

Solving k-city multiple travelling salesman using genetic algorithm

Alikapati Prakash^{1,2}, Uruturu Balakrishna³, Manogaran Thangaraj⁴, Thenepalle Jayanth Kumar⁴

¹Department of Mathematics, Jawaharlal Nehru Technological University Anantapur, Anantapuramu, India

²Department of Humanities and Sciences, Vemu Institute of Technology, Pakala, India

³Department of Science and Humanities, Sreenivasa Institute of Technology and Management Studies, Bangalore, India

⁴Department of Mathematics, Faculty of Engineering and Technology, JAIN (Deemed-To-Be University), Kanakapura, India

Article Info

Article history:

Received Jun 11, 2024

Revised Mar 27, 2025

Accepted Jun 8, 2025

Keywords:

Genetic algorithm

Multiple travelling salesman problem

Travelling salesman problem

TSPLIB

Zero-one integer linear programming problem

ABSTRACT

This paper addresses a novel variant of the classical multiple traveling salesman problem (MTSP) i.e. k-city multiple traveling salesman problem (k-MTSP). The problem can describe as follows. Let there are n cities, m salesman positioned at depot city and a predefined positive value k . The distance between each pair of cities is known. The objective of the k-MTSP is to determine a collection of m closed tours for salesman, which covers exactly k (including depot city) of n cities such that the total distance covered is minimum. The k-MTSP can be seen as a combination of both subset selection and permutation characteristics. From the through literature review, it is found that this study on k-MTSP is first of its kind to the best of author's knowledge. The paper introduces a zero-one integer linear programming (0-1 ILP) formulation alongside an efficient genetic algorithm (GA), designed to address k-MTSP. No comparative studies carried out due to the absence of existing studies on k-MTSP. However, the developed GA is tested over various benchmark test cases from TSPLIB and results are reported, which may potentially serve as basis for further comparative studies. Overall findings demonstrate that the GA consistently produces best solutions within reasonable computational times for relatively smaller and medium test cases, suggesting its robustness and effectiveness in tackling the k-MTSP. However, to enhance consistency and efficiency, particularly for larger datasets, further algorithm improvements are necessary.

This is an open access article under the CC BY-SA license.



Corresponding Author:

Uruturu Balakrishna

Department of Science and Humanities, Sreenivasa Institute of Technology and Management Studies

Dr. Visweswarai Road, Bangalore-Tirupathi, Bypass Rd, Murukambattu, Andhra Pradesh 517127, India

Email Id: balusitams.maths@gmail.com

1. INTRODUCTION

The traveling salesman problem (TSP) stands as a well-known combinatorial optimization problem within the domains of computer science and mathematics. Its core objective is to identify the optimal route that originates and concludes in the same city, covering all cities in a given list while covering the distances between each pair of cities. Expanding upon the classical TSP, the multiple traveling salesman problem (MTSP) introduces a variation wherein multiple salesmen are assigned the task of visiting a specified set of cities. The aim is to determine the most efficient routes for all salesmen, collectively minimizing the overall traversal distance, while ensuring each city is visited only once by a single salesman. Addressing the MTSP proves to be a more challenging compared to the conventional TSP due to the additional constraint of involving multiple salesmen. Similar to the TSP, the MTSP is categorized as NP-hard problem. This

classification implies that as the number of cities and salesmen increases, finding the most efficient solution becomes progressively more computationally demanding. Due to its practical significance, the classical MTSP and its variants has been extensively studied [1]–[8] and wide variety of solution methods have been developed.

Reviewing the existing studies on MTSP and its allied problems, the study [9] introduces a novel genetic algorithm (GA) chromosome and operators for the MTSP, demonstrating superior computational performance and smaller search space compared to prior methods through theoretical analysis and computational testing. A novel grouping GA is developed by [10] for the MTSP, which optimizes two objectives namely traversal distance and minimizing the maximum travel distance for any salesman. This approach shown superior performance compared to existing literature approaches on both objectives. A novel crossover method, termed the "two-part chromosome crossover" to address the MTSP through the application of a GA, aiming to achieve near-optimal solutions is developed [11]. The multiple depot MTSP, which involves an unlimited number of salespeople visiting a specific group of cities studied and developed polyhedral based branch-and-cut algorithm [12]. An efficient evolutionary algorithm that integrates two revised versions of the imperialist competitive algorithm and the Lin-Kernighan algorithm is developed for solving classical MTSP [13]. A novel variant of classical MTSP known as the colored TSP has been introduced [14], for which two hybrid algorithms namely hill-climbing based GA and simulated annealing-based GA are developed for achieving best quality solutions. Soylu [15] studied MTSP involving two separate objective functions: minimizing the longest tour length and the total length of all tours, by proposing a general variable neighborhood search. Subsequently, two metaheuristic techniques for the MTSP are proposed: one based on the artificial bee colony algorithm, and the other utilizing the invasive weed optimization algorithm [16]. The gravitational emulation local search algorithm [17] is developed to tackle the symmetric MTSP. This algorithm relies on the principles of local search, incorporating two fundamental physics parameters: velocity and gravity. It is also note that MTSP exhibits a strong connection with various optimization models, including the vehicle routing problem (VRP) [18], [19]. Xu *et al.* [20] introduced the two-phase heuristic algorithm for solving the MTSP, which integrates an improved version of the k-means algorithm to sort the cities to be visited based on their unique capacity constraints and spatial positions.

