. Return the best solution found.

6- **Plotting the Best Route**: After finding the best route using simulated annealing, the code plots the path of the best route on a 2D graph.

7- **Setting Initial Parameters**: The code sets initial parameters such as initial temperature, cooling rate, and the number of iterations for the simulated annealing algorithm.

8- **Printing Results**: Finally, the code prints the best route found, its corresponding distance, and the length of the route.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3-**Similar applications in the market**:**

* Concorde TSP Solver: Concorde is one of the most renowned TSP solvers. It's a high-performance solver developed in C that can solve very large instances of the TSP optimally.
* LKH (Lin-Kernighan Heuristic): LKH is a heuristic algorithm for solving the TSP. It's implemented in C and offers both sequential and parallel versions.
* TSPLIB: TSPLIB is a library of instances for the TSP, including both benchmark instances and real-world problem instances. It's widely used for testing and benchmarking TSP algorithms.
* TSP Solver and Generator: This is a comprehensive software package for solving and generating instances of the TSP. It includes various algorithms such as brute-force, nearest neighbor, simulated annealing, genetic algorithms, etc.
* OR-Tools: OR-Tools is an open-source library developed by Google for solving combinatorial optimization problems. It includes implementations of various algorithms for the TSP, including integer linear programming (ILP) formulations and metaheuristic approaches.

**4-** A literature review of Academic publications (papers) relevant to the idea( particle **Swarm Optimization (PSO) and Simulated Annealing for TSP problem)**

**1-**

**Simple summary:** Traveling salesman problem (TSP) is a well-established NP-complete problem and many evolutionary techniques like particle swarm optimization (PSO) are used to optimize existing solutions for that. PSO is a method inspired by the social behavior of birds. In PSO, each member will change its position in the search space, according to personal or social experience of the whole society. In this paper, we combine the principles of PSO and crossover operator of genetic algorithm to propose a heuristic algorithm for solving the TSP more efficiently. Finally, some experimental results on our algorithm are applied in some instances in TSPLIB to demonstrate the effectiveness of our methods which also show that our algorithm can achieve better results than other approaches.

[**https://www.tandfonline.com/doi/full/10.1080/23311835.2015.1048581**](https://www.tandfonline.com/doi/full/10.1080/23311835.2015.1048581)

**2-**

**Simple summary:**

The Probabilistic Traveling Salesman Problem (PTSP) is a variation of the well known Traveling Salesman

Problem (TSP). This problem arises when the information about customers demand is not available at the moment of the

tour generation and/or the tour re-calculating cost is too elevated. In this article, a Hybrid Algorithm combining Particle

Swarm Optimization (PSO) and Simulated Annealing (SA) is proposed, in order to solve the PTSP. The PSO heuristic

offers a simple structured algorithm which supplies a high level of exploration and fast convergence, compared with other

evolutionary algorithms. The SA algorithm is used to improve the particle diversity and to avoid the algorithm being

trapped into local optimum. Two well-known benchmarks of the literature are used and the proposed PSO-SA algorithm

obtains acceptable results. In fact, the hybrid algorithm improves the performance of simple PSO algorithm for all instances

[**https://sic.ici.ro/wp-content/uploads/2012/03/SIC\_2012-1-Art6.pdf**](https://sic.ici.ro/wp-content/uploads/2012/03/SIC_2012-1-Art6.pdf)

**3- Simple summary:**

To solve travelling salesman problems (TSPs), most existing evolutionary algorithms search for optimal solutions from zero initial information without taking advantage of the historical information of solving similar problems. This paper studies a transfer learning-based particle swarm optimization (PSO) algorithm, where the optimal information of historical problems is used to guide the swarm to find optimal paths quickly. To begin with, all cities in the new and historical TSP problems are clustered into multiple city subsets, respectively, and a city topology matching strategy based on geometric similarity is proposed to match each new city subset to a historical city subset. Then, on the basis of the above-matched results, a hierarchical generation strategy of the feasible path (HGT) is proposed to initialize the swarm to improve the performance of PSO. Moreover, a problem-specific update strategy, i.e. the particle update strategy with adaptive crossover and clustering-guided mutation, is introduced to enhance the search capability of the proposed algorithm. Finally, the proposed algorithm is applied to 20 typical TSP problems and compared with 12 state-of-the-art algorithms. Experimental results show that the transfer learning mechanism can accelerate the search efficiency of PSO and make the proposed algorithm achieve better optimal paths.

