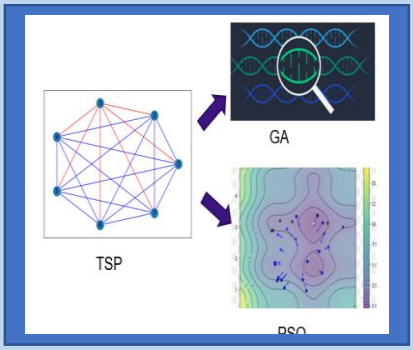
An-Najah University Journal for Research – A
Natural Sciences

A Comparison of Heuristic Algorithms for Solving the Traveling Salesman Problem

Younes Khdeir[1] & Ahmed Awad[1,*]

Accepted Manuscript, In press

Abstract: The Traveling Salesman Problem (TSP) is a challenging computational problem in combinatorial optimization that aims to visit all cities exactly once and return to the first city. Despite that numerous theoretical solutions have been proposed in the literature, finding the exact optimal solution remains computationally infeasible due to the NP-hard nature of the problem. To address this, many heuristic and optimization approaches have been developed to generate probabilistic results that are often approximations. This paper presents a comparison between four popular algorithms: steepest ascent hill climbing, simulated annealing, genetic algorithm with partially matched crossover, and Particle Swarm Optimization (PSO). The study examines how these algorithms can solve the TSP and avoid local minimum values while achieving a balance between research exploration and exploitation for an optimal solution. For a relatively large number of cities, the simulated annealing algorithm and genetic algorithm produce promising results whilst the genetic algorithm takes longer time to execute due to the iterative application of its variation operators.



Keywords: Traveling Salesman Problem (TSP), Simulated Annealing (SA), Genetic algorithm (GA), Particle Swarm Optimization (PSO)

Introduction

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem that belongs to the class of Non-Polynomial (NP) NP-hard problems. The problem involves determining the shortest possible route that visits each city exactly once and returns to the origin city, given a list of cities and the distances between each pair of cities. Due to its theoretical significance in computer science and operations research, TSP has been the focus of extensive research in optimization algorithms, graph theory, and computational complexity [1]. The TSP is widely recognized as one of the most extensively researched optimization problems, serving as a benchmark for evaluating many optimization methods. Despite its computational complexity, TSP has been widely used in various fields in computer science, specifically in artificial intelligence, to find the most efficient path for transferring data between different nodes. The solutions to the TSP problem are not only applicable to the problem per se but also can be translated into various combinatorial optimization problems [2]. Therefore, algorithms that can efficiently solve TSP are highly desirable. Such algorithms are critical for achieving optimal routes while minimizing the time and cost of transportation, network optimization, and hardware optimization [3–5]. The TSP has been a topic of research for several decades, with numerous theoretical solutions proposed. The brute-force approach of testing every possible solution is straightforward, but it can be computationally intractable and time-consuming, especially for large problem sizes where exact solutions are often unattainable due to the limitations of computer resources. Although no exact algorithm is known to solve the TSP in polynomial time, heuristics and optimization-based methods are often employed to produce satisfactory, albeit sub-optimal, solutions. As the worst-case run time of any TSP algorithm can grow in a factorial time with the number of cities, there is a constant need for novel and efficient algorithms to solve the problem [3]. The local search Hill Climb algorithm is employed to find the best solution by moving uphill toward the peak of the problem. The algorithm evaluates neighboring states in proximity to the current state until reaching a local optima [6]. Simulated Annealing (SA) algorithm has been utilized as well by incorporating temperature and cooling ratio parameters to vary the probability of movement between points in the search space, allowing exploration of a larger area of the search space. Similarly, the Genetic Algorithm (GA) is utilized to diversify the search space by creating new solutions through crossover and mutation. The key objective is to strike a balance between exploration and exploitation to obtain the optimal solution in the least possible time and cost. Additionally, the Particle Swarm Optimization (PSO) algorithm, inspired by social behavior, provides an advantage of rapid convergence to a nearly optimal solution through an efficient population-based approach that explores the search space [7].

The paper focuses on solving the TSP with different algorithms. In Addition, it evaluates their effectiveness in finding the shortest path for N cities of various sizes. Furthermore, we introduce a modified version of the Simulated Annealing (SA) algorithm that enhances its speed and improves its results. To represent the distances between cities, a fully connected one-way graph data structure is used. The number of iterations for An - Najah Univ. J. Res. (N. Sc.) Vol. 00 (0), 2023 46 each algorithm is adjusted based on the search space size. Finally, a

1 Department of Information Technology, Faculty of Engineering and Information Technology, An-Najah National University, Nablus, Palestine.
younes.khdeir@gmail.com
*Corresponding author: ahmedawad@najah.edu

comparison is conducted between the evaluated algorithms. Our contributions are summarized as follows: • We conduct a comparison in terms of performance of four algorithms to solve the TSP including: Steep Hill Climb, Simulated Annealing (SA), Genetic Algorithm, and Particle Swarm Optimization (PSO). • We propose an enhanced version of the SA algorithm to solve the TSP and validate its performance. • Both the cost and the runtime for each algorithm is evaluated on a set of benchmarks with varying sizes. The rest of the paper is structured as follows: In Section II, prior research on this topic is discussed. Section III provides a summary of the formulation and representation of the TSP. Section IV proposes a methodology that utilizes four algorithms to solve the TSP. Section V presents the experimental results and analysis. Finally, conclusions drawn from the findings and potential future work are presented in Section VI.