Given its significance as an optimization problem with practical applications, numerous researchers have utilized MTSP to address real-time scenarios. A new practical variation known as the open-close MTSP [21], which extends the classical MTSP. Over time, it has found relevance in various real-world situations. To cite few, wireless rechargeable sensor networks [22], natural disaster management [23], and optimal delivery distribution of LPG cylinders [24]. In addition to the cited works, researchers have also focused on developing hybrid algorithms for the MTSP and its allied problems to improve the solution quality [25]–[28]. Some of the latest studies on MTSP are as follows: Yang and Fan [29] implemented energy and resource consumption constraints to create the restricted multi-depot TSP. The consistent TSP searches for a minimum-cost collection of Hamiltonian paths described by [30], the firefly approach is presented to solve the single depot MTSP using a threshold technique [31]. A bi-objective MTSP with a load balancing constraint utilizing GA [32]. Veeresh *et al.* [33] introduce a meta-heuristic termed multi chromosome-based GA to solve open-closed MTSP.

Motivated by the above-mentioned works, the current research focuses on a novel variant called *k*-MTSP, and effective metaheuristic in the form of a GA. This GA incorporates complex mutation strategies, yielding optimal/near optimal results. According to the author's understanding, this GA represents the pioneering evolutionary technique applied to the *k*-MTSP. The subsequent sections of the paper are organized in the following manner. The second section will present the definition and formulation of the *k*-MTSP. Section 3 will provide a description of the GA and its operators. The computational findings will be showcased in section 4, while section 5 will end by summarizing the findings and discussing potential directions for future research.

2. MATHEMATICAL MODEL

The *k*-MTSP can be formally defined as follows: Let $G = (N, E)$ be an undirected weighted and connected graph, where $N = \{1, 2, \dots, n\}$ be the set of n cities/nodes (including depot city) and $E = \{(i, j) / (i, j) \in E, i \neq j\}$ be an edge/arc set. For each edge $(i, j) \in E$, a positive distance d_{ij} ($d_{ij} > 0, d_{ii} = \infty$ & $d_{ij} = d_{ji}$) is assigned, which indicate travel distance/cost from i^{th} city to j^{th} city. Let $K = \{1, 2, \dots, m\}$ be the set of m salesmen positioned at a depot city. Let x_{ij}^p be a binary variable, which takes the value 1 when salesman p traverses from city i to city j , and 0, otherwise. The cities other than the depot are known to be intervening cities. The *k*-MTSP aims to identify a set of m closed tours, encompassing k of the n cities, where each intervening city is visited by exactly one salesman with minimal distance. The formulation of the *k*-MTSP model is based on the following assumptions:

- There are n number of cities, out of which k cities are to be covered by m salesmen, all stationed at the depot city.
- All salesmen must start their routes from the depot city and conclude their tours there as well.
- The feasible solution comprises a set of m closed tours
- The allocation of cities to each salesman is dynamic, aiming to minimize the overall travel distance.
- Each city, except for the depot, must be visited precisely once by only one salesman.
- The number of cities visited by any salesman is constrained, with a minimum of 1 and a maximum of $k - m$ cities.

The nomenclature used in the model are given in Table 1 as follows:

Table 1. Nomenclature	
Notations	Description
$G = (N, E)$	An undirected weighted and connected graph
$N = \{1, 2, \dots, n\}$	Node set with n cities
$E = \{(i, j) / i, j \in V; i \neq j\}$	Edge set
$x_{ij}^p \in \{0, 1\}, \forall (i, j) \in E, p \in K$	A binary variable
$D = [d_{ij}]_{n \times n}$	A symmetric matrix representing distances between pairs of cities
$d_{ij}; (d_{ij} = d_{ji}; d_{ii} = \infty, d_{ij} > 0)$	Distance from i^{th} city to j^{th} city
k	A positive integer representing the total number of cities that all the salesmen need to cover (i.e. k out of n)
$y_i \in \{0, 1\}, i \in V$	A binary variable associated with visited cities

The mathematical model of k-MTSP is as follows:

$$\text{Minimize } Z = \sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij}^p, 1 \leq p \leq m \quad (1)$$

Subject to, the objective function (1) tells that minimization of total distance covered by m salesmen. The constraint set (2) and (3) confirms that exactly $k - 1$ (excluding depot city) of the n cities cover by m salesmen and a feasible solution must include $k + 1$ edges connecting the cities. Constraint sets (4) and (5) provides that m salesmen depart and returns the depot city. Constraint sets (6) and (7) aims that each city is being visited exactly once by only one salesman. Constraint (8) establishes the minimum and maximum number of cities that any salesman can visit. Constraint (9) is designed to prevent the formation of sub-tours within the solution. The constraint (10) indicates the binary variable x_{ij}^p , i.e. it takes 1 if p^{th} salesman travels from i^{th} city to j^{th} city and 0 otherwise. Finally, another binary variable y_i^p is introduced in the constraint (11), which takes 1 if p^{th} salesman visits i^{th} city and 0 otherwise.

$$\sum_{i=2}^n y_i^p = k - 1, p \in K \quad (2)$$

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij}^p = k + 1, i \neq j, p \in K \quad (3)$$

$$\sum_{j=2}^n x_{1j}^p = m, p \in K \quad (4)$$

$$\sum_{i=2}^n x_{i1}^p = m, p \in K \quad (5)$$

$$\sum_{i=1}^n x_{ij}^p = 1, i \neq j, \forall j \in N \setminus \{1\}, p \in K \quad (6)$$

$$\sum_{j=1}^n x_{ij}^p = 1, i \neq j, \forall i \in N \setminus \{1\}, p \in K \quad (7)$$

$$1 \leq \sum_{i=1}^n \sum_{j=1}^n x_{ij}^p \leq k - m, p \in K \quad (8)$$

$$+\text{Subtour elimination constraints} \quad (9)$$

$$x_{ij}^p \in \{0, 1\}, i \neq j, \forall i, j \in N, p \in K \quad (10)$$

$$y_i^p \in \{0, 1\}, \forall i \in N, p \in K \quad (11)$$

3. METHODOLOGY

The proposed GA is recognized as a commonly used metaheuristic within the domain of evolutionary computation studies, specifically designed for addressing problems involving the optimization of a combination [34]. This population-based approach continually refines the solution by enhancing its fitness value through iterative updates. Finding the best solution to the k -MTSP by using proposed algorithm include several crucial elements such as representation of chromosomes, population generation, fitness evaluation, selection, mutation and the general parameters of GA.

