IAM-TSP: Iterative Approximate Methods for Solving the Travelling Salesman Problem

Esra’a Alkafaween¹, Samir Elmougy², Ehab Essa³, Sami Mnasri⁴, Ahmad S.Tarawneh⁵, Ahmad Hassanat⁶
E-Learning and Education Resources Center, Mutah University, Karak, Jordan¹
Department of Computer Science, Mansoura University, Mansoura, Egypt², ³
Department of Computer Science, University of Tabuk, Tabuk, Saudi Arabia⁴
Computer Science Department, Mutah University, Karak, Jordan⁵, ⁶

Abstract—TSP is a well-known combinatorial optimization problem with several practical applications. It is an NP-hard problem, which means that the optimal solution for huge numbers of examples is computationally impractical. As a result, researchers have focused their efforts on devising efficient algorithms for obtaining approximate solutions to the TSP. This paper proposes Iterative Approximate Methods for Solving TSP (IAM-TSP), as a new method that provides an approximate solution to TSP in polynomial time. This proposed method begins by adding four extreme cities to the route, a loop, and then adds each city to the route using a greedy technique that evaluates the cost of adding each city to different positions along the route. This method determines the best position to add the city and the also the best city to be added. The resultant route is further improved by employing local constant permutations. When compared to existing state-of-the-art methods, our experimental results show that the proposed method is more capable of producing high-quality solutions. The proposed approach, with an average approximation of 1.09, can be recommended for practical usage in its current form or as a pre-processing step for another optimizer.

Keywords—Greedy algorithms; TSP; NP-hard problems; polynomial time algorithms; combinatorial problems, optimization methods

I. INTRODUCTION

At the beginning of the seventeenth century, Thomas Pennington and William Hamilton modeled the first mathematical problem corresponding to the Traveling Salesman Problem (TSP). This problem represents a game in which the winner connects twenty points by moving from one point to another, following some precise paths. This game/problem is called Hamilton Circuit Theory [1]. Afterward, in graph theory, TSP becomes a problem of identifying the optimal (shortest) Hamiltonian cycle visiting a set of cities (points) using a matrix of distances between the cities. TSP is one of the classic problems of combinatorial optimization, and it is widely investigated and considered a standard for assessing the performance of computational methodologies and algorithms.

The principal objective of TSP is that the salesman visits all the cities in the minimal tour and then returns to the starting point with the assumption that the distances between the cities are known and the necessity of visiting each city is only once. Despite high-size instances of TSP being resolved in the literature, TSP is proven to be NP-hard even for small-size instances [2].

The theoretical and practical relevance of the TSP problem comes from the fact that numerous applicable real-world and engineering problems can be solved by adapting the TSP solutions, such as circuit design [3], scheduling [4], and DNA [5, 6], in addition to logistics and transportation and even space exploration missions [7]. Therefore, the study of TSP has significant theoretical and practical value, leading to cost savings and other benefits.

Moreover, the application fields of TSP include any problem involving the search for the shortest paths. These fields vary in domains from big data classification [8, 9, 10], deployment and routing of IoT networks [11, 12, 13, 14, 15, 16], financial prediction [17, 18], image processing [19], and [20], computer vision [21], [22], and [23]. TSP can be defined by finding a Hamiltonian cycle that visits each vertex precisely once, with the least amount of total weight feasible, given a full undirected weighted graph G = (V, E) with vertex set V and edge set E. The total weight of a cycle is equal to the weights of each of its individual edges. The book in [24] presents a comprehensive mathematical and theoretical analysis of the calculability and complexity of the TSP. One of the critical issues of methods of resolution of TSP is the enhancement of the accuracy of the algorithm to rapidly find the optimal or near-optimal solutions.