Revious Work

Various approaches have been explored in the field of solving the TSP using optimization algorithms, as observed by previous studies. For instance, the researchers utilized the Steep Hill Climb algorithm to find optimal solutions for TSP problems, but the algorithm was found to be less effective for larger benchmarks [8]. A modified Simulated Annealing (SA) algorithm for TSP has been proposed to outperform quantum annealing algorithms using Leap hybrid solver from D-Wave Systems [9]. Other self-adaptive heuristics-based adjustment strategies optimize the routing of autonomous transportation systems in the context of dynamic and stochastic orienteering problem formulated as TSP. The proposed strategies outperform a static meta-heuristic algorithm in terms of solution quality [10]. An application of the asymmetric TSP to model a small restaurant's cooking process has been proposed. In this context, enhanced versions of the migrating bird's optimization algorithm are found to outperform both the original Migrating Bird's Optimization (MBO) and the simulated annealing algorithm [11]. Other works have employed the GA to solve TSP problems and noted that the crossover and mutation techniques employed in GA helped to explore the search space and diversify the population, making it effective for even larger-sized problems [4]. A new approach to solve the TSP by incorporating sub-tour division inspired by genetic algorithms. The simulation results on the traveling distance between cities in India show that the new approach provides a more accurate and robust solution than alternative methods [5]. The work published in [12] investigates the application of Particle Swarm Optimization (PSO) to the structural optimization design of composite materials for ships. The study uses experimental and comparative methods to analyze the optimization of ship composite material structure and shows promising results with high calculation accuracy. In this paper, we present a comprehensive overview of the latest findings and advancements in the field of TSP optimization algorithms, which have been developed by researchers worldwide. We summarize and compare the results of these algorithms and provide insights into their strengths and limitations. Furthermore, we propose an enhanced SA algorithm to solve the TSP. All algorithms are evaluated in terms of their cost and the runtime.

Problem Formulation

TSP is a mathematical problem whose goal is to find the shortest path between a set of points and locations that must be visited once and back to the starting point. The topology of the cities is represented as a weighted graph G where each vertex represents a city and a weight represents the distance between two cities in the topology. The key objective is to find the optimal tour in terms of time and cost. To solve the TSP, we must first determine the following:

Solution Representation:

The traveling salesman problem is a permutation problem in which the goal is to find the shortest path between N different cities, which is referred to as the tour. Thus, we represent each tour as a permutation of integers, where each integer denotes a city number in the graph. For example, consider the graph depicted in Fig 1 which consists of 7 cities. An example of a solution representation is [6, 7, 4, 5, 1, 2, 3]. This representation indicates a tour starts with city 6 and then returns to this city.

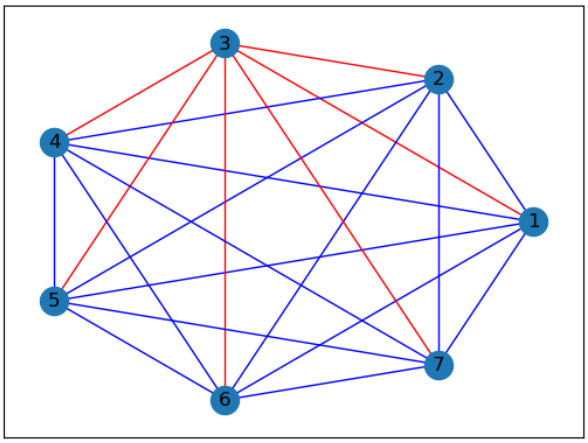


Figure (1): a graph with 7 nodes (cities) fully connected.