3.1. Chromosome representation

Finding optimal routes for multiple salesmen visiting each city once in the MTSP requires encoding solutions into chromosomes. There are primarily two methods: the single-chromosome [35] approach and the two-chromosome [36] approach. The initial method employs a chromosome whose length is $(n + m - 1)$, with n representing the total cities and m indicating the quantity of salesmen, to directly encode both cities visit order and assigned salesmen. The two-chromosome strategy involves utilizing a pair of chromosomes, each with a length of n . The first defines the universal city visit order, while the second assigns specific salesmen based on their corresponding position in the first chromosome. Emerging in the field of chromosome representation is a novel technique known as the two-part chromosome approach [9]. Every chromosome is divided into two segments: the initial segment is a sequence of cities, numbering $k - 1$, arranged in order from 2 up to n . The second part defines the route breakpoints for each of the m salesman within their assigned cities. The two-part chromosome for 10 city k -MTSP with 3 salesmen given in Figure 1. In this situation, the salesman 1 visits 3 cities $1 \rightarrow 7 \rightarrow 5 \rightarrow 8$, the salesman 2 visits 2 cities $1 \rightarrow 3 \rightarrow 9$ and the salesman 3 visits 2 cities $1 \rightarrow 2 \rightarrow 4$. The route plan for 3 salesmen covering 8 out of 10 cities given in Figure 2.

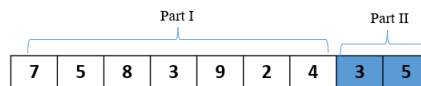


Figure 1. Two-part chromosome representation of 10 city k -MTSP with 3 salesmen

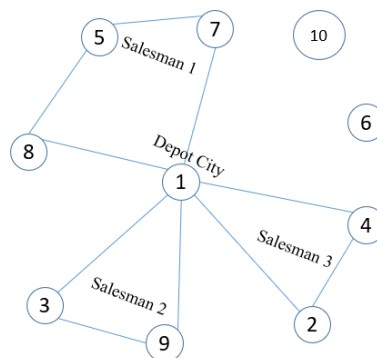


Figure 2. Route plan for 3 salesmen covering 8 out of 10 cities (Including depot city)

3.2. Initial population encoding

The success of a GA is greatly influenced by the quality of the initial population set, underscoring the significance of selecting the right encoding operator. In the realm of MTSP and its variations, studies indicate that permutation encoding stands out as the optimal choice for creating potent initial populations. This is evident in the widespread adoption of GA techniques employing permutation encoding in this context [20].

3.3. Fitness function

The fitness function is used to calculate the individual chromosomes of the population. The selection strategy in GAs relies on the importance of the fitness value, representing a crucial step. Specifically, a higher fitness value for a chromosome indicates an increased likelihood of being chosen for the next generation. In our study, the objective function is regarded as identical with the fitness function outlined in (1).

3.4. Selection operator

The selection operator significantly impacts the GA efficiency. The GA employs the conventional roulette wheel approach as its selection mechanism. This approach selects a chromosome for the breeding pool statistically, according to its fitness value from the population.

3.5. Mutation operator

Mutation is utilized in the GA to avoid convergence at local optima and to enhance the genetic variation among the population. This task utilizes a complex mutation operator that includes various operations: flip, swap, slide, as well as combinations like flip with modify breaks, swap with modify breaks, and slide with modify breaks. These mutation operators are employed to identify the shortest possible distance while also reducing the time required for computation. The flip operator is a mutation operation that reverses the order of a segment of the route between two insertion points. In Figure 3, the flip operator is applied to the best route among the three routes generated in each iteration of the loop. The swap operator in a GA is a mutation operator that exchanges the positions of two elements in a solution or individual. In Figure 4, the swap operation is applied to a route, where two cities are chosen, and their positions in the route are swapped. The slide operator is a mutation operator used in GA for finding the optimal distance. This operator involves moving a segment of the route to a new position within the same route which is given in Figure 5. The modify breaks operator in this context seems to be applied to the breaks or breakpoints in a route. In the context of a MTSP, breaks or breakpoints could represent specific locations where a salesman makes a stop or pauses during its route. Modifying breaks may involve changing the order or positions of these breaks which is given in Figure 6.