Indeed, heuristics and meta-heuristics are the most successful methods used to resolve the TSP [25, 26, 27]. In this regard, Genetic Algorithms (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) are among the most commonly used algorithms to resolve TSP. GA was successfully used for problems involving global search due to its rapid convergence and fast search process. However, it has some issues and has poor performance when achieving the local search. ACO is another robust optimizer with a high-resolution capacity. However, its convergence performance highly depends on the initial parameters, mainly the appropriate initial quality of the pheromone. PSO is characterized by its ease of use since it has a few numbers of initial parameters to set. However, other optimizers give better results for many TSP test problems. SA is another optimizer known for its capacity to avoid local optima and is often hybridized with other algorithms for this capacity.
However, like with ACO, the initial parameter values have a significant impact on the search process. Despite substantial research on TSP, its combinatorial nature suggests that there is still opportunity for development in this field, motivating the work of this paper. TSP is inextricably tied to handling other combinatorial problems with comparable characteristics, such as the Traveling Salesman Problem with Time Windows (TSPTW), Vehicle Routing Problem (VRP), and Capacitated Vehicle Routing Problem (CVRP). TSP approaches and algorithms are frequently used as a foundation for addressing these related optimization challenges. As a result, progress in TSP solving may provide useful insights and tactics relevant to a broader class of combinatorial problems in logistics, transportation, and network optimization. Because these challenges are interdisciplinary, knowledge and approaches can be transferred across fields, creating a more thorough grasp of combinatorial optimization.

The major contribution of this paper lies in presenting an Iterative Approximate Method for Solving TSP (IAM-TSP), which provides a polynomial-time approximate solution to the TSP. The proposed method begins by including four extreme cities along the path. Following that, a loop successively adds each city to the route by calculating the cost at various points along the route. The approach considers each city to add and finds the most suitable location to put it, the one that minimizes the total cost. This method is improved further by using local constant permutations on the output of IAM-TSP, which considerably improves the final route, referred to as IAM-TSP+.

We evaluated the proposed algorithms against standard TSP datasets to determine how they performed in terms of key performance measures. The proposed IAM-TSP performed nearly identically to some of the typical TSP approximation algorithms given in the results section, but the proposed IAM-TSP+ surpassed all approximation methods compared on the majority of TSP instances.

The rest of the paper is organized as follows. Section II illustrates and discusses the advantages and drawbacks of the recent most relevant studies resolving the TSP. Section III identifies and investigates the proposed methodology for resolving the TSP. Section IV presents the experimental setup and the results, Section V concludes the study.

II. RELATED WORK

Given TSP's extensive history, its cutting-edge landscape is extremely diverse. To solve the TSP, several techniques, paradigms, and approximations algorithms have been used. Heuristics, for example, are a type of approximation method aimed at finding near-optimal solutions to NP-hard problems in polynomials time. One popular and simple method for building a TSP tour involves starting the tour at any node, traversing minimum-cost arcs to each successive node until all nodes are visited, and then returning to the starting node to finish the tour. This method is known as the Nearest Neighbor heuristic [28].

There are also many popular methods, such as: the 2-Opt heuristic [29], Farthest Insertion algorithm [30], Nearest Insertion algorithm [30], Cheapest Insertion algorithm [30], Arbitrary Insertion algorithm [31], Repetitive Nearest Neighbor algorithm [32], Concave hull with heuristics, and Concave Hull No Heuristic [33]. In what follows, the main recent studies proposing TSP resolution methodologies are investigated: The study in [33] introduces a concave hull-based algorithm to resolve the Euclidean symmetric TSP by establishing concentric hulls and then merging them in one tour. Two novel metrics are suggested: the “Average Waiting Distance” and the “min AWD” of a tour. The results show that an optimal AWD does not guarantee the optimal tour.

In study [34], the authors enhanced the accuracy of TSP solutions for numerous sizes of the problem. The used algorithm is a modified ACO with a better convergence during the TSP search process. To prevent trapping into local optima, this algorithm decreases the high concentration of pheromone during the route selection. The diversity in the algorithm is ensured using entropy weighted learning. According to the results, this work improved ACO and solved TSP better than the standard ACO.

A new variant of TSP, called TSPJ, which includes the schedule of jobs in a set of positions, was introduced in study [35]. Since in TSP, the transport time is longer than the operation time, the considered objective in TSPJ is to minimize the make-span, equal to the needed time to achieve the longest job. The resolution of the TSPJ involves four local search methods. Using the CPLEX system, the results indicated that the solutions given by the four used heuristics are too close to the optimal (a gap less than 6%). In the same regard, the study in [36] aims to resolve the TSPJ. Applicative and practical contexts of TSPJ had been discussed, as well as the parameters specific to TSPJ such as the completion time, configuration time, and resource variation.