Evaluation function

We strive to find solutions that minimize total cost while also shortening the time it takes to find a solution. A solution P is evaluated in terms of the path length it takes from the starting city to the end city within the path, denoted by eval(P), which is formulated in eq.(1), wherein (xi , yi) denotes a pair of cities that belongs to P, and dist(xi , yi) represents the distance between city xi and city yi in the topology.

$$\boldsymbol{eval(P)} = \sum_{\forall (x,y) \in P} \boldsymbol{dist}(x_i, y_i) \quad (1)$$

Thus, The TSP is formulated as an optimization problem given in eq.(2), where S represents the search space, the set of all permutations to solve the problem. Notice that the search space size is (N − 1)!/2, given that the graph is undirected.

$$minimize_{P \in S} eval(P) \qquad (2)$$

$$\& \backslash \boldsymbol{underset}\{\boldsymbol{P} \in \boldsymbol{S}\}\{"\{\boldsymbol{minimize}\}\} \& \& \exists \mathbf{r}\{\boldsymbol{eval}\}(\boldsymbol{P}) \backslash \backslash$$

Neighborhood

We define a neighbor to a solution by swapping two adjacent cities in the permutation. For instance, given the permutation [1, 7, 2, 6, 4, 5, 3], some of the neighboring solutions are: [7, 1, 2, 6, 4, 5, 3] [1, 2, 7, 6, 4, 5, 3] [1, 7, 6, 2, 4, 5, 3] [1, 7, 2, 4, 6, 5, 3] [1, 7, 2, 6, 5, 4, 3]. The TSP problem, when tackled with brute force, exhibits an factorial time complexity of O(n!), with n representing the city count. As the factorial growth rate is exceptionally high, this approach becomes infeasible for larger city sets. Consequently, we turn to alternative exploration methods to swiftly obtain optimal solutions.

The TSP problem, when tackled with brute force, exhibits an factorial time complexity of O(n!), with n representing the city count. As the factorial growth rate is exceptionally high, this approach becomes infeasible for larger city sets. Consequently, we turn to alternative exploration methods to swiftly obtain optimal solutions.

Proposed Methodology

In order to find the best solution to the TSP problem, we use several algorithms, including hill climbing, Simulated Annealing (SA), the Genetic Algorithm (GA), as well as a Particle Swarm Optimization (PSO) algorithm. Proper comparisons of their results are then conducted.

Hill-Climbing Algorithm

A hill climbing algorithm is a local search algorithm that moves uphill in order to find the top of the hill or the best solution to the problem. It comes to an end when it reaches a point where no neighbor has a higher value. A hill climbing algorithm node has two components: state and value. When a good heuristic is available, hill climbing is commonly used. We don't need to maintain and manage the search tree or graph in this algorithm because it only keeps a single current state [13]. After reviewing various hill-climbing algorithms, the steepest hill-climbing algorithm was chosen for implementation in the TSP problem. This algorithm selects the optimal neighbor from among all possible neighbors and advances to a better current state. Each iteration examines a new local area at random, following the prescribed termination condition. The starting point in the hill climbing algorithm impacts the algorithm's performance. Furthermore, this algorithm might suffer the flat and shoulder problem. To resolve those issues, we have modified the algorithm in such a way that the best previously selected neighbour is repeatedly returned to the algorithm within the same iteration.

Simulated Annealing (SA) Algorithm

SA is an algorithm utilized to avoid being stuck at a local minimum or sub optimal solution. This algorithm chooses a random move rather than the best move. If a random move improves the condition, it takes the same path as before. Otherwise, the algorithm takes the path with a probability of less than one given in eq.(3) [14]. In this formula: P(vn) denotes the probability of accepting the new solution vn, eval(vc) represents the evaluation (cost) of the current solution vc whilst eval(vn) denotes the evaluation (cost) of the new solution vn, and T denotes the current temperature.

$$\boldsymbol{P(vn)} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{\frac{\boldsymbol{eval(vc) - eval(cn)}}{\boldsymbol{T}}}} \quad (\mathbf{3})$$

