Decision Science Letters 13 (2024) 401–414

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Solving the single depot open close multiple travelling salesman problem through a multi-chromosome based genetic algorithm

M. Veeresh[a], T. Jayanth Kumar[a*] and M. Thangaraj[a]

[a]*Department of Mathematics, Faculty of Engineering and Technology, JAIN (Deemed-To-Be University), Kanakapura, Bangalore-562112, Karnataka, India*

C H R O N I C L E

Article history:
Received: October 10, 2023
Received in revised format: November 20, 2023
Accepted: January 12, 2024
Available online: January 12, 2024

Keywords:
Open close multiple travelling salesman problem
Meta-heuristic
Genetic Algorithm
TSPLIB

A B S T R A C T

The multiple travelling salesman problem (MTSP) extends the classical travelling salesman problem (TSP) by involving multiple salesman in the solution. MTSP has found widespread applications in various domains, such as transportation, robotics, and networking. Despite extensive research on MTSP and its variants, there has been limited attention given to the open close multiple travelling salesman problem (OCMTSP) and its variants in the literature. To the best of the author's knowledge, only one study has addressed OCMTSP, introducing an exact algorithm designed for optimal solutions. However, the efficiency of this existing algorithm diminishes for larger instances due to computational complexity. Therefore, there is a crucial need for a high-level metaheuristic to provide optimal/best solutions within a reasonable timeframe. Addressing this gap, this study proposes a first meta-heuristic called multi-chromosome-based Genetic Algorithm (GA) for solving OCMTSP. The effectiveness of the developed algorithm is demonstrated through a comparative study on distinct asymmetric benchmark instances sourced from the TSPLIB dataset. Additionally, results from comprehensive experiments conducted on 90 OCMTSP symmetric instances, generated from the renowned TSPLIB benchmark dataset, highlight the efficiency of the proposed GA in addressing the OCMTSP. Notably, the proposed multi-chromosome-based GA stands out as the top-performing approach in terms of overall performance. Further, solutions to symmetric TSPLIB benchmark instances are also reported, which will be used as a basis for future studies.

1. Introduction

The Multiple Traveling Salesman Problem (MTSP) represents an extension of the classical Traveling Salesman Problem (TSP). In MTSP, the challenge involves coordinating multiple salesman to efficiently cover a specified set of cities, ensuring each city is visited exactly once, and the salesman collectively return to the starting point, aiming to minimize the total traversal cost or distance. The MTSP is closely connected with diverse optimization problems such as vehicle routing problem (VRP) (Braekers *et al.,* 2016) and task assignment problem (TAP) (Öncan, 2007). Indeed, MTSP can be viewed as a relaxed version the VRP, assuming unlimited vehicle capacity and customers with unit demands. While MTSP shares similarities with the TSP and the TAP, it imposes restrictions on multiple visits to the same city and the formation of sub-tours. Consequently, a solution to MTSP can be effectively applied to tackle the challenges posed by VRP or TAP.

The MTSP has undergone extensive research and has found applications in diverse fields, including transportation, robotics, and networking. In different practical scenarios, the "salesman" in MTSP can represent entities such as trucks, robots, or drones. The cities to be covered by the salesman correspond to customers in transportation and logistics distribution, critical sites in disaster management, targets in strategic war operations, sensor nodes in wireless sensor networks, and victims in emergency missions (Cheikhrouhou & Khoufi, 2021). Due to its broad applicability, MTSP has garnered significant attention from the research community. Researchers have explored various variants of the problem, such as MTSP with multiple depots (Malik *et al*., 2007), solid MTSP (Changdar *et al.,* 2017), MTSP with precedence constraints (Sarin *et al.,*

* Corresponding author.
E-mail address: jayanth.maths@gmail.com (T. Jayanth Kumar)

doi: 10.5267/dsl.2024.1.006

2014), MTSP with time windows (Kara & Bektas, 2006), Open close MTSP (Thenepalle & Singamsetty, 2019) and Open MTSP with load balancing (Thenepalle & Singamsetty, 2021) etc. Due to NP-hardness, no known algorithm is presented that works within polynomial time for solving the MTSP and its variants (Garey & Johnson, 1979). MTSP, being a generalized form of TSP, allows solution techniques developed for TSP to be applicable to MTSP and its variations. These solution approaches can broadly be categorized into two types: heuristics or metaheuristics, which provide solutions that are optimal or near-optimal without guaranteeing their quality, and exact algorithms, which assure optimal solutions. As the problem size grows, achieving an optimal solution becomes increasingly challenging and computationally expensive. Consequently, the application of heuristics or metaheuristics becomes essential for solving practical MTSP models within reasonable time constraints.

Concerning the solution techniques, some of the heuristic/metaheuristics techniques devoted for MTSP by implementing biological characteristics such as Evolutionary approach (Bao *et al.,* 2021; Sofge *et al.,* 2002), Particle swarm optimization (PSO) (Yan *et al.,* 2012), Ant colony optimization (ACO) (Changdar, 2023; Wang *et al.,* 2020; Yousefikhoshbakht, 2013), Hybrid algorithm (Jiang *et al.,* 2020), Genetic algorithm (GA) (Gomes *et al.,* 2021; Király & Abonyi, 2011; Lou *et al.,* 2021; Singamsetty & Thenepalle, 2021), Learning-based metaheuristic approach (Belhor *et al.,* 2023), Efficient routing optimization with discrete penguins search algorithm (Mzili & Riffi, 2023) etc. Various heuristic and metaheuristic approaches have been proposed for the Multiple Traveling Salesman Problem (MTSP) and its variations, each carrying its unique strengths and challenges. For instance, Particle Swarm Optimization (PSO) is known to be prone to local optima, while Ant Colony Optimization (ACO) is computationally demanding, characterized by a slower convergence rate. On the other hand, Genetic Algorithms (GA) tend to exhibit early convergence and can be heavily influenced by the initial population. Despite these complications associated with GA, literature shows to its effectiveness, as it is widely employed for successfully addressing the MTSP and its diverse variants (Xu *et al.,* 2018).

Examining the evolution of Genetic Algorithms (GA), Bagley (1967) was a pioneer in introducing the concept, followed by Holland, who conducted a scientific exploration into the mechanism of the survival of the fittest in 1975. Subsequently, the literature has witnessed significant progress in the development of GA, particularly in its application to addressing the MTSP and its variants, which can be seen in following works (Al-Omeer & Ahmed, 2019; Alves & Lopes, 2015; Brown *et al.,* 2007; Carter & Ragsdale, 2006; Harrath *et al.,* 2019; Kaliaperumal *et al.,* 2015; Király & Abonyi, 2011; Shuai *et al.,* 2019; Tang *et al.,* 2000; Thenepalle & Singamsetty, 2021; Xu *et al.,* 2018; and Yuan *et al.,* 2013). The cited studies motivate us to develop an efficient GA for solving the OCMTSP, aiming to find optimal solutions within a short timeframe. Table 1 summarizes the various instances of MTSP and its variants addressed through different solution methodologies.