In study [37], the authors used the aim was to resolve a TSP variant called the multiple TSP (mTSP) problems using a hybrid algorithm. The latter relies on an EAX heuristic to optimize the intra-tour and a tabu search neighborhood search to optimize the inter-tour. The objectives of the problem were the minimization of both the longest path and the total traveled distance. To reduce the neighborhood search time, a reduction approach is proposed to avoid computing the nonpromising possible solutions. The experiments involve a comparison with five approaches tested on 41 known TSP test problems and 36 new large-size ones. However, the other variants of TSP were not assessed, and comparisons with other classes of heuristics were not achieved.

The study in [38] suggests a new seriation strategy named “tree-penalized Path Length” (tpPL). Data seriation is a famous problem in data analysis. Its the process of sequencing and ordering data according to their similarity. TSP in this study is considered a seriation method. The goal is to linearly order the data using simultaneously the TSP, tpPL, and optimal leaf order (OLO) methods. Optimal paths are transferred from TSP to OLO. In terms of computational complexity, TSP and tpPL have the same order of complexity. Hence, TSP heuristics may be used to resolve the tpPL. Tested on more than forty datasets, the tpPL has a better performance than TSP and OLO with the same computational complexity. The study in [39] introduces a compression-based TSP heuristic for data micro-aggregation.
Micro-aggregation is a method for disrupting and aggregating personal data using the concept of k-anonymity. By simultaneously considering the respect for privacy and the usefulness of data, the introduced TSP heuristic was the most efficient in solving the problem of micro-aggregation. In contrast, heuristics relying on TSP encrust scalability issues. Unlike other heuristics, the algorithm proposed in this study can reduce the execution time of the TSP. The tests carried out on small and medium-sized data affirm the trade-off between computation time and the rate of loss of micro-aggregation information.

In study [40], a new problem, named the traveling thief problem (TTP), is proposed by combining the knapsack problem (KP) and the TSP. The PTT resolution method relies on sequences selection heuristics. A set of selection operators/thieves are involved to select the cities/objects to progressively create the tour.

The study in [41] introduces two new TSP versions. Named pollution TSP (PTSP) and energy minimization TSP (EMTSP), the new versions add environmental constraints to the TSP. The aim of PTSP is to reduce fuel consumption, and carbon emissions. The aim of EMTSP is to reduce the cost of a trip according to the distance and the carried load. The method of resolution of the two TSP variants, MILP-GA-LS, relies on a mixed integer linear programming model to identify initial possible solutions, then multi-operator GA to enhance the found solutions. Afterward, an iterative local search algorithm was used to enhance the solutions. However, numerous constraints were not considered such as the time-window of customers. Moreover, an issue arises concerning the complexity of the introduced approach, since it was proven that, for small-size instances of PTSP and EMTSP, exact methods give better results than the MILP-GA-LS.

In study [42], a new ACO variant is proposed to solve the TSP. Named DAACO; this algorithm dynamically changes the number of ants to avoid falling into local optima and to prevent long time of convergence. Besides, DAACO uses a local selection process to enhance the quality of ants and the needed time for the search process. Among the used twenty TSPLIB test problems, all the DAACO solutions were optimal except one. These results confirm the advantageous quality of solutions and time of convergence of DAACO compared to other optimizers.

In study [43], a hybrid Ant Colony (AC)-Tabu Search (TS)-Firefly Algorithm (FA) called ACTS-FATS is proposed. The hybridization avoids the probability of premature convergence, then, reduces the chance of trapped in local optima. Tested with the TSPLIB95, the hybridization does not generate additional execution time compared to AC, TS and FA.

In study [44], Dhouib-Matrix, a column-row method, is proposed to resolve polynomial time TSP. The process of this method is as follows: after defining the distance matrix, a start position is selected to choose rows. Then, columns are discarded, and the route is transformed to a tour. The advantage of the introduced Dhouib Matrix is that it needs only n iterations to find the route between n cities.

In study [45], a comparative study is proposed to discuss the recent algorithms and methodologies used to resolve the TSP using metaheuristics. The focus is set on the numerous versions of the BA, FA and PSO optimizers.