[**https://academic.oup.com/jcde/article/9/3/933/6590609**](https://academic.oup.com/jcde/article/9/3/933/6590609)

**4-**

**Simple summary:** Although metaheuristic algorithms, such as PSO, have excellent performance than heuristic algorithms, they fall in premature convergence when used for complex problems in daily life. In other words, this concept causes particles, which show the role of feasible solutions to the problem, to be stuck in local optimal points, and spend a long time without improvement. Therefore, the quality of the obtained answers cannot grow continuously and achieve quality solutions. For this reason and to overcome premature convergence, several modifications are presented to improve the PSO algorithm in the literature [41–45]. In the PSO algorithm, *pbest* and *gbest* are considered, while in the proposed algorithm, called MPSO, the best current answer is also considered as *gcbest*. The method is that the solutions *gbest* and *gcbest* have two coefficients of *a1* and *a2*, respectively, in which the sum of these two variables is equal to 100%. The values of these two coefficients change over time as the algorithm runs, as described in the following. Finally, three local search algorithms are considered for intensification to further improve the answers if *gbest* or *pbest* is updated. The steps of the proposed algorithm are described as follows.

[**https://www.hindawi.com/journals/complexity/2021/6668345/**](https://www.hindawi.com/journals/complexity/2021/6668345/)

**5-**

**Simple summary:**

Nowadays, the systems that are inspired by biological structures have gained importance and attracted the attention of researchers.

The Multiple Travelling Salesman Problem (MTSP) is an extended version of the TSP. The aim in the MTSP is to find the tours for m

salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting nodes is

minimized. The Particle Swarm Optimization (PSO) algorithm which is a meta-heuristic algorithm based on the social behaviour of birds.

In this article, 2 algorithms based on PSO, called APSO and HAPSO, were proposed to solve the MTSP. The APSO algorithm is based on

the PSO and 2-opt algorithms, the path-relink and swap operators. While the HAPSO algorithm is based on the GRASP, PSO and 2-opt

algorithms, the path-relink and swap operators. In the experiments, 5 TSP instances are used and the algorithms are compared with the GA

and ACO algorithms. According to the results, the HAPSO algorithm has the better performance than the other algorithms on the most

instances. Moreover the HAPSO algorithm produces more stable results than the APSO algorithm and the performance of the HAPSO

algorithm is better in all the MTSP instances. Therefore, the HAPSO algorithm is more robust than the APSO algorithm.

**https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.ijisae.org/IJISAE/article/download/1012/583&ved=2ahUKEwiRxbWKiYGGAxWd\_7sIHZHTBV84ChAWegQIBRAB&usg=AOvVaw0JBVg0186RuuhC0I2afj0h**

**6-**

**Simple summary:** The Travelling Salesman Problem, briefly TSP is a description of a large class of prob-

lems known as combinatorial optimization problems. The concept of the problem is as

follows: A salesman has a number of cities to visit with a distance or time between two

cities, he wants to take the shortest possible route from all these cities so that he does

not pass from the same city twice and eventually return to where he left off [1]. TSP

is usually simple to explain but very hard to solve, if the number of cities is small, the

answer can be easily found by looking at all possible routes and choosing the shorter

route, but with the increasing the number of cities, this method will become inefficient

and the complexity of solving the problem also increases [2].

Computational techniques stimulated by natural phenomenon were of great interest

in the recent years. The natural phenomenon of insects or large animals is studied to

develop different computing techniques in last few decades [2]. Various combinatorial

optimization problems such as Traveling Salesman Problem (TSP), Job-shop Scheduling

Problem ( JSP), and Vehicle Routing Problem (VRP) were also approached by various

modern heuristic methods, like ACO, IPSO and SFLA [3].

In this paper we give an overview of some meta-heuristic (ACO, PSO, IPSO and SFLA)

algorithms to get the shortest route for the TSP. These algorithms are introduced in

Section 2. Section 3 gives the experimental results of ACO, PSO, modified PSO, SFLA

and modified SFLA for TSP under the MATLAB tool. Consequently, in Section 4, we

conclude the paper with a summarization of results by emphasizing on the importance

of this study.

