Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 12, Issue 7, July 2021: 4520-4528

Research Article

Solving an Open *k*-city Travelling Salesman Problem with Ordered Constraint: An Exact Lexi-search Algorithm

[1]Sk.Mastan, [2]U.Balakrishna, [3]G.Sankar Sekhar Raju, [4]T. Jayanth Kumar

Research Scholar, JNTUA, Ananthapuramu, India-515002

Professor, SITAMS, Chittoor, India-517127

Professor, JNTUA, Pulivendula, India-516390

Assistant Professor, Jain (Deemed-to-be University), Kanakapura Rd, Bengaluru, Karnataka, India-562112

Abstract

Thetravelling salesman problem (TSP) with the usual closed-loop setup has been extensively considered. However, TSP with open-loop variant arises in several real-time transportationmodels but has been given very limited attention. This paper addresses anopen *k*-city travelling salesman problem with ordered constraint (*k*OTSPO), a variant of an openTSP in which the salesman need not return to the home city, enough to traverse *k* out of *n* cities and certain cities should be visited in the predetermined order. Although a wide variety of solution techniques for solving TSP and its variants are available in the literature, most of them are heuristic or meta-heuristic algorithmsdue to their NP-hard nature. Thus, very less consideration has been attained for the exact algorithms. This paper develops an exact Lexi-search algorithm (LSA) that effectively enumerates the feasible solutions and the exact solution can be obtained systematically.As there is no existing study on the current model, no comparative results are reported. A numerical example is also illustrated in the provision of the theory.

Keywords: Open travelling salesman problem, Np-hard, Open-loop, Lexi-search algorithm

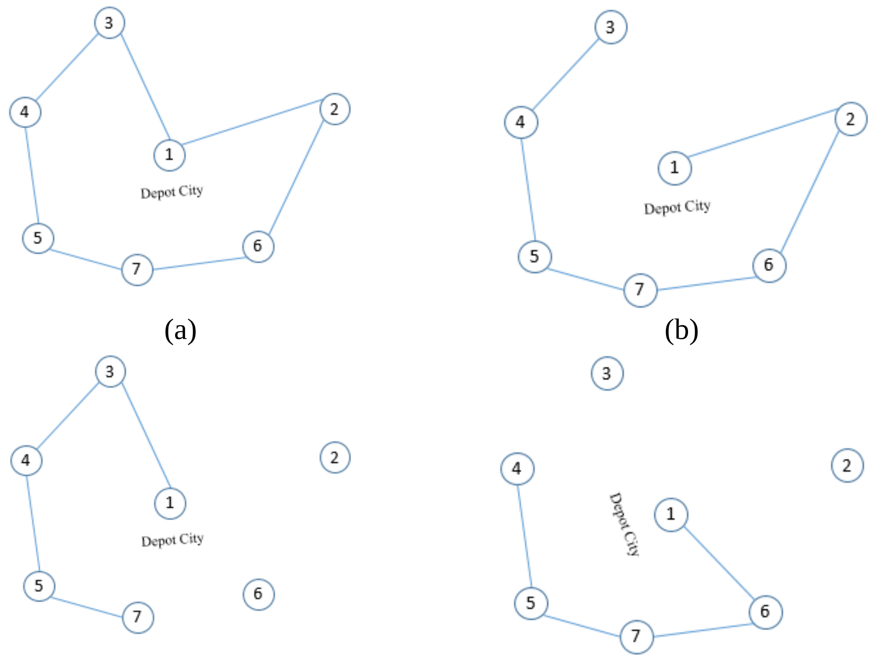
1. Introduction

The travelling salesman problems (TSP) look for to determine the least cost route/closed tour for a salesman to traverse n cities. In graph theory context, the solution to a TSP is same as the least-weight Hamiltonian cycle in a weighted graph. The TSP models have been widely studied and a large number of solution techniques including heuristics [Chauhan et al., 2012 [1]; Akhand et al., (2020) [2]], meta-heuristics [Halim and Ismail (2019) [3]] and exact algorithms [Laporte (1992) [4], Battarra et al., (2014) [5]; Hatamlou, (2018) [6]] have been devoted for solving TSP. Some of the diversified applications where TSP can be seen as vehicle routing problem (Bertsimas and Simchi-Levi, (1996) [7]), Transportation (Hacizade and Kaya, (2018) [8]), Computer wiring (Bharati and Kalshetty, (2016) [9]), UAV swarming (Sathyan et al., (2015) [10]) etc.