In study [46], a new version of the TSP, called TSP-D is proposed. The latter is a classic TSP that involves a truck and a drone (Unmanned Aerial Vehicles (UAV)). In TSP-D, it is assumed that UAV has low battery capacities and can transport cargo per flight. The issue is to determine which UAV should serve which client. The tests demonstrate that the time of service can be reduced by using the UAVs with distinct speeds according to the cargo weight. However, the used samples (number of clients) are between 30 and 60, which makes the method valid only for small-size TDP-D instances.

The authors in [47] propose an algorithm to minimize travel costs and maximize the overall profit. Simulated annealing (SA) and genetic algorithm (GA) with dedicated mutation operators were used. A concept of tour plots is used to generate the final solutions. In terms of computation time, the SA is better than the GA.

In study [48], Qi-ACO, an ACO based on quantum computing, is developed to resolve the four-dimensional TSP (4DTSP). Fuzzy type-2 variables are used due to the uncertain aspect of the investigated problem of travel emissions and costs. The 4DTSP is characterized by the existence of numerous paths and conveyances between the cities. A process is implemented in Qi-ACO for generating qubits considering the constraints of carbon emission, cost, and time. The initialization and update of pheromone is based on qubit. A faster computation is achieved in Qi-ACO due to the quantum calculations. Performed statistical tests to confirm the performance of the introduced approach. However, numerous constraints are not considered such as vehicle speed and route selection. The readers are referred to [49] for more comprehensive methodologies on modeling and designing greedy heuristic methods to resolve the TSP.

III. THE PROPOSED ITERATIVE APPROXIMATE METHODS FOR SOLVING TSP (IAM-TSP)

The IAM-TSP is proposed in this paper as an approximation to the TSP, which has a greedy algorithmic nature. It begins by selecting four cities from the input set to represent the east, north, west, and south most positions. These cities are added to the route list in the following order: east, north, west and south, followed by a return to the beginning point (east). The algorithm then enters a loop, adding each remaining city to the path one by one. The method calculates the cost of adding a city to various points along the route throughout each iteration. It carefully investigates all feasible positions for each city, picking the best position and city at the lowest cost. Finally, this approach computes the route's total cost and outputs the final result.

The cost is calculated using the Euclidean distance between the consecutive cities on the route. Algorithm 1 shows the pseudocode processes of the proposed IAM-TSP. Fig. 1 illustrates the progress of the IAM-TSP solution of the Rat195 LIBTSP real-world problem [50]. Because the TSP is a
minimization problem, the term “best” refers to the smallest value throughout this paper, i.e., minimum route.

According to Algorithm 1, the proposed IAM-TSP has a time complexity of $O(n^2)$, where $n$ is the number of cities in the Cities xy list. This is because, in the while loop, the algorithm finds the best position to put each city from the Open list into Rout and checks every feasible solution, which takes $O(n)$ time. This operation is repeated until Open is empty, which means that in the worst scenario, the while loop runs $n$ times, giving a total time complexity of $O(n^2)$.

The space complexity of the solution is $O(2n + n^2)$, where $n$ is the number of cities in the Cities xy list. This is because we use two lists Rout and Open of size $n$, which take $O(2n)$ space, and the quadratic space comes from the space reserved by the Possible Solutions matrix, whose size is $n$ rows, each hosting $n$ cities. The rest of the data structures used in the solution take $O(1)$ or $O(n)$ space, so the overall space complexity can be asymptotically approximated to $O(n^2)$. However, if we establish a square matrix storing the distances between each city and the others, space complexity stays asymptotically quadratic too; this is not done in this study, but it is a typical approach to removing the burden of distance computation. It should be emphasized that IAM-TSP provides an approximation solution for the TSP, and it does not ensure finding the optimal solution, because finding the best position for each city alone does not guarantee finding the optimal solution, which necessitates the involvement of all cities at the same time.

In order to improve the IAM-TSP performance, and because involving all cities makes the problem NP-hard, we opt for involving a constant number of local cities ($k$), and calculate all the permutation sequences starting from the first city in the output Rout until the $k$ city, finding the best solution and updating the Rout during this process, after which the algorithm goes into a loop moving by one city to find the next $k$ permutations until $n - k$, this enhanced version is called IAM-TSP+. Algorithm 2 shows the pseudocode processes of the proposed IAM-TSP+, and Fig. 2 depicts the IAM-TSP+ resultant Route of the ATT48 TSPLIB real-world problem [50] in comparison to its optimal route.