The MTSP can be categorized into two main types based on the nature of the salesman routes: open MTSP and open-close MTSP. In the open MTSP, all salesman initiate their routes from the depot city and are not obligated to return to the starting city after completing the assigned cities. On the other hand, the open-close MTSP (OCMTSP) involves salesman starting from the depot city, visiting a specified number of cities, with only certain salesman required to return to the depot city. The goal is to minimize the overall distance or cost covered in the routing process. This model finds practical applications in transportation and logistics distribution, particularly in scenarios where vehicle operations are outsourced.

For instance, consider Fig. 1, which illustrates two variants of the classical MTSP: open MTSP and open-close MTSP. In this representation, numbered circles denote cities to be visited by the salesman, with city 0 serving as the depot city where all salesman commence their routes. Fig. 1 depicts a scenario of open MTSP involving 3 salesman and 10 cities, including the depot city. As shown, salesman 1 covers 3 cities, salesman 2 visits 3 cities, and salesman 3 covers 3 cities. Similarly, Fig. 2 illustrates a scenario of OCMTSP involving 3 salesman (1 internal and 2 external) and 10 cities, including the depot city. As shown, salesman 1 covers 3 cities and returned to depot city, salesman 2 just visits 3 cities, and salesman 3 covers 3 cities. The varying routes and the decision of which salesman return to the depot city contribute to minimizing the overall distance or cost incurred in the routing process. This flexibility in route planning is particularly beneficial in logistics and transportation optimization.

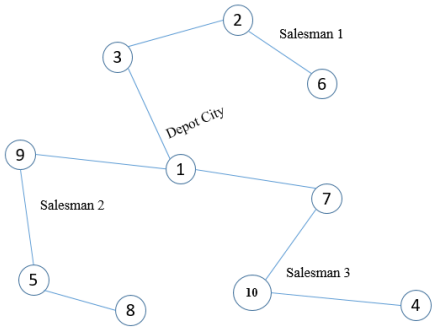

Fig. 1. An arbitrary open MTSP solution with 10 cities and 3 salesman

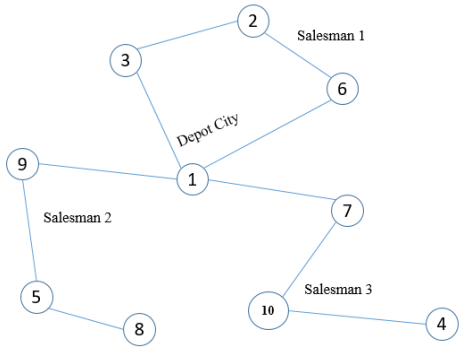

Fig. 2. An arbitrary open close MTSP solution with 10 cities and 3 salesman

Upon reviewing the existing literature, it becomes evident that, with the exception of the OCMTSP model, the other models mentioned have received significant attention. To the best of the author's knowledge, the only investigation into OCMTSP, addressed by an exact lexi-search algorithm (Thenepalle & Singamsetty, 2019), stands as the only study on this specific topic in the literature. However, the existing exact algorithm for OCMTSP demonstrates effectiveness primarily for smaller instances but proves computationally expensive when applied to higher dimensions. Recognizing this limitation, there is a pressing need to devise high-level, efficient metaheuristic approaches capable of producing optimal solutions within practical time constraints. To address this gap, the present study aims to introduce a multi-chromosome-based Genetic Algorithm (GA) for solving OCMTSP. The performance of this proposed approach is thoroughly analyzed through computational experiments, aiming to provide insights into its performance.

The rest of the paper is organized as follows: Section 2 describes a detailed description of the problem statement and its formulation. The preliminary concepts of GA are presented in Section 3. Extensive computational results are reported in Section 4. Finally, conclusions are given in section 5.

2. Problem Description and Mathematical Model

2.1 Problem Statement

The OCMTSP is formally stated as follows: Let $G=(N,E)$ be a directed connected weighted graph in which $N=\{1,2,...,n\}$ represents the set of n cities/nodes (together with depot city) and E denote an edge set. Let $d_{ij}(d_{ij}\neq d_{ji})$ be an asymmetric travel distance from i^{th} city to j^{th} city that is assigned to each edge $(i,j)\in E$. Let $K=\{1,2,...,m\}$ be the set of m (where $m=p+q;\ m\leq n$) salesman located at a depot city (say α, $\alpha\in N$), of which p salesman constitutes for closed paths and q salesman establishes open tours. If the salesman visits j^{th} city from i^{th} city then the decision variable x_{ij} assumes 1, and $x_{ij}=0$, otherwise. The OCMTSP objective is to find a solution involving p closed paths and q open paths, such that each city is to be covered by just one salesman and the total distance covered by m salesman is minimized. Note that, the lower and upper bound on the number of cities visited by any salesman is 1 and $n-m+1$.

2.2 Assumptions

The below assumptions are used to formulate the OCMTSP:

a. There are n cities to be covered by m salesman located at the depot city, of which a predefined p salesman are intended for closed paths and q salesman are employed for open paths.
b. All the salesman is asked to begin at the depot city and only p salesman are required to come back to the depot city, whereas the other q salesman is not necessary to return.
c. The values $m,p,\&q$ are predefined.
d. The cities assigned dynamically to each salesman in order to minimize the overall traversal distance.
e. Each salesman has to cover atleast 1 city and at most $n-m+1$ cities.

2.3 Mathematical Model

The OCMTSP can be expressed as a zero-one integer linear programming (0-1 ILP) as follows: In this article, a salesman objective (distance) is established that generally assures that the distance found is reduced. The mathematical model formulated by Thenepalle & Singamsetty, (2019) is considered without any modifications in the present study

a. This objective is to minimize overall traversal distance

The objective is to minimize overall traversal distance. This objective function promises that the distance involved in covering all the cities is minimized. It can be shown as:

$$\min\ Z=\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}\,x_{ij} \tag{1}$$

b. Constraints

$$\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ij}=m+n-q-1 \tag{2}$$

$$\sum_{j=1}^{n} x_{\alpha j} = m, \ \forall \ \alpha \in N \tag{3}$$

$$\sum_{i=1}^{n} x_{i\alpha} = p, \ \forall \ \alpha \in N \tag{4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \ \forall \ j \in N / \{\alpha\} \tag{5}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \ \forall \ i \in N / \{\alpha\} \tag{6}$$

$$1 \leq | S_k | \leq n - m + 1; \ \forall \ k \in K, \text{where } S_k \text{ be the set of cities covered by the } k^{th} \text{ salesman} \tag{7}$$

$$+ \text{Sub tour/illegal tour elimination constraints} \tag{8}$$

$$x_{ij} \in \{0,1\} \ \forall i, j \in N \tag{9}$$

The 2nd Constraint guarantees that any feasible solution must include $m + n - q - 1$ edges. The 3rd and 4th Constraints ensure that m salesman departs from the depot city (α) and exactly p salesman must return to it. The 5th and 6th Constraints guarantee that a salesman visits each city exactly once and can depart from each city at most once. The 7th Constraint ensures that each salesman covers atleast 1 city and at most $n - m + 1$ number of cities. The 8th Constraint is intended to eliminate the sub-tours from the solution. Finally, Constraint (9) denotes the binary variable.