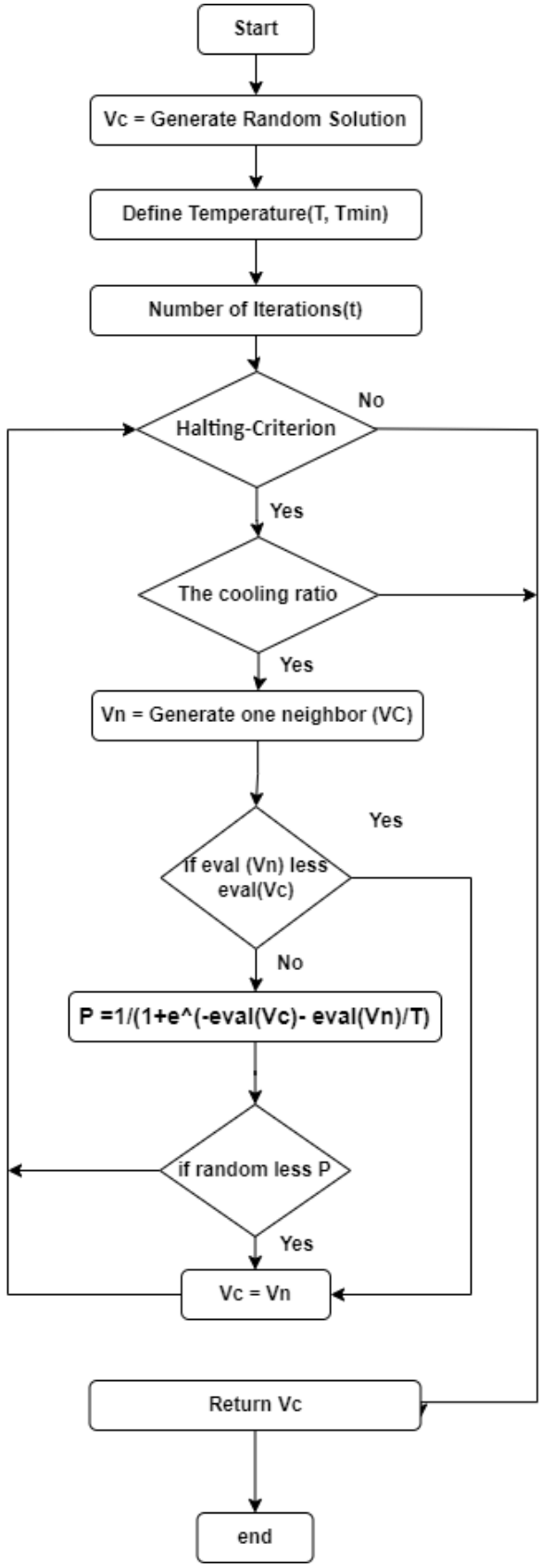


Figure (2): Flowchart of the Simulated Annealing algorithm for solving the TSP problem.

The moving direction must be determined probabilistically in each step in order to avoid becoming trapped in a local optimum and to progress toward the global optimum. While the search

process is progressing and approaching the final result, the size of the search step must be reduced. This allows to move quickly in the beginning and slowly in the end. The SA method requires that the search be continued until a good-enough solution is found or the stopping criteria are met. Furthermore, this method is sensitive to how its parameters, such as the search step and moving direction, are defined and tuned. Because the SA method is a heuristic search algorithm, it is sensitive to its starting point in the search space. A parameter T is also used to calculate the probability. It is analogous to the temperature in an annealing system. Uphill movements are more likely at higher T values. They become increasingly unlikely as T approaches zero, and the SA becomes hill climbing. T in a SA optimization problem begins high and gradually decreases according to the annealing schedule. To find the best solution when using the SA algorithm, it is critical to carefully select and tune the following parameters as shown in Fig 2:

1. The initial temperature. We typically start with a relatively high temperature and gradually reduce it to allow the algorithm to perform as much exploration as possible.
2. Termination Condition: The number of iterations is selected to be proportional to the number of cities. Such selection is essential to allow the algorithm to converge faster for a small-sized topology.
3. The cooling ratio: The temperature drops gradually, shrinking by 10% after each termination condition.
4. The frozen temperature: We have modified the algorithm to enhance the speed of exploration, prevent being trapped in local minima, and address convergence issues in each iteration. In order to expedite the search and learning process, we've implemented the following modifications: Control the decaying in the upper temperature towards the lower temperature. Introduce a variable value, which is initially set to be twice the number of cities. This variable decreases with each trial where no solution is accepted. The rate at which this variable decreases is tied to the reduction of the freezing temperature, ultimately reaching the desired level when the variable reaches zero.

Genetic Algorithm (GA)

The GA handles a population of possible solutions. Each solution is represented by a chromosome, which is merely an abstract representation. A set of reproduction factors is applied directly to the chromosomes, and they are used to perform mutations and recombination on the problem solutions in order to obtain the greatest variety of solutions and explore as much of the search space as possible without becoming stuck in the local minima[15].

Following the implementation of the GA algorithm, efforts were made to adjust and treat the following factors in order to achieve good results:

1. Initialization

The genetic algorithm process begins with the generation of a population of solutions. Each of the solutions is referred to as a chromosome. A chromosome contains or is defined by a set of parameters known as genes (cities). Using our algorithm implementation, we generated a random set of N solutions, which we call population. It is derived from the number of cities.

Here are some examples:

a. ***Gene:*** The solution's gene, City ID (6), is a chromosomal element.

b. ***Chromosomes:*** [6, 7, 4, 5, 1, 2, 3]

c. ***Population:*** [1, 7, 2, 6, 4, 5, 3] [7, 1, 2, 6, 4, 5, 3] [1, 2, 7, 6, 4, 5, 3] [1, 7, 6, 2, 4, 5, 3]

2. Fitness Assignment

The fitness function is used to assess a solution's ability to compete with other solutions. Solutions are evaluated based on their fitness function at each iteration. This degree also influences the likelihood of reproductive selection. The greater the fitness score, the greater the likelihood of reproductive selection. As previously stated, the problem's evaluation function is the sum of the distances between all cities. The evaluation function was converted into probability to calculate the fitness function with normalization. Thus, given a solution P, the fitness value of this solution is inversely proportional to its evaluation (the total distance), as given in eq.(4).