**https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://knepublishing.com/index.php/KnE-Social/article/view/1394/3196&ved=2ahUKEwiOsoPzioGGAxWK87sIHbWLCEs4ChAWegQIChAB&usg=AOvVaw2BwbAR3XTr4jo6NPlLCMjc**

**5-** the Dataset employed:

We'll assume the salesperson lives in a boring, flat, 2D Cartesian plane, and that each city can be described simply as existing at a single (x, y) location . Each datafile describes some number of cities, one city per line in the file, where each city has a index (first city is index-0) and a location.

We used the large dataset.

Link: https://www.kaggle.com/datasets/mexwell/traveling-salesman-problem

**6-**Details of the algorithms:

**PSO:**

1. Initialization: PSO starts by initializing a population of random solutions. These solutions are called particles.
2. Evaluation: Each particle's fitness is evaluated using an objective function, which provides a measure of how good the particle's current position is, in our case it is the distance fitness function which we aim to minimize.
3. Update Velocity using *vid*​(*t*+1)=*w*⋅*vid*​(*t*)+*c*1​⋅*r*1​⋅(*pid*​(*t*)−*xid*​(*t*))+*c*2​⋅*r*2​⋅(*pgd*​(*t*)−*xid*​(*t*))

And Update Position using *xid*​(*t*+1)=*xid*​(*t*)+*vid*​(*t*+1)

In these equations:

*-* ***Vid(t)*** *is the velocity of the* ***𝑖-th*** *particle in the* ***𝑑-th*** *dimension at iteration* ***𝑡****.*

*-* ***Xid******(t)*** *is the position of the* ***𝑖-th*** *particle in the* ***𝑑-th*** *dimension at iteration* ***𝑡****.*

*-* ***W*** *is the inertia weight, which controls the impact of the previous velocity on the current velocity.*

*-* ***C1*** *​ and* ***C2*** *are acceleration coefficients representing the cognitive and social components of the update process, respectively.*

***- Pid​ (t)*** *is the best-known position of the* ***𝑖-th*** *particle in the* ***𝑑-th*** *dimension at iteration* ***𝑡****.*

*-* ***Pgd ​(t)*** *is the best-known position of any particle in the swarm in the* ***𝑑-th*** *dimension at iteration* ***𝑡****.*

*-* ***r1 ​*** *and* ***r2*** *​ are random values sampled from a uniform distribution in the range [ 0, 1 ] [0,1]. These values introduce stochasticity to the algorithm.*

Each particle adjusts its velocity and position based on its own best-known position and the best-known position in the swarm. This is where the "swarm intelligence" aspect comes into play – particles are influenced by both their own experience and the experiences of their neighbors.

1. Iterate: Steps 2 and 3 are repeated for a certain number of iterations or until a termination criterion is met, such as finding a satisfactory solution.
2. Termination: The algorithm terminates when a stopping criterion is met, such as reaching a maximum number of iterations, finding a solution good enough for the problem, or running out of computational resources.

**Simulated annealing:**

1 Initialization:

Initialize the temperature parameter 𝑇T and the initial solution 𝑥current current.

2 Iterative Improvement:

Generate a new solution 𝑥new new by perturbing the current solution.

3 Acceptance Criteria:

Calculate the change in energy: Δ𝐸=𝐸(𝑥new)−𝐸(𝑥current)ΔE=E(xnew)−E(xcurrent).

If Δ𝐸<0ΔE<0, accept the new solution: 𝑥current=𝑥newxcurrent=xnew.

If Δ𝐸≥0ΔE≥0, accept the new solution with probability 𝑃(Δ𝐸,𝑇)=𝑒−Δ𝐸𝑇P(ΔE,T)=e−TΔE.

4 Cooling Schedule:

Update the temperature parameter 𝑇T according to the cooling schedule.

5 Termination:

The algorithm terminates when a stopping criterion is met, such as reaching a maximum number of iterations or when the temperature drops below a certain threshold.

**7-**Development platform:

-tools: jupyter , VScode,..

- programming languages: python.

- libraries: numpy , random , math , matplotlib .

**8-**compare two algorithm:

A diagram of a route

Description automatically generated

PSO:

Best Distance : 27.85821677051303

A diagram of a route

Description automatically generatedSimulated Annealing :

Best Distance: 28.630640665259335