3. Genetic algorithm

This section provides an overview of both the conventional Genetic Algorithm (GA) and the proposed algorithm. The GA stands out as one of the widely adopted metaheuristic algorithms in evolutionary computation for addressing various combinatorial optimization problems, as highlighted by Goldenberg (1989). Originating from the survival of the fittest strategy introduced by Holland in 1975, GA is an adaptive exploration method. Genetic Algorithms are extensively utilized for solving the Multiple Traveling Salesman Problem (MTSP) and its variants, as evidenced by the following studies: (Tjandra *et al.,* 2022 and Wang *et al.,* 2021). Typically, a GA commences with an initial set of solutions constituting the initial population, referred to as chromosomes, where all genetic information is encoded. Each element within the chromosome is treated as a gene. The efficiency of a chromosome is evaluated through a fitness value. The GA process involves selecting two parent chromosomes with high fitness values randomly from the population. Following the selection, a crossover operation is performed between the chosen parent chromosomes to generate two new chromosomes. The resulting chromosomes replace the old ones if they exhibit superior fitness values. To maintain diversity within the population, a mutation operation is applied to the newly generated chromosomes. This cycle of selection, crossover, and mutation iterates, producing novel chromosomes until the size of the new population matches that of the old one. The updated population is then utilized to initiate the next iteration. Notably, the probability of selecting superior chromosomes for crossover is higher, and the newly generated chromosomes are more likely to inherit the characteristics of their parent chromosomes. The search process continues for several generations until predefined conditions are met, constituting a complete iteration of the classical GA.

3.1. Proposed GA

To achieve an optimal or suboptimal solution for the OCMTSP, various crucial factors come into play. These factors encompass chromosome representation, population initialization, computation of fitness values, selection mechanisms, crossover operations, mutation operators, as well as the fine-tuning of GA parameters. The diversity in GA approaches stems from variations in how encoding, crossover, and mutation operations are employed, leading to divergence in the search process. Consequently, the redesign of these fundamental operations is imperative to ensure the attainment of optimal or suboptimal solutions. The foundational components of the proposed GA for OCMTSP are outlined as follows:

a. Chromosome representation

To efficiently address the OCMTSP, it is crucial to adopt a suitable chromosome representation. Given that the MTSP is a generalized form of the TSP, it is possible to adapt TSP chromosome representations with minor adjustments for OCMTSP. Various techniques have been proposed for representing TSP solutions as chromosomes, including path representation (Larranaga *et al.,* 1999), matrix representation (Khan *et al.,* 2009), double chromosome representation (Riazi, 2019), and multi-chromosome representation (Singh *et al.,* 2018).

In this study, the OCMTSP solution is depicted using a multi-chromosome representation, a technique previously employed by researchers such as Albayrak and Allahverdi (2011) and Király and Abonyi (2015). This approach involves utilizing the same number of chromosomes as there are salesman. The length of each chromosome, which dynamically changes, is

determined by the number of cities assigned to each respective salesman. This choice of representation is made to suit the specific characteristics and constraints of the OCMTSP, aligning with the objectives of this research. Assume that the first chromosome is p_1, the second is p_2, and so on. As a result, total number of cities in the multi-chromosome representation equals to $\sum_{i=1}^{m} p_i = n-1$ (since depot city is not included in any of the chromosome). Cities are randomly chosen to be assigned to salesman. For example, let there will be 17 cities $(n=17)$ including depot city (α = 1), four salesman (m = 4) are positioned at the depot city. Of the four salesman, two salesman (p = 2) are intended for closed paths and other two (q = 2) are devoted for open paths. Then the multi-chromosome representation of an arbitrary OCMTSP solution with seventeen (n = 17) cities and with four salesman $(m = p+q = 4$; where $p = 2$ & $q = 2)$ is shown in Fig. 3. For the four salesman involved, the chromosomes are partitioned into four distinct parts, each uniquely represented by a different colour, as depicted in Fig. 3. The numerical values within each segment denote the cities covered by the respective salesman. To create a closed tour for each salesman, the home city (1, for instance) is inserted at both the beginning and the end of each segment. It's important to note that the insertion of the home city (1) at the end is only applicable for closed tours but not for open tours. The route plan for the four salesman is outlined below:

Salesman 1: 1→ 4 → 9 → 15 → 12 →6→ 1; **Salesman 2**: 1→ 14→ 5 → 7 →3 → 1 and

Salesman 3: 1→ 2 → 10 → 17; **Salesman 4**: 1→ 8 → 16 → 13→11. Here, closed paths are associated with the salesman 1 and 2, whereas, open paths are assigned to the salesman 3 and 4.

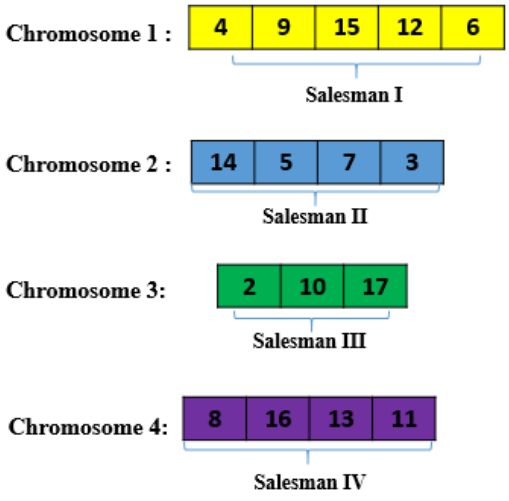

Fig. 3. Illustration of a multi-chromosome representation for a 17-city OCMTSP with 4 salesman

b. Evaluation of fitness function

The objective function, as expressed in Eq. (1), aligns with our fitness function in this study. In the context of the OCMTSP, the fitness value serves as a measure of the total traversal distance incurred by m salesman while visiting a set of n cities, including the home city. It is noteworthy that, in this context, a higher fitness value indicates a more favourable chromosome. Therefore, the better the chromosome in terms of solving the OCMTSP, the greater the corresponding fitness value. The fitness function effectively captures the essence of optimizing the total traversal distance for each salesman, and thus, maximizing the fitness value signifies superior solutions within the genetic algorithm framework.

c. Selection

In our approach, we employ tournament selection, where 8 individuals compete for survival. The chromosome exhibiting the lowest fitness value successes in the tournament, earning the privilege of being chosen for the generation of new individuals. The selected individual seamlessly integrates into the new population without any modifications. This process ensures that more favourable chromosomes are likely to be chosen, while less desirable ones are discarded. The underlying principle of this selection strategy is to transmit high-quality chromosomes to the next generation, enhancing both evaluation efficiency and the convergence towards optimal and sub-optimal solutions.

d. Mutation Operators

Following the selection process, the mutation operation is promptly executed. Its primary goal is to prevent the GA from becoming trapped in local optima and to enhance the genetic diversity within the population. In this study, we incorporate

seven mechanisms—namely, Flip, Swap, Slide, Crossover, Flip + Crossover, Swap + Crossover, and Slide + Crossover—adopted from the work of Király & Abonyi (2015). These mechanisms collectively contribute to enhancing the population of candidate solutions across generations, thereby maintaining genetic diversity within the population. Consequently, this approach mitigates the risk of the algorithm being confined to local optima by facilitating exploration across a broader search space. As previously discussed, there exist various genetic operators in the literature. For instance, a multi-chromosomal mutation can be derived by combining a series of single-chromosomal mutations. While the majority of these operators can be derived from others, introducing a new representation may necessitate the inclusion of additional genetic operators. The operators outlined below can be constructed from existing simple operators. The terms "In-route mutations" and "Cross route mutations" denote two distinct sets of mutation operators. In-route mutation operators, such as flip or gene sequence inversion, modify only two genes within a chromosome and operate within a single chromosome. Larranaga *et al.,* (1999) serve as reliable sources for these "classical" representations and operators. On the other hand, a cross-route mutation operator simultaneously alters several chromosomes. It is important to note that this operator may bear similarity to the standard crossover operator when utilizing conventional notation and treating chromosomes as individual entities.