$$\boldsymbol{f(P) = \frac{1}{eval(P)}} \quad \textbf{(4)}$$

3. Selection

The selection phase involves the selection of individuals to reproduce offspring. The selected individuals are then arranged in pairs of two to increase reproduction. These individuals then pass on their genes to the next generation. There are several types of selection methods available. In our work, we have selected a roulette wheel to select pairs of solutions based on a fitness function, It is also called fitness proportionate selection, since the probability to select a solution P , denoted by P rob(P), is proportional to its fitness value, as given in eq.(5), wherein, the sum of probabilities for all solutions in the population Spop is utilized to normalize the probability. [16].

$$\boldsymbol{Prob(P) = \frac{f(P)}{\sum_{\forall P_i \in S_{pop}} f(P_i)}} \quad \textbf{(5)}$$

4. Reproduction

The reproduction stepchild after the selection process. Two distinct factors play a role in changing the genetic makeup of the next generation in this step.

a. Crossover: The intersection is crucial in the reproductive stage of the genetic algorithm. The crossover operator mixes the genetic information of two current-generation parents to create a new individual which represents the offspring. The mixing nature is determined by the crossover mechanism. There are many types of crossover to solve the traveling salesman problem, the most popular of which are order crossover (OX), partially matched crossover (PMX), and cycle crossover (CX). We have utilized the PMX in our work because it provides a high level of diversity and gene exploration while preserving the genes of the parents [17]. A single individual (a potential solution) might be selected for mating, and it can potentially mate with itself. PMX is applied within genetic algorithms through the following sequence: Start with two initial solutions, two positions are randomly selected within the permutation of both parents. Then, proceed to transfer the segment between the specified crossover positions from one parent to the corresponding positions within the offspring. To finalize the offspring's formation, integrate values from the other parent into the remaining positions, ensuring no duplication with those already found in the copied segment from the first parent. As per the research methodology, this process yields two offspring, inheriting genetic information from both parents while concurrently broadening the diversity of potential solutions [18].

An example of a PMX crossover between two parents to produce offspring is as follows, where | denotes a crossing point:

i. The first parent: [2, 7,| 6, 3, 5,| 1, 4]

ii. The second parent:[5, 2,| 4, 1, 3,| 6, 7]

iii. offspring:[2, 7, 4, 1, 3, 5, 6]

iv. offspring:[1, 2, 6, 3, 5, 4, 7]

b. Mutation: To maintain population diversity and solve the problem of early convergence, a mutation agent introduces random genes into the offspring (a new baby). We manipulated genes and chromosomes in the TSP problem by swapping city locations. We examine all permutations in order to find the best solution, not just switching cities.

5. Termination

The algorithm concludes its execution either when it reaches the maximum number of iterations or when it enters the convergence stage. This convergence stage is characterized by a distinct lack of diversity among solutions within a single generation's reproductive phase.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a computational technique inspired by the social behavior of organisms, to optimize objective functions. PSO algorithm has been effective in solving various optimization problems, including the Traveling Salesman Problem (TSP), where each particle represents a potential solution. The algorithm operates on a population of particles, each influenced by its own and the swarm's best solution found.

Following the implementation of the PSO algorithm, efforts were made to adjust and treat the following factors in order to achieve the best results:

1. Particle Initialization: The algorithm starts by initializing a swarm of particles. Using our algorithm implementation, we generated a random set of N Particles that is proportional to the number of cities.
2. Fitness Evaluation: The fitness evaluation is conducted in PSO for the TSP. Fitness is based on the length of the tour, with updates made when a new best solution is found Using the previous Equation 1.
3. Update particles: At each iteration, the particles are updated by adjusting their velocities and positions based on their current positions and the best positions found by the swarm.
   a. Update Velocity: The velocity adjustment process in PSO for optimizing objective functions as with Equation 6.
   b. Update Position: The position of each particle is then updated by adding its velocity to its current position as given in eq. (7). The new position at time (t+1) is updated by adding the current position (pt) to the velocity (vt) at time t+1.

$$\boldsymbol{v}[\boldsymbol{t}+\mathbf{1}] = \boldsymbol{w}\cdot\boldsymbol{v}[\boldsymbol{t}] + \boldsymbol{c_1}\cdot\mathbf{rand}\cdot(\boldsymbol{p}_{\mathbf{Best}} - \boldsymbol{p}) + \boldsymbol{c_2} \cdot \mathbf{rand}\cdot(\boldsymbol{g}_{\mathbf{Best}} - \boldsymbol{p}) \quad (\mathbf{6})$$

$$\mathbf{p_{t+1}} = \mathbf{p}\boldsymbol{t} + \mathbf{v}\boldsymbol{t} + \mathbf{1} \quad (\mathbf{7})$$

The notation used for PSO, where p represents the current position, v is the current velocity, pBest is the best position found by the particle, gBest is the best position found by the swarm, and the rand is a random variable.