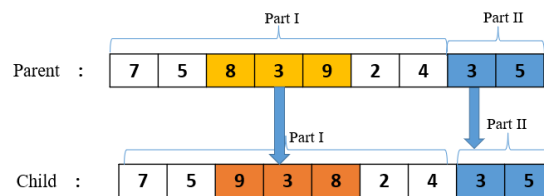


Figure 3. Flip operator

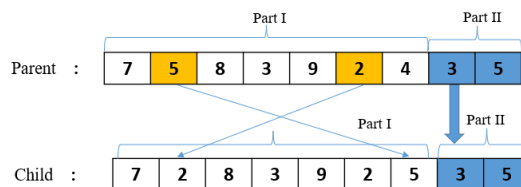


Figure 4. Swap operator

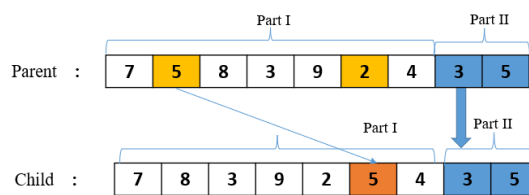


Figure 5. Slide operator

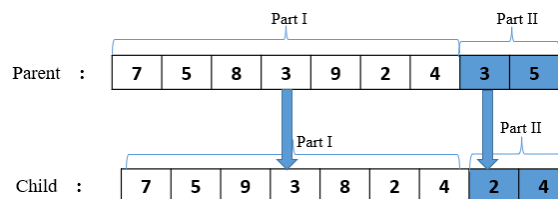


Figure 6. Modify breaks operator

The flip and modify breaks operator combine the flip operator to reverse a segment of the route with the modify breaks operator to introduce variability in the breaks. This creates a new solution by altering the order of cities and the breaks simultaneously, which is given in Figure 7. The swap and modify operator combine the swap operator with the modification of breaks. In Figure 8, it swaps the order of two cities in the route and simultaneously modifies the breaks associated with that route. Figure 9 generates a new solution by sliding a segment of the route to the right and then modifying the breaks randomly. The combination of these operations provides diversity in the generated solutions during the GA optimization process.

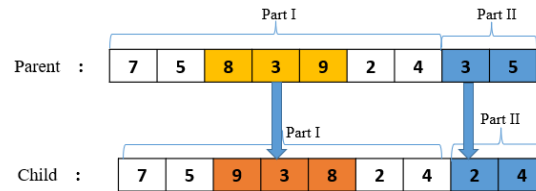


Figure 7. Flip and modify breaks operator

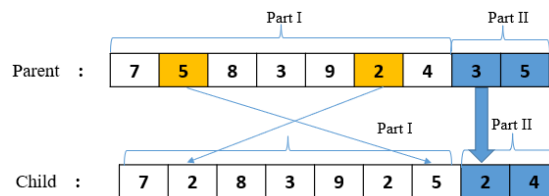


Figure 8. Swap and modify breaks operator

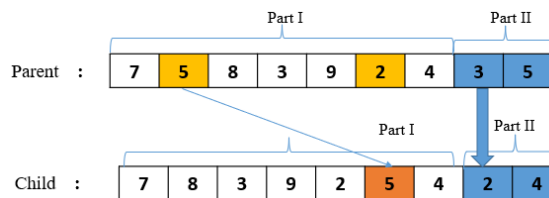


Figure 9. Slide and modify breaks operator

3.6. GA parameters

The effectiveness of the algorithm relies on the values of various parameters, including the size of the population, the rate of mutation probability, and the criteria for termination. In this study, the size of the population is set at 80, representing the number of chromosomes in a single generation. The study does not explore into the crossover operator, but it emphasizes the use of a sophisticated mutation operator to achieve its intended objectives.

4. RESULTS AND DISCUSSION

This section presents the computational results of the proposed GA. As there is no comparative study devoted for k -MTSP, diverse benchmark datasets sourced from TSPLIB (<http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>) were utilized to execute the algorithm. The performance of the algorithm was measured based on the best, worst, and average results for each test case, as well as the average CPU runtimes. The MATLAB R2023b was utilized to code the algorithm, which was executed on a personal computer featuring an Intel(R) Core (TM) i3-10110U CPU running at 2.10 GHz and equipped with 8 GB of RAM. The GA is carried out independently ten times for each test case, and the best, worst, and mean results are documented following every cycle. In total, 10 instances from TSPLIB have been examined. Each test case is run independently for 10 times. For every case, where each instance involves four different salesman

sizes, determine the best, worst and average results, and also record the average CPU runtimes. These test scenarios are characterized by Euclidean, two-dimensional symmetry and distinct node scales, ranging from 48 to 318 cities. The initial city in each instance is designated as the home city. Three distinct scenarios $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \left\lfloor \frac{n}{6} \right\rfloor\right)$ are contemplated for each of these 10 instances. Tables 2 to 4 present computational results for instances involving $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \left\lfloor \frac{n}{6} \right\rfloor\right)$. In each of these Tables 2 to 4, the initial column displays serial numbers, indicating 40 distinct numerical instances, the second column indicates name of the test instance alongside the associated count of cities, situated in the third column. The next two columns represent the quantity of salesman and the corresponding best, worst, and average results, denoted in size. The particular settings for the suggested algorithm are detailed in Table 2. The CPU runtimes for each case are provided in Tables 2 to 4, same is shown in Figure 10.

The results in Table 2 indicate significant variability in the algorithm's performance across different benchmark instances, particularly highlighting the impact of instance complexity on run times. Instances with more nodes, such as *bier127* and *gil262*, exhibit notably longer run times, suggesting challenges in handling larger datasets. While some instances, like *eil51*, show consistent performance with close run times across solutions, others display instability, as seen in *bier127*, where the gap between best and worst times is substantial. Interestingly, best solutions do not consistently improve with more solutions, reflecting potential inefficiencies. Overall, the findings suggest that while the algorithm performs well on smaller instances, its scalability and reliability on larger problems warrant further tuning and optimization to enhance efficiency and performance.