To begin, the application of an in-route mutation known as the flip operator is initiated, modifying a set of genes within a chromosome. Subsequently, a Swap operator is employed, transposing the gene sequences between two chromosomes, resulting in the creation of a new offset. Following this, a slide operator is applied, moving the last gene from each chromosome to the beginning of another. In addition to in-route mutations, a cross-route mutation employed in this study is the crossover operator, which executes a one-point crossover between two chromosomes. In this process, two random crossover points are selected, and the tails of the two chromosomes are swapped to generate new offspring. The effective utilization of these basic mutation operators is visually depicted in Figs. 4-7.

Utilizing the aforementioned simple mutations, more complex mutations, namely Flip + Crossover, Swap + Crossover, and Slide + Crossover, can be derived. Figure 8 depicts the process when two cross-route operators are sequentially applied, resulting in a complex mutation. Initially, a Swap operator is employed, transposing the gene sequences between two chromosomes and moving the last gene from each chromosome to the beginning of another. Subsequently, another Swap operator is applied, generating the new offset.

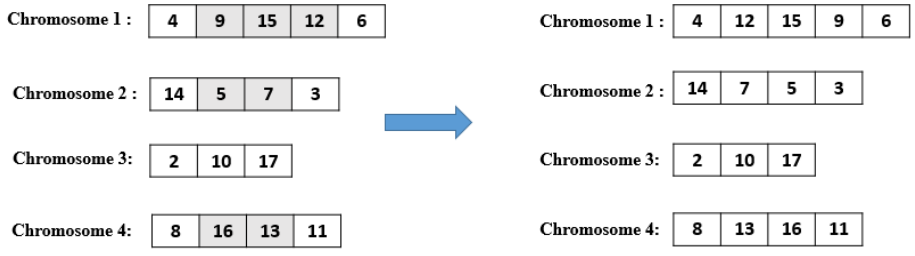

Fig. 4. Illustration of in-route mutation – "Flip"

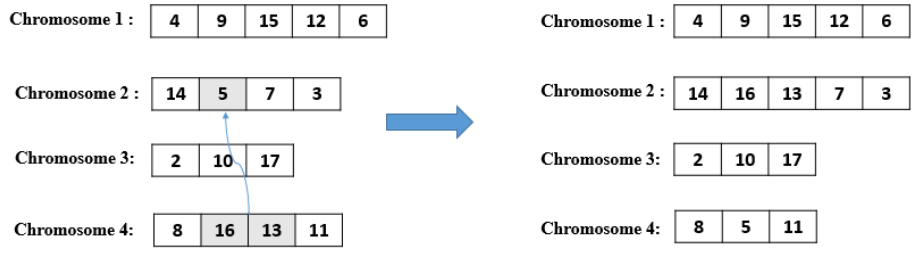

Fig. 5. Illustration of cross-route mutation – "Swap"

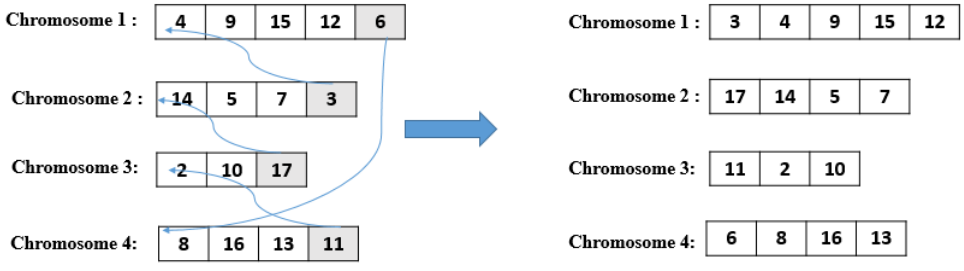

Fig. 6. Illustration of cross-route mutation – "Slide"

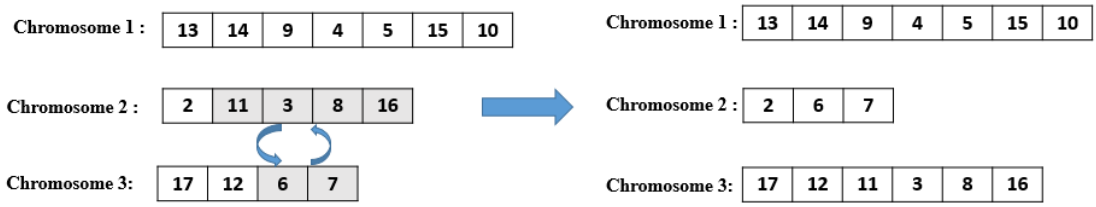

Fig. 7. Illustration of cross-route mutation – "One-point crossover"

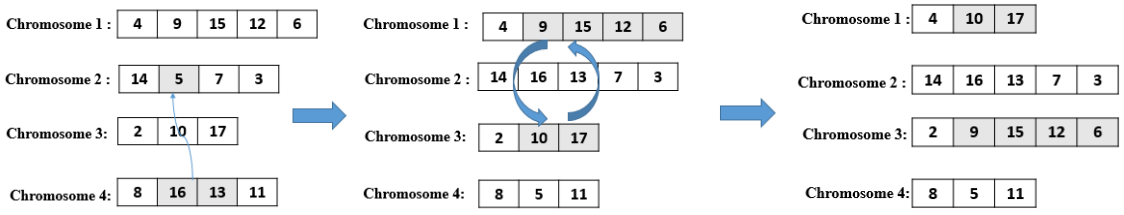

Fig. 8. Illustration of cross-route mutation – complex mutation : "Swap + Crossover"

d. GA parameters

The algorithm's effectiveness is determined by its parameter values, encompassing the population size, mutation probability rate, and termination criterion. A thorough calibration process was undertaken to establish optimal values for the genetic algorithm parameters. The population size is set at 160, and the iteration number is fixed at 3000. The utilization of crossover and mutation operators with diverse probability values is designed to guide the Genetic Algorithm (GA) toward the convergence of efficient solutions. The termination condition for the current GA is contingent upon reaching the maximum number of generations. The flow diagram of the proposed GA is illustrated in Fig. 9.