**Algorithm 1: The proposed IAM-TSP algorithm**

Require: Cities xy (List of cities with x and y coordinates) of size $n$ (number of cities)
Ensure: Cost (Total cost of the route using Euclidean distance), and Route (the best possible sequence of cities)

1: Create an empty list Rout to store the order of visited cities.
2: Get the East, North, West, and South cities and add them to the Rout list. This is done using the minimum/maximum of x and y coordinates.
3: Initialize a Boolean list Visited of size $n$ (number of cities) and set all elements to false.
4: Set the Visited status of the cities in Rout to true (East, North, West, and South).
5: Create an empty list Open to store the cities that have not been visited.
6: Add all cities in Cities xy to Open if their Visited status is false.
7: while Open is not empty do
8: Create an empty list of list Possible Solutions to store the possible solutions (routs).
9: for each city i in Open do
10: Find the best location to insert it into Rout and add the new route to Possible Solutions.
11: end for
12: Find the best solution Best in Possible Solutions and keep track of the city ID.
13: Update Rout by Best.
14: Remove the city ID from Open that satisfies Best.
15: end while
16: Calculate the cost of the Rout using the Euclidean distance and store it in Cost.
17: return Cost, Rout

**Fig. 1.** The progress of the IAM-TSP solution of the Rat195 LIBTSP real-world problem, the progress is made from left to right and from top to bottom, and the results are displayed after ten iterations from the previous state.
Algorithm 2: The proposed IAM-TSP+ algorithm

Require: Cities xy of size n and k=5 (local permutations)
Ensure: Cost, Route
1: Route=IAM-TSP(Cities xy)
2: for i = 1 to n − k do
3: Create an empty list of list Possible Solutions
4: Possible Solutions=all k local permutations of Route(from i to i + k)
5: Update Cost(Possible Solutions)
6: index = argmin(Cost)
7: Route = Possible Solutions[index]
8: end for
9: return Cost(Route), Route

According to Algorithm 2, the proposed IAM-TSP+ has a time complexity of $O(n^2 + n.k!)$, where n is the number of cities in the Cities xy list. This is because the improved version uses the proposed IAM-TSP as an initial solution, this time is added to the time consumed by sliding the k permutations along the Route, which consumes $O(n.k!)$ in the worst scenario, giving a total time complexity of $O(n^2 + n.k!)$. Since k is a constant number, the time complexity of $O(n.k!)$ is often greater than that of $O(n^2)$. $O(n.k!)$ may still be more efficient than $O(n^2)$ for small values of k, but as (k) grows higher, the growth rate of $O(n.k!)$ quickly exceeds that of $O(n^2)$. Therefore, the time complexity of IAM-TSP+ can be asymptotically approximated to $O(n.k!)$. The space complexity of the IAM-TSP+ is similar to that of the proposed IAM-TSP, with the exception of the space required by the possible Solutions matrix, which contains k! rows of routes, each of which hosts n cities, resulting in a total space complexity of $O(n^2 + n.k!)$, which can be asymptotically approximated to $O(n.k!)$. This is a problem for machines that have limited memory resources, particularly when k is large, and therefore, k needs to be decided based on the available memory resources.

IV. EXPERIMENTAL SETUP AND RESULTS

To verify the quality and effectiveness of the proposed methods for solving TSP, IAM-TSP and IAM-TSP+ were applied to nine TSP instances, each with a known optimal solution. Those TSPs are from the TSPLIB [50], which has vertices between 40 and 500, namely: a280, att48, berlin52, KroA100, ch150, ch130, pr76, lin105, and pcb442. We chose these specific instances to facilitate comparison to other methods that have been repeatedly used in many studies.

We compare the performance of the proposed methods to other related methods that proposed in the recent years; these include:

- NN: Nearest Neighbor algorithm
- FI: Farthest Insertion algorithm
- CH: Concave hull with Heuristic.
- CNH: Concave hull without Heuristic.
- NI: Nearest Insertion algorithm.
- CI: Cheapest Insertion algorithm
- AI: Arbitrary Insertion algorithm
- RNN: Repetitive Nearest Neighbor algorithm
- 2-Opt: 2-Opt algorithm

It deserves to be noted that the aforementioned methods were not developed for this work; rather, we directly reference their results on each standard TSP as stated by [33], where all parameters used for each method can be obtained. It is also worth noting that the majority of these methods’ results were obtained by repeating each method a number of times and reporting the average performance. However, we do not need to do the same for the proposed methods because each produces the same result on a specific TSP regardless of how many times the method is run.