Table 2. Results of proposed algorithm on benchmark instances with $\left\lfloor \frac{n}{2} \right\rfloor$

SN	Instances	n	m	Solution			Avg. CPU run time (in seconds)
				Best	worst	Average	
1	att48	48	2	18,107	25,347	22,778.7	2.850
2			3	21,824	28,411	25,423.4	3.594
3			4	25,822	32,505	29,605.2	3.115
4			5	28,581	33,887	30,805.8	3.766
5	eil51	51	2	282	302	294.8	2.914
6			3	291	334	317.9	3.052
7			4	336	367	355.1	3.074
8			5	345	402	382.6	3.176
9	berlin52	52	2	4,210	5,716	5,022	3.345
10			3	4,299	5,733	5,122.2	3.247
11			4	5,001	6,223	5,697.3	3.515
12			5	5,143	6,737	5,781.8	3.658
13	st70	70	2	461	523	503	3.537
14			3	492	623	553.5	3.771
15			4	550	699	635	4.246
16			5	630	829	731.6	4.399
17	rat99	99	2	661	709	691.9	3.366
18			3	730	844	779	3.739
19			4	833	1,003	873.2	3.987
20			5	931	1,052	979.8	4.317
21	eil101	101	2	473	497	487	4.739
22			3	505	554	525.7	4.309
23			4	543	586	564.3	4.815
24			5	578	639	606	5.053
25	bier127	127	2	49,884	51,414	50,633	5.120
26			3	51,982	55,291	53,739.6	5.686
27			4	53,899	57,719	55,679.4	5.545
28			5	56,851	61,791	58,710	8.705
29	pr152	152	2	38,871	39,956	39,427.5	4.405
30			3	45,582	60,453	51,876.6	4.283
31			4	63,117	69,944	66,374.5	5.039
32			5	68,523	91,056	76,467	5.092
33	gil262	262	2	2,191	2,522	2,305.3	8.423
34			3	2,458	3,071	2,738.1	9.119
35			4	2,909	3,430	3,188.1	9.690
36			5	3,187	3,712	3,442.7	10.506
37	lin318	318	2	28,445	29,477	28,829	6.011
38			3	29,474	31,986	30,919.6	6.521
39			4	30,471	35,457	32,954.2	6.875
40			5	32,941	41,480	35,740.5	7.242

The results in Table 3 reveal notable trends in the algorithm's performance across various benchmark instances, highlighting both strengths and challenges. For smaller instances like att48 and eil51, the algorithm demonstrates consistent and efficient run times, with average times remaining relatively low, indicating effective optimization strategies. However, as the problem size increases, particularly in instances like bier127 and pr152, run times escalate significantly, highlighting the algorithm's struggle with larger datasets. The variability in best and worst run times, especially in bier127, suggests that certain instances may require tailored optimization techniques. Interestingly, some instances, such as gil262, show impressive best run times despite their size, indicating potential for improvement in algorithm efficiency. Overall, while the algorithm performs well on smaller instances, its scalability and stability across larger problems call for further refinement to enhance overall performance.

Table 3. Results of proposed algorithm on benchmark instances with $\left\lfloor \frac{n}{4} \right\rfloor$

SN	Instances	n	m	Solution			Avg. CPU run time
				Best	worst	Average	(in seconds)
1	att48	48	2	13,475	16,829	15,121.5	2.217
2			3	14,969	22,509	18,154.1	2.121
3			4	15,624	24,134	20,280.6	2.164
4			5	18,766	27,325	24,550.3	2.164
5	eil51	51	2	165	242	196.5	2.161
6			3	186	235	213.3	2.396
7			4	217	262	242.2	2.534
8			5	239	303	272.4	2.484
9	berlin52	52	2	2,111	3,435	2,871.4	2.161
10			3	2,122	3,450	2,999.7	2.432
11			4	2,589	4,431	3,251.7	2.359
12			5	2,825	4,162	3,543.7	2.258
13	st70	70	2	304	372	345.9	2.509
14			3	366	462	409.2	2.574
15			4	371	494	439	2.894
16			5	453	662	519.1	2.838
17	rat99	99	2	357	361	357.6	2.584
18			3	404	429	409.2	2.892
19			4	459	526	474.4	3.194
20			5	532	623	578.7	3.094
21	eil101	101	2	238	333	297.1	2.956
22			3	283	364	338.5	2.932
23			4	315	421	385.1	2.967
24			5	358	449	406.5	3.147
25	bier127	127	2	19,218	19,453	19,312.4	4.005
26			3	20,304	21,338	20,747.6	5.074
27			4	22,087	23,450	22,661.6	4.750
28			5	23,871	25,946	25,020	5.498
29	pr152	152	2	21,516	21,516	21,516	2.948
30			3	22,993	31,983	27,076	3.739
31			4	26,387	40,744	33,000.5	4.371
32			5	31,494	47,052	38,439.2	4.283
33	gil262	262	2	1,386	1,435	1,406.9	5.302
34			3	1,550	1,895	1,673.7	5.589
35			4	1,931	2,229	2,065.1	6.214
36			5	1,843	2,505	2,293.8	6.584
37	lin318	318	2	11,466	12,429	12,002.9	4.566
38			3	12,075	14,593	13,317.6	4.761
39			4	13,009	16,182	14,436.6	4.945
40			5	15,484	17,537	16,496.9	5.212

Finally, the results in Table 4 demonstrate important insights about the algorithm's performance. It performs well on smaller instances like att48 and eil51, with consistently low average run times. However, larger instances, such as bier127 and pr152, show a significant increase in run times, indicating scalability issues. Variability in best and worst run times, especially in bier127, suggests that the algorithm may struggle with certain configurations. In contrast, rat99 demonstrates stability with equal best and worst times across solutions. Overall, the algorithm is effective for smaller problems in this context, improvements are needed to enhance performance for larger datasets and ensure more results that are consistent.