Although the TSP received a great attention from the researchers, the works on open travelling salesman problem (OTSP) is still limited. However, in the practical routing models, the salesman needs not to cover all the given *n* cities, but enough to cover *k* out of *n* cities. This problem is known as *k*-city open travelling salesman problem (*k*OTSP). This problem can be applied in various transportation models (Chieng and Wahid, (2014) [11]). Some of the studies on OTSP and its variants are OTSP (Vashisht et al., (2013) [11]; Wang et al., (2013) [12]; Sengupta et al., (2019) [13]; Abdulrazaq et al., (2019) [14]), *k*OTSP (Chieng and Wahid, (2014) [15]), Open close multiple travelling salesman problem (Thenepalle and Singamsetty, (2019) [16]). The above citied are the only studies which were devoted for OTSP to the best of author's knowledge. Some of the studies related to *k*-cityrouting models namelyBi-criteria *k*-city TSP with time threshold (Thenepalle and Singamsetty, (2018) [17]), *k*-city open TSP (Singamsetty et al., (2021) [18], and LPG distribution (Thenepalle and Singamsetty, (2021) [19]).

This study addresses an extension of *k*OTSP called open *k*-cities travelling salesmanproblem with ordered constraint in which the salesman starts from the home city and visits only *k* out of *n* cities, need not return back to the home city and certain specified cities should be visited in the predetermined order with minimum cost. To best of author's knowledge, this variant is only the first study to consider *k*OTSP together with ordered constraint.

Fig.1 depicts the variants of the TSP, OTSP, *k*OTSP and*k*OTSPO. Fig.1 (a) represents the classical TSP where the salesmen starts from the depot city and cover all the given 6 cities and return back to the depot city. Fig. 1 (b) denote a feasible solution of OTSP where the salesman begins from the depot city and cover all the given 6 cities and need not come back to the depot city. However, Fig. 1 (c) depicts a feasible solution of *k*OTSPin which the salesman starts from the depot and need not cover all the 7 cities, but he is restricted to traverse only 4other cities. Finally, Fig. 1 (d) indicates a feasible solution of *k*OTSPO where the salesman begin from depot city and asked to cover only 4 other cities with the ordered constraint (6, 7). This means, the salesmanis constrained to visit 6th city from any city followed by 7th city.



(a) (b)

(c) (d)

Fig.1 Different variants of travelling salesman problem with 7 cities (TSP).
(a) classical TSP, (b) Open travelling salesman problem (OTSP),
(c) *k*-city open travelling salesman problem (*k*OTSP) (d) *k*-city open travelling salesman problem with ordered constraint (*k*OTSPO)

The paper is structured as follows. In Section 2, the problem statement and its mathematical formulation are provided. In Section 3, we present the basic concepts of LSA and the proposed LSA. Finally, a numerical illustration is presentedin Section 4, followed by conclusions in Section 5.

2. Mathematical Formulation

Let an undirected weighted and complete graph $G = G(V, E)$, where $V = \{1, 2, 3, ..., n\}$ and $E = \{(i, j) / i, j \in V\}$ respectively denote the set of n cities and edges. City 1 is the depot city, where the salesman starts his tour. To each edge $(i, j) \in E$ is assigned a positive traversal cost $c_{ij} (c_{ij} \neq c_{ji})$. Let k be a non-negative integer such that $k \leq n$. Let $(c_1, c_2, c_3 ... c_m); 2 \leq m \leq n$ be a collection of cities, which are to be visitedin a specified order. Here, the salesman is restricted to visit the city c_1 before visiting the cities $c_2, c_3, c_4, ..., c_m$, and visit the city c_2 before visiting the cities $c_3, c_4, ..., c_m$ and so on. This constraint is known as city ordering constraint.The problem *k*OTSPO seeks to find the minimum cost salesman tour that starts from depot city and need not come back to the depot city after covering exactly k (including depot city) out of n cities. Further, of the $k-1$ cities, desired cities are to be visited in a specified order. Let the decision variable $x_{ij}, (i, j) \in I \times J$ assumes value 1 when the salesman visits j^{th} city from i^{th} city and 0 otherwise. Another variable z_i that take 1 when i^{th} city is covered and 0 otherwise, then the problem *k*OTSPO can be mathematically formulated as:

(kOTSPO) $\underset{X \in S}{Min}\ c_{ij} x_{ij}$

$$Where\ S = \left\{ X = (x_{ij}) \in R^{n \times n} \;\middle|\; \begin{array}{c} \sum_{j \in V/\{1\}} x_{1j} = 1 \\ \sum_{i \in V/\{1\}} x_{i1} = 0 \\ \sum_{i \in V} x_{ij} \le 1; \forall\, j \in V \\ \sum_{j \in V} x_{ij} \le 1; \forall\, i \in V \\ x_{1c_1} + x_{c_1c_2} + x_{c_2c_3} + ... + x_{c_{i-1}c_i} = k-1;\ c_1, c_2, .., c_i \in V/\{1\} \\ \sum_{i \in V} x_{ij} - \sum_{c \in V/\{1\}} x_{jc} \le 1;\ \forall\, j \in V \\ \sum_{i \in V} z_i = k \\ + \text{City ordering constraint} \\ x_{ij}, z_i \in \{0,1\};\ (i, j) \in E, i \in V \end{array} \right\} \quad (1)$$