We performed simulation experiments using C# of Microsoft visual studio 2022. The hardware and software specifications of the system are as follows:

11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz, 8.00 GB RAM, and Windows 11 Pro, 64-bit operating system.

The results of the proposed methods to the other compared methods are shown in Table I, in which the optimal tour length (cost) is recorded as reported in the TSPLIB standard library. As it can be seen from Table I, the proposed IAM-TSP+ outperforms the IAM-TSP on all TSPs, which is to be expected given that the latter is an input to the former and the former employs nine local permutations along the resultant route, giving it more chances to identify better solutions.
TABLE I. PERFORMANCE COMPARISON OF THE PROPOSED METHODS TO NINE RELATED METHODS ON DIFFERENT TSP Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Optimal</th>
<th>IAM-TSP</th>
<th>IAM-TSP+</th>
<th>CH</th>
<th>CNH</th>
<th>FI</th>
<th>NI</th>
<th>CI</th>
<th>AI</th>
<th>NN</th>
<th>RNN</th>
<th>2OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a280</td>
<td>2579</td>
<td>3051.084</td>
<td>2978.079</td>
<td>3335.9</td>
<td>3448</td>
<td>2953</td>
<td>3072.7</td>
<td>3110.7</td>
<td>2995.7</td>
<td>3369.7</td>
<td>3037.7</td>
<td>3018.1</td>
</tr>
<tr>
<td>att48</td>
<td>33523</td>
<td>3595.13404</td>
<td>3487.093</td>
<td>35618</td>
<td>35811.2</td>
<td>35267</td>
<td>37893.3</td>
<td>36391.9</td>
<td>35723.1</td>
<td>42112.7</td>
<td>39237</td>
<td>37458.4</td>
</tr>
<tr>
<td>berlin52</td>
<td>7542</td>
<td>8497.33404</td>
<td>8031.31761</td>
<td>9013.7</td>
<td>9013.7</td>
<td>8175</td>
<td>9097.7</td>
<td>9007.3</td>
<td>8346.4</td>
<td>9265.4</td>
<td>8182.2</td>
<td>8368.6</td>
</tr>
<tr>
<td>ch150</td>
<td>6428</td>
<td>7367.615815</td>
<td>7167.696739</td>
<td>7176.5</td>
<td>7148</td>
<td>8066.9</td>
<td>7988.7</td>
<td>7228.8</td>
<td>7734.2</td>
<td>7078.4</td>
<td>7379.4</td>
<td></td>
</tr>
<tr>
<td>ch130</td>
<td>6110</td>
<td>6664.37646</td>
<td>6584.10652</td>
<td>7038</td>
<td>7038.8</td>
<td>6855</td>
<td>7381.6</td>
<td>7164.9</td>
<td>6625.8</td>
<td>7747.4</td>
<td>7198.7</td>
<td>6755.5</td>
</tr>
<tr>
<td>kroA100</td>
<td>21282</td>
<td>23375.94197</td>
<td>22212.77837</td>
<td>22899.1</td>
<td>23310.1</td>
<td>22874</td>
<td>25957</td>
<td>25073.5</td>
<td>23270.7</td>
<td>27084.2</td>
<td>24699</td>
<td>24401.9</td>
</tr>
<tr>
<td>pr76</td>
<td>108159</td>
<td>115547.4825</td>
<td>113155.4313</td>
<td>115790</td>
<td>118877</td>
<td>117173</td>
<td>130029</td>
<td>126837</td>
<td>116098</td>
<td>145227</td>
<td>130921</td>
<td>118838.3</td>
</tr>
<tr>
<td>lin105</td>
<td>14379</td>
<td>16285.06763</td>
<td>15948.15549</td>
<td>15596.1</td>
<td>15730.7</td>
<td>15331</td>
<td>18287.9</td>
<td>17327.6</td>
<td>15802.6</td>
<td>18646</td>
<td>16939</td>
<td>16322.2</td>
</tr>
<tr>
<td>pcb442</td>
<td>50778</td>
<td>57860.0934</td>
<td>57458.28</td>
<td>66961.6</td>
<td>72425</td>
<td>57537.9</td>
<td>60667.9</td>
<td>59493.1</td>
<td>58001.5</td>
<td>64819.2</td>
<td>59975</td>
<td>57354.4</td>
</tr>
</tbody>
</table>