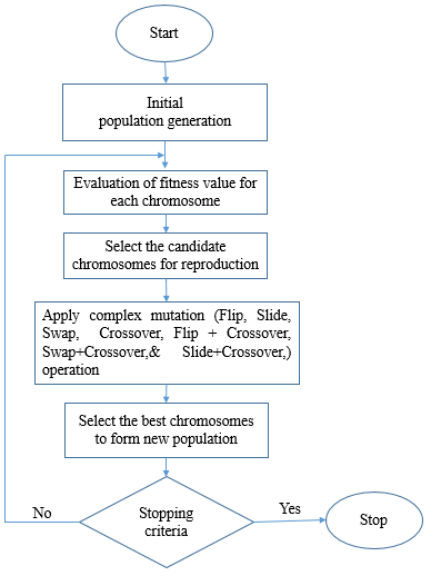

Fig. 9. The workflow of proposed GA

4. Computational Results

In this section, we present the experimental results attained by employing the proposed GA. Subsequently, we provide a comprehensive analysis of the algorithm's performance by comparing it to alternative state-of-art algorithms. Our experimentation involved the utilization of diverse benchmark instances sourced from TSPLIB to assess the algorithm's performance. The implementation of the algorithm was carried out in MATLAB 2023a, executed on a PC equipped with MS Windows 2010 and an Intel Core i3-5005U CPU operating at 2 GHz and an 8GB RAM. The algorithm was subjected to 3000 iterations, population size 160, with a termination condition defined as the stabilization of the best solution over ten consecutive iterations. The reported results represent the mean outcomes derived from 50 independent runs of the algorithm.

4.1 Comparative results of OCMTSP over asymmetric TSP benchmark instances

This section presents a comparative study of OCMTSP over asymmetric TSP benchmark instances generated from TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/). To measure the performance of the proposed GA, it has been tested over 17 asymmetric TSP benchmark instances generated from *br17, ftv33, ftv35, and ftv44.* The GA results obtained are compared against with the best know results achieved by Lexi-search algorithm (LSA) (Thenepalle & Singamsetty, 2019) and the TSP solver LKH-3 (http://webhotel4.ruc.dk/~keld/research/LKH-3/). Table 1 comprehensively presents the results of our analysis. Notably, in 3 out of the 17 instances, the solutions obtained through the GA align precisely with those derived from Lexi-search algorithm (LSA) and the TSP solver LKH-3. For the remaining cases, the GA solutions demonstrate proximity to the best-known solutions. This observation is reinforced by examining deviation percentage values calculated using equation (10). Additionally, it is noteworthy that the proposed GA exhibits a marked advantage in terms of CPU runtime, ranging from 6.12 seconds to 6.92 seconds.

$$Deviation\,(\%) = \frac{Best\,GA\,solution - Optimal\,solution}{Optimal\,solution} \times 100 \qquad (10)$$

Table 1 Comparative results of GA against LSA and LKH-3 for OCMTSP over asymmetric TSP benchmark instances

Instance	|*N*|	*m*	*p*	*q*	BKS by LSA	LKH-3	Proposed GA		Deviation (%)	CPU Runtime (In Sec.)
							Best	Worst		
br17	17	5	3	2	35	35	39	48	11.43	6.62
		6	4	2	41	41	46	54	12.20	6.48
		4	3	1	35	35	35	41	0.00	6.17
		5	2	3	30	30	30	36	0.00	6.42
		6	2	4	33	33	38	44	15.15	6.65
ftv33	34	5	3	2	1240	1239	1305	1378	5.33	6.80
		7	4	3	1278	1278	1278	1416	0.00	6.92
		6	3	3	1225	1225	1273	1359	3.92	6.71
		6	2	4	1185	1185	1206	1325	1.77	6.87
ftv35	36	6	2	4	1284	1283	1308	1421	1.95	6.85
		5	2	3	1307	1304	1338	1405	2.61	6.64
		8	3	5	1324	1324	1404	1516	6.04	6.28
		7	3	4	1328	1328	1439	1497	8.36	6.12
ftv44	45	5	3	2	1619	1577	1737	1870	10.15	6.67
		4	3	1	1691	1595	1732	1861	8.59	6.76
		6	4	2	1678	1629	1776	1878	9.02	6.78
		6	3	3	1603	1549	1674	1891	8.07	6.84
ftv55		5	3	2	–	–	1906	2174	-	6.45
		6	3	3	–	–	1829	2112	-	7.03
		7	4	3	–	–	2017	2163	-	7.21
		8	5	3	–	–	2134	2432	-	7.30

|*N*|=*n* – Number of cities; *m* – Number of salesman; *p* – Number of internal salesman; *q* – Number of external salesman; BKS– best known solution by LSA ; GA– Genetic algorithm

4.2 Experimental results of OCMTSP over symmetric TSP benchmark instances

The author's would like to highlight that as far as their knowledge extends, there exists only a single study addressing the asymmetric OCMTSP. Consequently, no evaluations comparing algorithmic performance against symmetric benchmark test cases have been conducted. However, comprehensive experimentation has been undertaken, involving the evaluation of the proposed Genetic Algorithm (GA) on symmetric TSP benchmark instances. The benchmark dataset comprises a total of 90 distinct test cases, categorized into 15 groups. Each group consists of six distinct test cases, each with varying parametric (i.e. $m, p \,\&\, q$) values. The results of these experiments are presented in Table 2.

The results reveal that the CPU runtime required for the proposed GA to determine the best solution varies between 23.05 seconds and 28.70 seconds, as visually depicted in Fig. 10. It's important to emphasize that CPU runtimes exhibit inconsistency, making it challenging to pinpoint which specific parameters or factors might be influencing these variations in performance. These findings demonstrate the strong performance of the proposed algorithm in terms of computational efficiency. Further, mean absolute deviation (MAD) for each test case has been computed. Based on the obtained, it is evident that the MAD implies that the solutions generated by the proposed GA in ten separate runs exhibit a slight tendency to cluster around the average solution. It should also be emphasized that the results presented here can serve as a valuable reference for future research comparisons. Finally, the best route plans for the instance *eil51* in each of the six cases are now depicted in Figs. 11-16.