Table 4. Results of proposed algorithm on benchmark instances with $\lfloor \frac{n}{6} \rfloor$

SN	Instances	n	m	Solution			Avg. CPU run time (in seconds)
				Best	worst	Average	
1	att48	48	2	7,841	11,818	10,067.7	1.849
2			3	12,236	16,556	14,677.8	1.977
3			4	11,033	17,862	13,286.5	1.821
4			5	13,022	18,671	16,335.3	1.845
5	eil51	51	2	124	163	144.4	2.113
6			3	144	185	163.7	2.260
7			4	155	224	194.3	2.369
8			5	176	242	200.5	2.362
9	berlin52	52	2	1,363	2,373	2,122.9	1.893
10			3	1,796	2,477	2,188	1.976
11			4	2,030	2,848	2,403.6	2.258
12			5	1,956	2,680	2,241.8	1.921
13	st70	70	2	223	348	282.8	2.186
14			3	250	380	325.2	2.311
15			4	328	426	376.2	2.261
16			5	385	552	481.7	2.301
17	rat99	99	2	257	257	257	2.211
18			3	305	305	305	2.499
19			4	366	376	367	2.522
20			5	449	449	449	2.520
21	eil101	101	2	203	245	217.1	2.309
22			3	202	269	244.4	2.776
23			4	253	314	282.6	2.887
24			5	273	332	309.4	2.809
25	bier127	127	2	12,270	12,405	12,293.1	4.075
26			3	13,329	13,885	13,544.7	3.007
27			4	14,815	15,547	15,171.6	3.894
28			5	16,513	17,689	16,903.6	3.359
29	pr152	152	2	21,909	21,909	21,909	2.432
30			3	27,054	32,856	2,8981.1	3.296
31			4	34,512	41,890	38,073.4	3.618
32			5	44,688	54,987	48,811.8	3.383
33	gil262	262	2	1,147	1,329	1,231.4	4.104
34			3	1,277	1,494	1,375	4.243
35			4	1,480	1,808	1,685.6	4.632
36			5	1,796	2,243	1,968.3	4.843
37	lin318	318	2	7,645	8,142	7,841	3.938
38			3	8,445	9,418	9,108.6	3.969
39			4	9,379	10,909	10,057.1	4.148
40			5	10,633	12,922	11,901.2	4.284

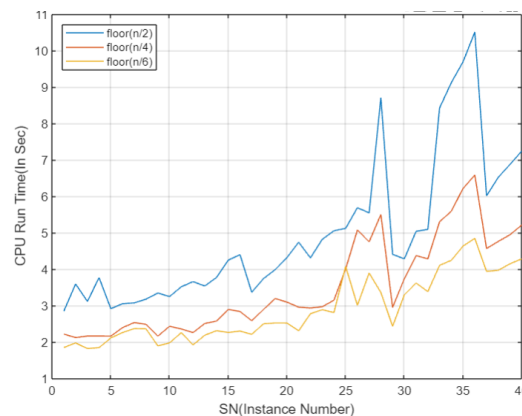


Figure 10. Average CPU run times of all instances

Overall, the results from Table 2 reveal consistent performance on smaller datasets like eil51, where average runtimes remain stable. In contrast, Table 3 shows that instances like bier127 and pr152 experience higher variability, which may be due to the complexity of these larger problem sizes. The increased runtime for these instances indicates challenges in scalability for the GA algorithm. The overall trend indicates that the algorithm performs well with smaller datasets, as shown by the consistent results for instances like eil51 and att48 in Table 2. However, as problem size increases shown in Tables 3 to 4, performance variability

becomes more evident, particularly in larger instances like bier127. This suggests that the algorithm needs further optimization for handling scalability issues effectively. In conclusion, while the proposed GA algorithm performs efficiently on smaller datasets, it faces challenges with scalability as the problem size increases. The variability in results across larger instances suggests that the algorithm may benefit from further optimization, such as hybridization with other techniques or parallelization for larger datasets [28], [37]. Future work could focus on improving the algorithm's robustness and efficiency to handle complex, large-scale k-MTSP instances more effectively.

5. CONCLUSION

In this study, we addressed a unique variant of classical MTSP, specifically the *k*-MTSP, utilized widely in outsourcing, transportation and logistics distribution. The aim of this problem is to find a set of complete tours for *m* salesman, covering precisely *k* out of the *n* cities, with the aim of minimizing the overall traversal distance or cost. According to the author's knowledge, this is the first GA developed for the *k*-MTSP. Given the absence of *k*-MTSP studies, no comparative studies are carried out. However, various benchmark test instances from the TSPLIB have been used to evaluate the effectiveness of the GA. The computational results of the proposed algorithm exhibit significant potential in attaining optimal/near optimal results for the *k*-MTSP. Being the first evolutionary algorithm for the *k*-MTSP, our proposed GA approach will serve as a reference point for subsequent research on the *k*-MTSP. Overall, the algorithm effectively finds best solutions, but enhancements are needed to improve consistency and efficiency, especially for larger datasets. For future work, we recommend exploring hybrid algorithms that combine GAs with techniques like simulated annealing, ant colony optimization, and machine learning methods, which could help address the challenges posed by large datasets. Additionally, incorporating parallel and distributed computing strategies could improve scalability. Other possible enhancements to the k-MTSP model include the integration of elements such as time windows, multiple depots, and other real-world variations to increase the model's applicability in practical scenarios.

ACKNOWLEDGMENTS

We would like to thank to our guide for his unwavering guidance, invaluable insights, and encouragement throughout the research process.

FUNDING INFORMATION

No funding is raised for this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Alikapati Prakash	✓	✓			✓	✓			✓					
Uruturu Balakrishna	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓		
Manogaran Thangaraj	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			
Thenepalle Jayanth Kumar	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		

C : C onceptualization	I : I nterpretation	Vi : V isualization
M : M ethodology	R : R esources	Su : S upervision
So : S oftware	D : D ata Curation	P : P roject administration
Va : V alidation	O : Writing - O riginal Draft	Fu : F unding acquisition
Fo : F ormal analysis	E : Writing - Review & E ditng	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

This research did not involve human participants or animals. Therefore, ethical approval was not required.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.