4. PSO Parameters: The PSO algorithm involves tuning several parameters. In this particular application, the number of particles and iterations are determined based on the number of cities. The values of c1 and c2 are both set to 1.5, while w is set to 0.7, as these parameter values have demonstrated successful performance in a variety of optimization problems in the literature.
5. Termination: The algorithm terminates when a stopping criterion is met, such as a maximum number of iterations.

Experimental Results and Analysis

Python programming language and Jupyter Notebook were utilized to conduct experiments on various algorithms using different benchmarks for solving the TSP with varying numbers of cities. A structured dataset was generated for each number of cities and used as input for all the algorithms tested. The experiments were conducted separately for each number of cities, and the resulting data was processed and analyzed.

The distances were randomly selected from a set of values categorized as relatively short, medium, or large. To ensure that all cities were connected to each other, we created a fully connected unidirectional graph, with equal distances between cities in both directions. The data was then read and converted to a data frame using the Pandas library. Additional libraries, such as NumPy, Random, and Matplotlib, were also utilized in the analysis. This study addresses the TSP with a large number of cities, aiming to find the most efficient route for a vendor using various optimization algorithms.

The results of these algorithms were recorded, including the number of iterations, time to find the optimal solution, number of cities, and evaluation function. Table I presents a comprehensive overview of the recorded results from implementing the algorithms multiple times with different numbers of cities.The algorithms included in the comparison are: Steepest Ascent Hill Climbing (SA-HC) algorithmm, Genetic Algorithm (GA), Particle Swarm Optimization (PSO) algorithm, the classical Simulated Annealing (C_SA) algorithm, and the proposed enhanced version of the SA algorithm (Enhancement SA).

The plots depicted in Fig 3 and Fig. 4 illustrate the convergence of different optimization algorithms toward optimal solutions and their ability to surmount local minima for a TSP benchmark of 100 cities. Our analysis delves first into the performance of the hill climbing algorithm, which faces challenges in escaping local optima but yields satisfactory outcomes with fewer cities. However, when dealing with a larger number of cities, the algorithm encounters greater difficulty and becomes trapped in a local minimum, as shown in the fluctuations in the SA-HC curve in Fig 3. In contrast, the enhanced simulated annealing algorithm delivers remarkable results, demonstrating smooth convergence towards the optimal solution for both small and large city counts. This is evident in the second set of plot in Fig 3. On the whole, the genetic algorithm consistently achieves excellent results and performs similarly to the simulated annealing algorithm in all scenarios. However, it tends to require more time, especially when dealing with a larger number of cities, as indicated by the third plot in Fig 3. As for the Particle Swarm Optimization (PSO) algorithm, it shows promising results with quick convergence when dealing with fewer cities, but it encounters challenges as the city count increases. One of the main issues with the PSO algorithm is its inclination to be confined to a local optimum and its struggle to escape from it. In summary, most stages lead to achieving the optimal solution, but there is some variability in the speed and computational cost associated with each algorithm's evolution to reach the solution as shown in Fig.4. The time complexity of various algorithms is depicted in Fig.4 and Fig.5. All algorithms

exhibit a semi-quadratic time complexity except the enhanced simulated annealing algorithm which exhibits a semi-linear time complexity. This reflects remarkable proficiency in discovering highly efficient solutions within a relatively short time frame, maintaining consistency across different city counts.

Thus, while optimal solutions were achieved in the majority of scenarios, as illustrated in Fig.5 and detailed in Table I during a comparison with the classical simulated annealing algorithm, it outperformed the latter in terms of achieving optimal solutions and in the development of the fitness function and the complexity of the run time. This performance improvement came with only a marginal difference in execution time.

Table (1): A comparison between various algorithms for solving TSP.

Optimization Algorithm		Fitness Function Evolution					Run Rime Complexity (sec)				
Num Cities	Max Iterations	SA-HC	GA	POS	C_SA	Enhancement SA	SA-HC	GA	POS	C_SA	Enhancement SA
7	**21**	104	**61**	**61**	**61**	147	0.04	0.47	**0.29**	0.38	0.13
20	**60**	208	186	245	241	173	1.9	9.01	5	1.6	**1.37**
50	**150**	479	475	757	681	427	70.03	205.02	75.82	7.52	**7.19**
100	**300**	1167	918	1880	966	741	1264.21	3272.7	1288.15	33.25	**28.87**