Table 2
GA performance results on symmetric TSP benchmark instances for OCMTSP

Instance	|*N*|	*m*	*p*	*q*	Proposed GA			Mean absolute deviation	CPU Runtime
					Best	Worst	Mean		
att48	48	4	2	2	33361.4	37128.56	35173.49	1139.186	25.34
		4	3	1	35604.1	39945.8	37934.7	1173.1	25.81
		5	3	2	35629	27828.1	36496.5	693.3904	26.73
		5	2	3	32822.85	37346.93	34753.88	1034.603	24.93
		6	3	3	35742.83	39636.78	37216.01	713.083	26.46
		6	4	2	38272.72	39916.64	39370.36	373.4672	26.57
eil51	51	4	2	2	452.6784	511.2921	469.5353	14.008254	23.05
		4	3	1	466.425	494.797	478.962	6.443216	23.92
		5	3	2	463.687	512.112	482.091	11.3762	24.10
		5	2	3	455.6285	488.746	468.4005	10.36111	24.93
		6	3	3	475.454	510.5049	489.2125	8.86626	25.34
		6	4	2	494.7597	519.4884	505.6106	8.544784	25.96
berlin52	52	4	2	2	7586.835	8424.335	7931.529	196.09442	23.21
		4	3	1	7893.51	8687.9	8449.74	226.6396	23.88
		5	3	2	7810.855	8828.088	8246.682	232.4367	24.91
		5	2	3	7557.154	8286.813	7889.646	188.97516	25.06
		6	3	3	7696.653	9152.853	8221.316	305.89788	25.82
		6	4	2	8140.038	8934.301	8567.904	179.811	26.03
st70	70	4	2	2	697.4394	728.9744	712.53703	6.85011	23.73
		4	3	1	735.4455	799.5496	766.2036	18.31968	24.86
		5	3	2	726.852	774.82	750.864	12.31896	23.94
		5	2	3	697.0462	724.1514	709.7671	7.38743	25.79
		6	3	3	731.1449	766.7738	754.7678	9.54792	27.78
		6	4	2	767.673	814.1438	795.8641	11.5507	25.96
eil76	76	4	2	2	575.8659	630.9071	600.8822	12.74806	24.61
		4	3	1	595.435	641.946	615.711	11.63961	24.03
		5	3	2	607.011	645.771	624.112	9.245	25.30
		5	2	3	576.0584	625.2755	596.96	13.42114	25.86
		6	3	3	588.1477	657.4689	615.093	20.196196	26.84
		6	4	2	632.0019	667.3908	644.222	8.07833	27.21
pr76	76	4	2	2	105467	116057	111261.26	2216.42	24.02
		4	3	1	115731	122752	119782	1718.27	23.88
		5	3	2	117523	125156	120001	1796.544	25.44
		5	2	3	113527.1	118431.8	115626.1	1328.79	24.89
		6	3	3	118915.2	125947.6	121323.2	1242	26.84
		6	4	2	122898.3	130930.1	127788.57	2511.396	27.21
gr96	96	4	2	2	524.3464	593.7042	570.64574	14.220152	23.99
		4	3	1	570.636	613.654	597.045	9.684316	24.90
		5	3	2	580.726	613.419	594.639	11.01297	25.98
		5	2	3	545.5041	592.8673	568.6016	13.037284	26.08
		6	3	3	572.1919	612.56	592.95109	11.23225	27.43
		6	4	2	601.3454	652.6115	635.72836	10.9706	26.94
rat99	99	4	2	2	1350.443	1425.148	1385.2342	18.39896	26.65
		4	3	1	1406.191	1481.838	1444.7665	22.2817	26.14
		5	3	2	1444.89	1551.1	1486.96	23.49656	28.62
		5	2	3	1366.866	1429.921	1406.282	19.84928	27.76
		6	3	3	1433.659	1508.559	1472.9271	19.29072	26.94
		6	4	2	1498.812	1593.716	1555.0075	31.9035	27.14
rd100	100	4	2	2	8235.437	8868.961	8559.9134	176.24072	24.66
		4	3	1	8823.73	9153.765	9017.2148	98.06084	25.64
		5	3	2	8742.42	8998.39	8882.01	80.7202	26.81
		5	2	3	8428.115	9080.572	8739.408	217.9691	25.40
		6	3	3	8589.89	9398.539	9082.7171	214.91128	27.88
		6	4	2	8877.86	9617.527	9348.9187	220.78836	27.52
pr107	107	4	2	2	39660.88	41961.71	41037.002	745.3516	25.21
		4	3	1	42363.75	44967.32	43687.389	669.7388	24.33
		5	3	2	41820.7	49464.9	45623.5	2417.362	26.81
		5	2	3	39632.08	41347.81	40298.92	441.9876	26.92
		6	3	3	41426.72	43936.04	42865.367	505.8884	27.88
		6	4	2	44400.43	47751.89	46473.428	1043.0044	26.71
gr137	137	4	2	2	871.5383	906.2047	889.74713	11.066684	26.72
		4	3	1	939.7482	997.3293	974.3590	14.049608	24.22
		5	3	2	987.999	1006.91	997.229	5.19134	25.26
		5	2	3	953.8393	990.8595	981.3576	6.635984	27.21
		6	3	3	1043.887	1098.558	1074.5439	17.42652	26.94
		6	4	2	1064.442	1142.739	1125.0803	15.74638	28.13

Table 2
GA performance results on symmetric TSP benchmark instances for OCMTSP (Continued)

Instance	$\|N\|$	*m*	*p*	*q*	Proposed GA			Mean absolute deviation	CPU Runtime (In Sec.)
					Best	Worst	Mean		
ch150	150	4	2	2	7852.04	8205.907	8013.0271	112.5723	25.30
		4	3	1	7471.565	8249.651	8021.609	192.405	26.15
		5	3	2	7625.57	8184.28	7948.06	159.1553	25.39
		5	2	3	7359.895	8137.189	7943.105	165.08588	25.12
		6	3	3	7880.665	8461.061	8110.5373	153.94136	27.54
		6	4	2	8075.022	8443.949	8287.0034	120.83192	26.31
d198	198	4	2	2	21026.24	22807.33	22027.447	477.4316	25.80
		4	3	1	22591.03	24364.6	23661.997	400.753	26.14
		5	3	2	24519.6	25890.5	25353.2	334.2564	26.02
		5	2	3	23428.73	24216.49	23817.87	243.765	26.69
		6	3	3	25961.58	27442.89	26788.02	361.476	27.71
		6	4	2	27781.63	29289.32	28826.913	419.673	27.84
gr202	202	4	2	2	651.5022	715.5991	687.34464	19.0114	25.18
		4	3	1	683.9505	727.6095	708.99506	9.5166	26.47
		5	3	2	687.173	781.258	751.113	19.8872	26.12
		5	2	3	665.071	711.6968	694.3388	11.1299	26.80
		6	3	3	713.7567	790.7828	765.31276	16.1709	28.70
		6	4	2	747.3569	805.2652	784.25438	15.7902	27.32
a280	280	4	2	2	4868.733	5314.822	5166.8091	106.00546	25.32
		4	3	1	5084.362	5413.642	5283.3941	84.92568	26.54
		5	3	2	5246.59	5837.96	5645.53	111.8587	27.40
		5	2	3	5159.341	5538.281	5378.421	96.6912	27.49
		6	3	3	5309.086	5769.293	5624.2374	107.87732	27.82
		6	4	2	5508.558	5860.253	5689.2671	87.6179	27.80

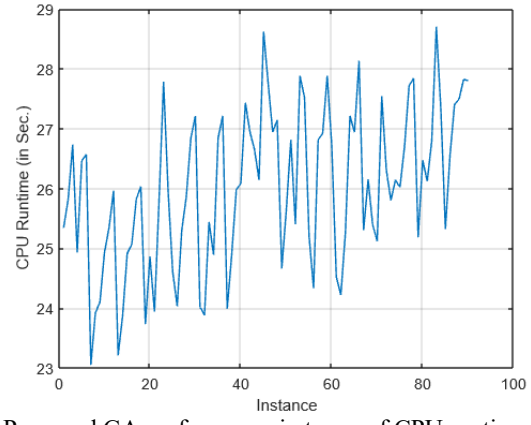

Fig. 10. Proposed GA performance in terms of CPU runtime (in sec.)