REFERENCES

- [1] G. Laporte and Y. Nobert, "A cutting planes algorithm for the m-salesmen problem," *The Journal of the Operational Research Society*, vol. 31, no. 11, 1980, doi: 10.2307/2581282.
- [2] A. I. Ali and J. L. Kennington, "The asymmetric M-travelling salesmen problem: a duality based branch-and-bound algorithm," *Discrete Applied Mathematics*, vol. 13, no. 2–3, pp. 259–276, 1986, doi: 10.1016/0166-218X(86)90087-9.
- [3] B. Gavish and K. Srikanth, "Optimal solution method for large-scale multiple traveling salesmen problems," *Operations Research*, vol. 34, no. 5, pp. 698–717, 1986, doi: 10.1287/opre.34.5.698.
- [4] K. C. Gilbert and R. B. Hofstra, "A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem," *Decision Sciences*, vol. 23, no. 1, pp. 250–259, 1992, doi: 10.1111/j.1540-5915.1992.tb00387.x.
- [5] J.-Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 63, no. 3, pp. 337–370, Jun. 1996, doi: 10.1007/BF02125403.
- [6] S. Somhom, A. Modares, and T. Enkawa, "Competition-based neural network for the multiple travelling salesmen problem with minmax objective," *Computers and Operations Research*, vol. 26, no. 4, pp. 395–407, 1999, doi: 10.1016/S0305-0548(98)00069-0.
- [7] V. Bhavani and M. S. Murthy, "Truncated M-travelling salesmen problem," *Opsearch*, vol. 43, no. 2, pp. 152–177, 2006, doi: 10.1007/bf03398771.
- [8] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006, doi: 10.1016/j.omega.2004.10.004.
- [9] A. E. Carter and C. T. Ragsdale, "A new approach to solving the multiple traveling salesperson problem using genetic algorithms," *European Journal of Operational Research*, vol. 175, no. 1, pp. 246–257, 2006, doi: 10.1016/j.ejor.2005.04.027.
- [10] A. Singh and A. S. Baghel, "A new grouping genetic algorithm approach to the multiple traveling salesperson problem," *Soft Computing*, vol. 13, no. 1, pp. 95–101, 2009, doi: 10.1007/s00500-008-0312-1.
- [11] S. Yuan, B. Skinner, S. Huang, and D. Liu, "A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms," *European Journal of Operational Research*, vol. 228, no. 1, pp. 72–82, 2013, doi: 10.1016/j.ejor.2013.01.043.
- [12] E. Benavent and A. Martínez, "Multi-depot multiple TSP: a polyhedral study and computational results," *Annals of Operations Research*, vol. 207, no. 1, pp. 7–25, 2013, doi: 10.1007/s10479-011-1024-y.
- [13] H. Larki and M. Yousefikhoshbakht, "Solving the multiple traveling salesman problem by a novel meta-heuristic algorithm," *Journal of Optimization in Industrial Engineering*, vol. 7, no. 16, pp. 55–63, 2014.
- [14] J. Li, M. C. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2390–2401, 2015, doi: 10.1109/TCYB.2014.2371918.
- [15] B. Soyly, "A general variable neighborhood search heuristic for multiple traveling salesmen problem," *Computers and Industrial Engineering*, vol. 90, pp. 390–401, 2015, doi: 10.1016/j.cie.2015.10.010.
- [16] P. Venkatesh and A. Singh, "Two metaheuristic approaches for the multiple traveling salesperson problem," *Applied Soft Computing*, vol. 26, pp. 74–89, 2015, doi: 10.1016/j.asoc.2014.09.029.
- [17] A. S. Rostami, F. Mohanna, H. Keshavarz, and A. A. R. Hosseinabadi, "Solving multiple traveling salesman problem using the gravitational emulation local search algorithm," *Applied Mathematics and Information Sciences*, vol. 9, no. 2, pp. 699–709, 2015, doi: 10.12785/amis/090218.
- [18] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuysse, "The vehicle routing problem: state of the art classification and review," *Computers and Industrial Engineering*, vol. 99, pp. 300–313, 2016, doi: 10.1016/j.cie.2015.12.007.
- [19] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective UAV path planning for search and rescue," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 5569–5574, doi: 10.1109/ICRA.2017.7989656.
- [20] X. Xu, H. Yuan, M. Liptrott, and M. Trovati, "Two phase heuristic algorithm for the multiple-travelling salesman problem," *Soft Computing*, vol. 22, no. 19, pp. 6567–6581, 2018, doi: 10.1007/s00500-017-2705-5.
- [21] J. K. Thenepalle and P. Singamsetty, "An open close multiple travelling salesman problem with single depot," *Decision Science Letters*, vol. 8, no. 2, pp. 121–136, 2019, doi: 10.5267/j.dsl.2018.8.002.
- [22] Z. Wei *et al.*, "The path planning scheme for joint charging and data collection in WRSNs: a multi-objective optimization method," *Journal of Network and Computer Applications*, vol. 156, 2020, doi: 10.1016/j.jnca.2020.102565.
- [23] O. Cheikhrouhou, A. Koubaa, and A. Zarrad, "A cloud based disaster management system," *Journal of Sensor and Actuator Networks*, vol. 9, no. 1, 2020, doi: 10.3390/jsan9010006.
- [24] P. Singamsetty and J. K. Thenepalle, "Designing optimal route for the distribution chain of a rural LPG delivery system," *International Journal of Industrial Engineering Computations*, vol. 12, no. 2, pp. 221–234, 2021, doi: 10.5267/j.ijec.2020.11.001.
- [25] C. Jiang, Z. Wan, and Z. Peng, "A new efficient hybrid algorithm for large scale multiple traveling salesman problems," *Expert Systems with Applications*, vol. 139, 2020, doi: 10.1016/j.eswa.2019.112867.
- [26] P. Singamsetty, J. K. Thenepalle, and B. Uruturu, "Solving open travelling salesman subset-tour problem through a hybrid genetic algorithm," *Journal of Project Management*, vol. 6, no. 4, pp. 209–222, 2021, doi: 10.5267/j.jpm.2021.5.002.
- [27] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: applications, approaches and taxonomy," *Computer Science Review*, vol. 40, 2021, doi: 10.1016/j.cosrev.2021.100369.
- [28] S. Mahmoudinazlou and C. Kwon, "A hybrid genetic algorithm for the min-max multiple traveling salesman problem," *Computers and Operations Research*, vol. 162, 2024, doi: 10.1016/j.cor.2023.106455.
- [29] R. Yang and C. Fan, "A hierarchical framework for solving the constrained multiple depot traveling salesman problem," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5536–5543, 2024, doi: 10.1109/LRA.2024.3389817.
- [30] D. Díaz-Ríos and J. J. Salazar-González, "Mathematical formulations for consistent travelling salesman problems," *European Journal of Operational Research*, vol. 313, no. 2, pp. 465–477, 2024, doi: 10.1016/j.ejor.2023.08.021.