Furthermore, the proposed IAM-TSP+ outperforms not only IAM-TSP, but also, all methods compared on 5 TSPs, namely att48, berlin52, ch130, KroA100, pr76, followed by the FI method, which also performs well.

Aside from the FI, the proposed IAM-TSP outperforms many other methods without the need for additional improvement and achieves solutions closer to optimal in two instances: att48 and pr76. It is interesting to note that both of the proposed methods achieve performance that is very close to the optimal solution in some instances. In order to illustrate the different performances of both of the best performers, IAM-TSP+ and FI, Fig. 3 compares the resultant routes’ costs of both methods on several TSPs while Fig. 4, 5, and 6 provide comparisons of the proposed methods to the other nine related methods on small-scale TSP instances. As it can be seen from these figures, the proposed methods achieve better or comparable performance when compared to the other approximation methods that attempt to find the best possible TSP solution.

![Fig. 3. The routes’ cost results of the proposed IAM-TSP+ compared to that of the FI method.](image)

![Fig. 4. The routes’ cost results of the proposed methods compared to that of the other methods on Att48 TSP.](image)

![Fig. 5. The routes’ cost results of the proposed methods compared to that of the other methods on pcb442 TSP.](image)
The approximation ratio, which is the ratio of the method’s output to the optimal tour cost, quantifies how much the approximate solution differs from the optimal solution. The approximation ratios of all methods compared are presented in Table II. As it can be seen in the table, the proposed IAM-TSP+ delivers the best overall approximation, with an average of 1.09 overall TSPs, compared to the next competitor (FI), which has an average approximation of 1.10.

The proposed AIM-TSP also performs well, with an average approximation of 1.11, surpassing seven comparable methods. In addition to the approximation ratio, the error rate represents the percentage difference between the solution’s fitness value and the known optimal solution, and it is determined as follows [51]:

\[
\text{Error Rate} = \frac{\text{solution – optimal solution}}{\text{optimal solution}} \times 100\% \quad (1)
\]

The comparison results for the error rates are presented in Table III. As it can be seen in Table III, the proposed IAM-TSP+ delivers the minimum overall error rate, with an average of 8.51% overall TSPs, compared to the next competitor (FI), which has an average error rate of 9.50%. The proposed AIM-TSP also performs well, with an average error rate of 11.44%, surpassing seven comparable methods. Considering the costs, approximation ratios, and error rates of the resulting Routes, the proposed methods, specifically the IAM-TSP+, outperform all related methods aimed at solving the TSP in general.
V. CONCLUSION

In this paper, we presented two geometric-based approximation greedy algorithms for solving the classical TSP, one of which we call IAM-TSP which is proposed based on locating four extreme nodes/cities and then iterating to determine the best potential positions of each node/city. The other method is known as IAM-TSP+, and it is simply an improved version of the first one, by employing local constant permutations to improve the first method's result.

The experimental results of the proposed methods on nine TSP instances show that, when compared to nine recent related methods, the proposed methods (particularly the IAM-TSP+) provide promising solutions for the classical TSP. This is seen in the resulting routes’ cost effectiveness, approximation ratios, and error rates. The enhanced performance (of the IAM-TSP+) is due to the selection of the best local solution, which was accomplished by exploring all k-permutations and considering the best local solution after obtaining the initial solution from the pure IAM-TSP, where k = 5 in all experiments.

The proposed method’s limitations include the time and space complexity, which is rather high for the proposed IAM-TSP+, making the evaluation of the proposed methods on large TSP instances particularly difficult on restricted resource machines. As a result, parallel or distributed computation may facilitate the evaluation of proposed methods. In addition to employing the output route as an initial seed for the genetic algorithm [52, 53]. Our future research will focus on such issues.

REFERENCES