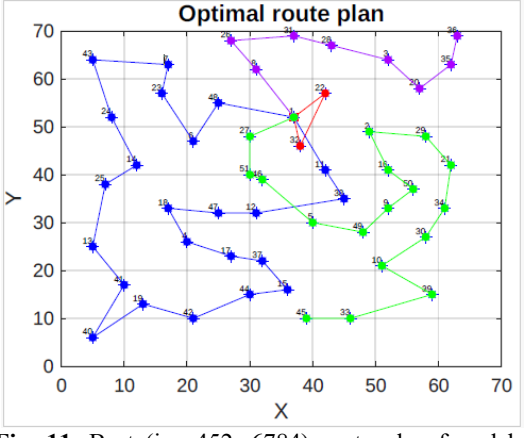

Fig. 11. Best (i.e. 452. 6784) route plan found by proposed GA on *eil51* with 4 salesman ($p = 2$ and $q = 2$)

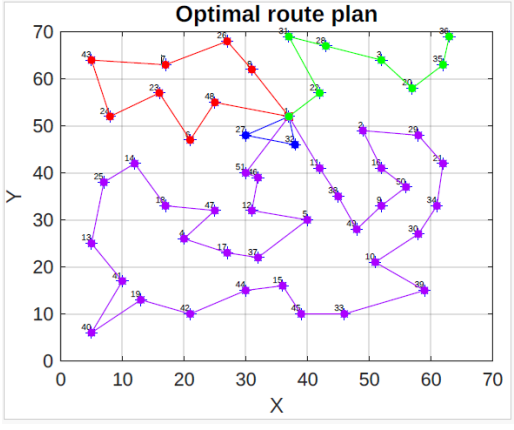

Fig. 12. Best (i.e. 466.4254) route plan found by proposed GA on *eil51* with 4 salesman ($p = 3$ and $q = 1$)

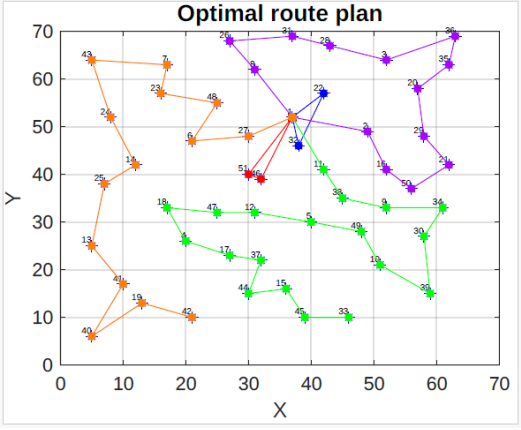

Fig.13. Best (i.e. 463.6873) route plan found by proposed GA on *eil51* with 5 salesman ($p = 3$ and $q = 2$)

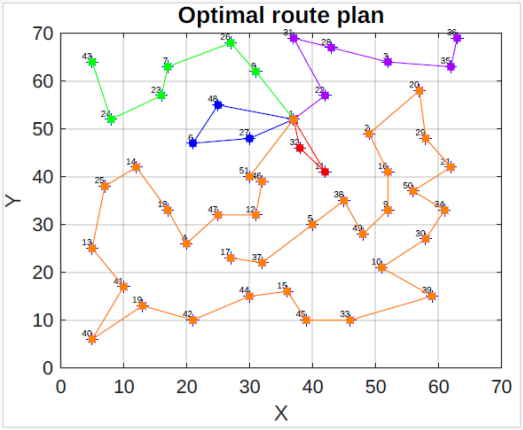

Fig.14. Best (i.e. 455.6285) route plan found by proposed GA on *eil51* with 5 salesman ($p = 2$ and $q = 3$)

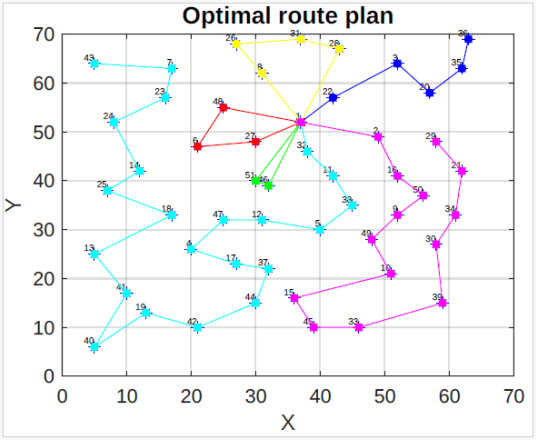

Fig. 15. Best (i.e. 475.4540) route plan found by proposed GA on *eil51* with 6 salesman ($p = 3$ and $q = 3$)

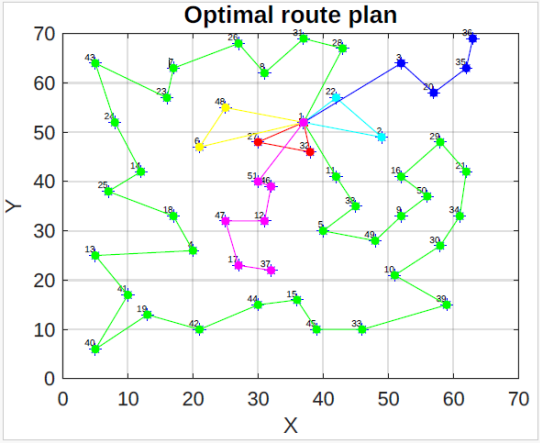

Fig. 16. Best (i.e. 494.7597) route plan found by proposed GA on *eil51* with 6 salesman ($p = 4$ and $q = 2$)

5. Conclusions

In this research paper, we developed a genetic algorithm that depend on a multi-chromosome approach to tackle the open close multiple traveling salesman problem (OCMTSP). As far as the author's knowledge, this is the second study and first meta-heuristic algorithm designed for addressing the OCMTSP. To assess the effectiveness of our GA, incorporating the multi-chromosome method, we conducted a comparative analysis over 17 asymmetric TSP benchmark instances against other established methods such as LSA, and LKH-3. Our computational results conclusively demonstrate that our proposed GA outperforms all other methods in terms of computational runtime and reasonably performs well in terms of solution quality. Additionally, we conducted a thorough validation of GA effectiveness and performance by testing it on a diverse set of instances 90 symmetric instances with different parametric $(i.e.\ m,\ p\ \&\ q)$ values derived from the TSPLIB Library.

Based on the statistical metrics employed, it is evident that the proposed algorithm excels in addressing OCMTSP. Notably, it's important to highlight that the proposed GA is versatile and effective for solving both symmetric and asymmetric instances of the problem. As it is new attempt, these findings serve as a valuable benchmark for future investigations, offering insights into the problem's complexity and highlighting potential areas for improvement using state-of-the-art algorithms. However, it is essential to acknowledge the limitations of this research. The proposed algorithm, while efficient, may not guarantee the optimal solutions. Despite the limitations, this work serves as a stepping stone for future research, as we look forward to seeing advancements in the context of OCMTSP scenerios.

Future research directions can focus on various aspects. Firstly, exploring more advanced algorithms, such as meta-heuristic techniques, could lead to further improvements in optimizing OCMTSP. Secondly, incorporating practical constraints into the model could render it more realistic and robust, aligning it better with real-world scenarios.

References

Albayrak, M., & Allahverdi, N. (2011). Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. *Expert Systems with Applications*, *38*(3), 1313 –1320.

Al-Omeer, M. A., & Ahmed, Z. H. (2019, April). Comparative study of crossover operators for the MTSP. In *2019 International Conference on Computer and Information Sciences (ICCIS)* (pp. 1–6). IEEE.