- [31] R. Nand, K. Chaudhary, and B. Sharma, "Single depot multiple travelling salesman problem solved with preference-based stepping ahead firefly algorithm," *IEEE Access*, vol. 12, pp. 26655–26666, 2024, doi: 10.1109/ACCESS.2024.3366183.
- [32] S. Lingamathan and P. Singamsetty, "Genetic algorithm to the bi-objective multiple travelling salesman problem," *Alexandria Engineering Journal*, vol. 90, pp. 98–111, 2024, doi: 10.1016/j.aej.2024.01.048.
- [33] M. Veeresh, T. J. Kumar, and M. Thangaraj, "Solving the single depot open close multiple travelling salesman problem through a multi-chromosome based genetic algorithm," *Decision Science Letters*, vol. 13, no. 2, pp. 401–414, 2024, doi: 10.5267/j.dsl.2024.1.006.
- [34] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Choice Reviews Online*, vol. 27, no. 2, pp. 27-0936-27-0936, 1989, doi: 10.5860/choice.27-0936.
- [35] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron and Steel Complex," *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000, doi: 10.1016/S0377-2217(99)00380-X.
- [36] Y. B. Park, "A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines," *International Journal of Production Economics*, vol. 73, no. 2, pp. 175–188, 2001, doi: 10.1016/S0925-5273(00)00174-2.
- [37] M. Tong, Z. Peng, and Q. Wang, "A hybrid artificial bee colony algorithm with high robustness for the multiple traveling salesman problem with multiple depots," *Expert Systems with Applications*, vol. 260, 2025, doi: 10.1016/j.eswa.2024.125446.

BIOGRAPHIES OF AUTHORS






Alikapati Prakash    holds a Bachelor of Science (B.Sc.) and Master of Science (M.Sc.) in Mathematics from S.V University, Tirupathi and pursuing doctoral degree in Operations Research in JNTUA, Ananthapuram, Andhra Pradesh. He is currently working as a Professor at Department of Humanities and Sciences, Vemu Institute of Technology, Chittoor, India. His research includes meta-heuristics and combinatorial optimization models. He can be contacted at email: alikapati.prakash@gmail.com.






Uruturu Balakrishna    holds a Bachelor of Science (B.Sc.) in Mathematics, Master of Science (M.Sc.) in Mathematics, Master of Philosophy (M.Phil.) in Operations Research, Ph.D. in Operations Research, besides several professional certificates and skills. He is currently working as Professor and Head in the Department of Science and Humanities in Sreenivasa Institute of Technology and Management Studies, Chittoor, India. He published more 25 papers in reputed indexed various international journals. His research areas of interest include combinatorial optimization problems, exact algorithms, routing and scheduling models. He can be contacted at email: balusitams.maths@gmail.com.



Manogaran Thangaraj    holds a Bachelor of Science (B.Sc.) in Mathematics, Master of Science (M.Sc.) in Mathematics, Master of Philosophy (M.Phil.) in Mathematics, Ph.D. in Operations Research, besides several professional certificates and skills. He is currently working with the Department of Mathematics, School of Computer Science and Engineering at JAIN (Deemed-To-Be-University), Bangalore, Karnataka, India. He is a member of the International Association of Engineers (IAENG). His research areas of interest include combinatorial optimization problems, scheduling problem and queueing theory. He can be contacted at email: mthangaraj51@gmail.com or m.thangaraj@jainuniversity.ac.in.



Jayanth Kumar Thenepalle    holds a Bachelor of Science (B.Sc.) in Statistics, Master of Science (M.Sc.) in Applied Mathematics, Ph.D. in Operations Research, besides several professional certificates and skills. He is currently working with the Department of Mathematics, School of Computer Science and Engineering at JAIN (Deemed-To-Be-University), Bangalore, Karnataka, India. He is a member of the International Association of Engineers (IAENG). His research areas of interest include combinatorial optimization problems, genetic algorithms. He can be contacted at email: jayanth.maths@gmail.com or jayanth@jainuniversity.ac.in.