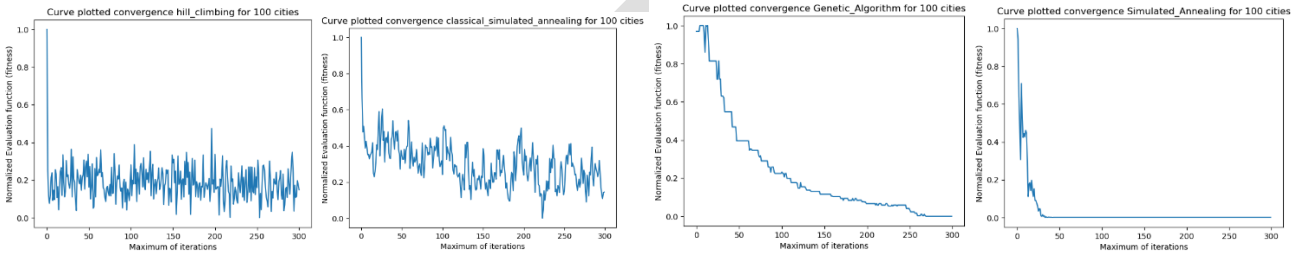


Figure (3): Convergence curves for different algorithms for a Benchmark of 100 cities. The curves shown (from left to right) represent: hill climbing algorithm, the enhanced SA algorithm, GA, and the PSO algorithm).

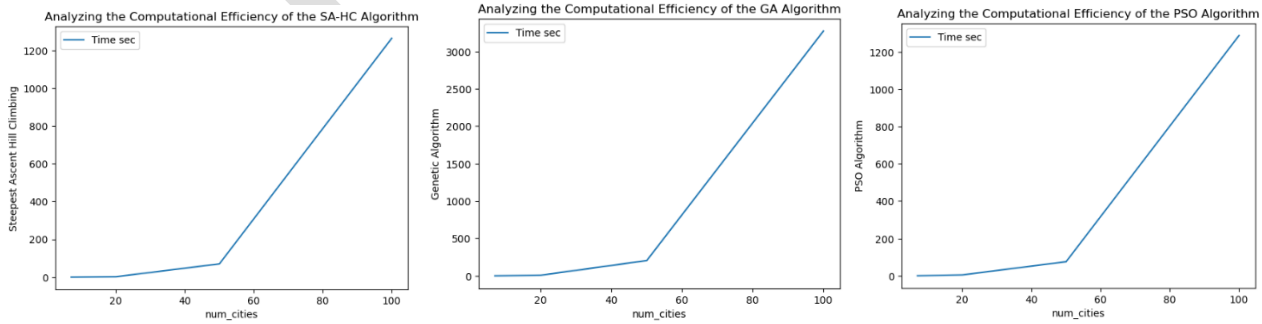


Figure (4): Time complexity for different algorithms (fromt left to right): hill climbing, GA, and PSO.

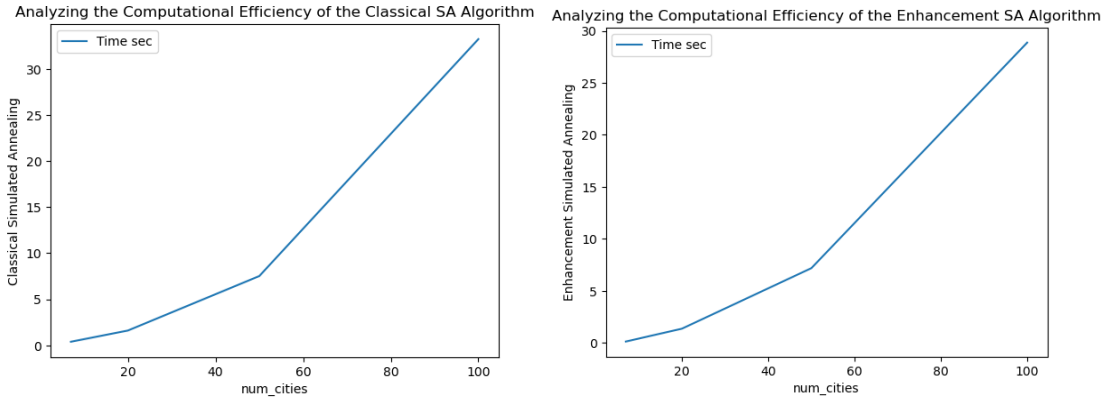


Figure (5): A comparison between the classical SA and the enhanced SA algorithms in terms of time complexity.

Conclusion AND FUTURE WORK

The Traveling Salesman Problem (TSP) is a computational problem that seeks to determine the shortest path between multiple points. Numerous heuristics and optimization techniques have been proposed to generate probabilistic results for solving this problem. The aim is to find algorithms that can provide efficient solutions for a large number of cities, in a timely manner and at a minimal cost. In this study, we explored several algorithms, including the hill-climbing, simulated annealing (SA) algorithm, the genetic algorithm with partially matching cross PMX, and the particle swarm optimization algorithm. We discussed their ability to solve optimization problems, avoid local minima, and balance search exploration and exploitation to find an optimal solution. Our results show that the SA and genetic algorithms produced excellent results for a relatively large number of cities. We plan to introduce other algorithms for comparison in future work.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

Not applicable.