Alves, R. M., & Lopes, C. R. (2015, May). Using genetic algorithms to minimize the distance and balance the routes for the multiple traveling salesman problem. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 3171–3178). IEEE.

Bagley, J. D. (1967). *The behavior of adaptive systems which employ genetic and correlation algorithms*. University of Michigan.

Bao, C., Yang, Q., Gao, X. D., & Zhang, J. (2021, December). A comparative study on population-based evolutionary algorithms for multiple traveling salesman problem with visiting constraints. In 2021 *IEEE Symposium Series on Computational Intelligence* (SSCI) (pp. 01–08). IEEE.

Belhor, M., El-Amraoui, A., Jemai, A., & Delmotte, F. (2023). Learning-based metaheuristic approach for home healthcare optimization problem. *Computer Systems Science and Engineering*, *45*, 1 –19.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313.

Brown, E. C., Ragsdale, C. T., & Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problem. *International Journal of Information Technology & Decision Making*, *6*(02), 333–347.

Carter, A. E., & Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, *175*(1), 246–257.

Changdar, C., Mondal, M., Giri, P. K., Nandi, U., & Pal, R. K. (2023). A two-phase ant colony optimization based approach for single depot multiple travelling salesman problem in Type-2 fuzzy environment. *Artificial Intelligence Review*, *56*(2), 965 –993.

Changdar, C., Pal, R. K., & Mahapatra, G. S. (2017). A genetic ant colony optimization based algorithm for solid multiple travelling salesman problem in fuzzy rough environment. *Soft Computing*, *21*(16), 4661–4675.

Cheikhrouhou, O., & Khoufi, I. (2021). A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. *Computer Science Review*, 40, 100369.

Garey, M. R., & Johnson, D. S. (1979). *A guide to the theory of NP-completeness, Computers and Intractability*. New York.

Goldenberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Newyork.

Gomes, D. E., Iglésias, M. I. D., Proença, A. P., Lima, T. M., & Gaspar, P. D. (2021). Applying a genetic algorithm to a m-TSP: case study of a decision support system for optimizing a beverage logistics vehicles routing problem. *Electronics*, *10*(18), 2298.

Harrath, Y., Salman, A. F., Alqaddoumi, A., Hasan, H., & Radhi, A. (2019). A novel hybrid approach for solving the multiple traveling salesman problem. *Arab Journal of Basic and applied sciences*, *26*(1), 103 –112.

Jiang, C., Wan, Z., & Peng, Z. (2020). A new efficient hybrid algorithm for large scale multiple traveling salesman problems. *Expert Systems with Applications*, *139*, 112867.

Kaliaperumal, R., Ramalingam, A., & Sripriya, J. (2015, March). A modified two part chromosome crossover for solving MTSP using genetic algorithms. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)* (pp. 1–4).

Kara, I., & Bektas, T. (2006). Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*, *174*(3), 1449–1458.

Khan, F. H., Khan, N., Inayatullah, S., &Nizami, S. T. (2009). Solving TSP problem by using genetic algorithm. *International Journal of Basic & Applied Sciences*, *9*(10), 79 –88.

Király, A., & Abonyi, J. (2011). Optimization of multiple traveling salesman problem by a novel representation based genetic algorithm. In *Intelligent computational optimization in engineering* (pp. 241–269). Springer, Berlin, Heidelberg.

Király, A., & Abonyi, J. (2015). Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API. *Engineering Applications of Artificial Intelligence*, *38*, 122 –130.

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., &Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, *13*(2), 129 –170.

Lou, P., Xu, K., Yan, J., & Xiao, Z. (2021, April). An Improved Partheno-Genetic Algorithm for Open Path Multi-Depot Multiple Traveling Salesman Problem. In *Journal of Physics: Conference Series* (Vol. 1848, No. 1, p. 012002). IOP Publishing.

Malik, W., Rathinam, S., & Darbha, S. (2007). An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem. *Operations Research Letters*, *35*(6), 747–753.

Mzili, I., Mzili, T., & Riffi, M. E. (2023). Efficient routing optimization with discrete penguins search algorithm for MTSP. *Decision Making: Applications in Management and Engineering, 6*(1), 730 –743.

Öncan, T. (2007). A survey of the generalized assignment problem and its applications. *INFOR: Information Systems and Operational Research*, *45*(3), 123–141.

Riazi, A. (2019). Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *SN Applied Sciences*, *1*(11), 1397.

Sarin, S. C., Sherali, H. D., Judd, J. D., & Tsai, P. F. J. (2014). Multiple asymmetric traveling salesman problem with and without precedence constraints: Performance comparison of alternative formulations. *Computers & Operations Research*, *51*, 64–89.

Shuai, Y., Yunfeng, S., & Kai, Z. (2019). An effective method for solving multiple travelling salesman problem based on NSGA-II. *Systems Science & Control Engineering*, *7*(2), 108 –116.

Singamsetty, P., & Thenepalle, J. (2021). An efficient genetic algorithm for solving open multiple travelling salesman problem with load balancing constraint. *Decision Science Letters*, *10*(4), 525–534.

Singh, D. R., Singh, M. K., Singh, T., & Prasad, R. (2018). Genetic algorithm for solving multiple traveling salesman problem using a new crossover and population generation. *Computación y Sistemas*, *22*(2), 491 –503.

Sofge, D., Schultz, A., & De Jong, K. (2002, April). Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema. In *Workshops on Applications of Evolutionary Computation* (pp. 153–162). Springer, Berlin, Heidelberg.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, *124*(2), 267–282.

Thenepalle, J., & Singamsetty, P. (2019). An open close multiple travelling salesman problem with single depot. *Decision Science Letters*, *8*(2), 121–136.

Tjandra, S. S., Setiawan, F., & Salsabila, H. (2022). Application of Genetic Algorithms to Solve MTSP Problems with Priority (Case Study at the Jakarta Street Lighting Service). *Journal on Optimization of Systems in Industries*, *21*(2), 75 –86.

Wang, M., Ma, T., Li, G., Zhai, X., & Qiao, S. (2020). Ant colony optimization with an improved pheromone model for solving MTSP with capacity and time window constraint. *IEEE Access*, *8*, 106872–106879.

Wang, Y. D., Lu, X. C., & Shen, J. R. (2021). Improved Genetic Algorithm (VNS-GA) using polar coordinate classification for workload balanced multiple Traveling Salesman Problem (mTSP). *Advances in Production Engineering & Management*, *16*(2), 173 –184.

Xu, X., Yuan, H., Liptrott, M., & Trovati, M. (2018). Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing, 22*(19), 6567–6581.

Yan, X., Zhang, C., Luo, W., Li, W., Chen, W., & Liu, H. (2012). Solve traveling salesman problem using particle swarm optimization algorithm. *International Journal of Computer Science Issues (IJCSI)*, *9*(6), 264.

Yousefikhoshbakht, M., Didehvar, F., & Rahmati, F. (2013). Modification of the ant colony optimization for solving the multiple traveling salesman problem. *Romanian Journal of Information Science and Technology*, *16*(1), 65–80.

Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesman problem using genetic algorithms. *European journal of operational research*, *228*(1), 72–82.