Author's contribution

The authors confirm contribution to the paper as follows: study conception and design: Ahmed Awad, theoretical calculations and modeling: Younes Khdeir; data analysis and validation|: Younes Khdeir, Ahmed Awad, manuscript preparation: Younes Khdeir, Ahmed Awad. All authors reviewed the results and approved the final version of the manuscript.

Funding

Not applicable.

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this article

Acknowledgements

Not applicable.

References

1] Toufik Mzili, Mohammed Essaid Riffi, and Ilyass Mzili. Artificial rat optimization with decision-making: A bioinspired metaheuristic algorithm for solving the traveling salesman problem. Decision Making: Applications in Management and Engineering, 2023.

2] A Rehash Rushmi Pavitra, I Daniel Lawrence, and A Muthukrishnan. A secure quantum technology for smart cities using travelling salesman problem (tsp): Application perspectives. In Handbook of Research on Quantum Computing for Smart Environments. IGI Global, 2023.

3] Hua Yang and Ming Gu. Learning tsp combinatorial search and optimization with heuristic search. In Neural Information Processing: 29th International Conference, ICONIP 2022, Virtual Event, November 22–26, 2022, Proceedings, Part IV, pages 409–419. Springer, 2023.

4] S Purusotham, T Jayanth, T Vimala, and K Ghanshyam. An efficient hybrid genetic algorithm for solving truncated travelling salesman problem. Decision Science Letters, 11(4):473–484, 2022.

5] Rahul Jain, Kushal Pal Singh, Arvind Meena, Kun Bihari Rana, Makkhan Lal Meena, Govind Sharan Dangayach, and Xiao-Zhi Gao. Application of proposed hybrid active genetic algorithm for optimization of traveling salesman problem. Soft Computing, 27(8):4975– 4985, 2023.

6] Anurup Naskar, Rishav Pramanik, SK Sabbir Hossain, Seyedali Mirjalili, and Ram Sarkar. Late acceptance hill climbing aided chaotic harmony search for feature selection: An empirical analysis on medical data. Expert Systems with Applications, 221:119745, 2023.

7] Kanchan Rajwar, Kusum Deep, and Swagatam Das. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artificial Intelligence Review, pages 1–71, 2023.

8] Przemysław Kowalik, Grzegorz Sobecki, Piotr Bawoł, and Paweł Muzolf. A flow-based formulation of the travelling salesman problem with penalties on nodes. Sustainability, 15(5), 2023.

9] Jakub Ptasznik. Solving traveling salesman problem using simulated annealing algorithm. PhD thesis, Instytut Informatyki, 2023.

10] Bijun Wang, Zheyong Bian, and Mo Mansouri. Selfadaptive heuristic algorithms for the dynamic and stochastic

orienteering problem in autonomous transportation system. Journal of Heuristics, 2023.

11] Tibet Duman and Ekrem Duman. Solving a new application of asymmetric tsp by modified migrating birds optimization algorithm. arXiv preprint arXiv:2301.06914, 2023.

12] Zhiding Dong and He Chen. Design of composite structure optimization model based on particle swarm optimization. In Proceedings of the 2nd International Conference on Cognitive Based Information Processing and Applications (CIPA 2022) Volume 2, pages 357–365. Springer, 2023.

13] Stefan Wintersteller, Martin Uray, Michael Lehenauer, and Stefan Huber. Improvements for mlrose applied to the traveling salesperson problem. In Computer Aided Systems Theory–EUROCAST 2022: 18th International Conference, Las Palmas de Gran Canaria, Spain, February 20–25, 2022, Revised Selected Papers, pages 611–618. Springer, 2023.

14] Deniz Dal and Esra Celik. Investigation of the impact of different versions of gcc on various metaheuristic-based solvers for traveling salesman problem. The Journal of Supercomputing, pages 1–47, 2023.

15] Martin Uray, Stefan Wintersteller, and Stefan Huber. Csrx: A novel crossover operator for a genetic algorithm applied to the traveling salesperson problem. arXiv preprint arXiv:2303.12447, 2023.

16] K Suresh, Briskilal Joseph, et al. Patient scheduling system for medical treatment using genetic algorithm. Journal of Population Therapeutics and Clinical Pharmacology, 30(8):268–273, 2023.

17] Xin-Ai Dou, Qiang Yang, Xu-Dong Gao, Zhen-Yu Lu, and Jun Zhang. A comparative study on crossover operators of genetic algorithm for traveling salesman problem. In 2023 15th International Conference on Advanced Computational Intelligence (ICACI), pages 1–8, 2023.

18] Ting Huang, Yue-Jiao Gong, Sam Kwong, Hua Wang, and Jun Zhang. A niching memetic algorithm for multisolution traveling salesman problem. IEEE Transactions on Evolutionary Computation, 24(3):508–522, 2020.