The objective function in *k*OTSPO represents the minimization of the overall traversal cost of touring k cities by the salesman. Concerning the constraints, each one is described in detail for better understanding. At first, the salesmen start from the home city and not necessarily return to the home city. The salesman can visit a city and departs from that city at most once. A Hamiltonian path with length $k-1$involves precisely $k-1$edges and k cities, thus the subsequent constraint is enforced to guarantee that there are $k-1$edges. However,this constraint does not guarantee the construction of the feasible path with those $k-1$edges. The degree of each city must be two (except the last city) in the path (one in degree and one out-degree). To maintain this, the subsequent constrainthas been introduced. Hence, it preserves the degree constraint for each city, except the last city in the path. The next constraint represents that the feasible tour that covers exactly k cities. Desired cities are to be visited in a specified order this can be done with city order constraint.Finally, the last constraint, x_{ij} and y_j represents the decision variables that take binary values.

3 Preliminaries of LSA

Obviously, solving combinatorial optimization problems (COP) using exact methods for exact/optimal solutions is highly expensive. Thus, the interest from the researchers has been growing in developing meta-heuristic approaches to tackle COP that may provide best or near-optimal solutions. In fact, effective decisions can be made through exact solutions. In this aspect, developing exact search methods have also been receiving much attention. The lexicographic search algorithm is one such exact method, which employs the concept of construction of words. The entire search process for an optimal schedule can be done in a systematic manner that is analogous to the search of a required word in a dictionary; thus, the name is given as "Lexi-search".

The main difficulty of solving COP using implicit enumeration methods is (1) verifying the feasibility (2) setting effective bounds. There is a difficulty in testing the feasibility for a few problems. To overcome this, a pattern recognition technique based Lexi-search approach (Murthy 1976) [20] has been developed and stated as follows:

"A unique pattern is connected with each solution of a problem. Partial pattern represents a partial solution. An alphabet-table characterizes with the assistance of which the words, representing the pattern are listed in a lexicographic or dictionary order. During the search for an optimal word, when a partial word is considered, first bounds are determined and then the partial words for which the value is less than the trail value are checked for the feasibility**".**

The basic concepts associated with Lexi-search algorithm (LSA) are described below.

3.1 Pattern

A two-dimensional matrix X related to the solution is called a pattern. The values of the pattern X calculated using (2). Where, $V(X)$ defines the value of the objective function and gives the overall cost represented by the solution X . Here, the terms pattern and solution refers the same.

$$V(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2}$$

3.2 Alphabet table

Typically, there exists n^2 ordered pairs for any two dimensional cost matrix $C = [c_{ij}]$ of order n. Since the model concerns about the symmetric cost matrix, either upper or lower triangular matrix is enough to generate the alphabet table. It is known that the upper or lower triangular matrices includes the principal diagonal elements also, but in the present model those elements are neglected as they forms self-loops. Therefore, for any symmetric distance matrix of order n consists of $(n^2 - n)/2$ ordered pairs and only those are considered to generate the alphabet table. All these ordered pairs are to be arranged in an ascending order subject to their costs and are labelled from 1 to $(n^2 - n)/2$.Let $SN = \{1, 2, ..., (n^2 - n)/2\}$, a set of $(n^2 - n)/2$ indices and C^* be an array of distance elements defined in such a way that if $a_1, a_2 \in SN$ and $a_1 < a_2$ then $C^*(a_1) \le C^*(a_2)$. The arrays CC, R and CL represents the cumulative distance, row and column (i.e. indices) of the distance element in C, respectively. For understanding, if $a_k \in SN$ then $C^*(a_k) = C(R(a_k), CL(a_k))$ be the distance in the position of $(R(a_k), CL(a_k))$ from the distance matrix, and $CC(a_k) = \sum_{i=1}^{k} C^*(a_i)$. The *alphabet table* can be formed by augmenting the arrays SN, C^*, CC, R and CL together. Let $L_r = (a_1, a_2, a_3, ..., a_r),\ a_i \in SN$, be an ordered sequence of r indices from SN. The pattern represented by the ordered pairs is independent of a_i in the sequence given by L_r. For uniqueness, the indices in L_r are arranged in ascending order, such that $a_i < a_{i+1},\ i = 1, 2, 3, ..., r-1$. The ordered sequence of r indices L_r from SN is called a word of length r. A word L_r is a sensible word if $a_i < a_{i+1},\ i = 1, 2, 3, ..., r-1$; otherwise, it is non-sensible. L_r is said to be a feasible word if it represents a feasible pattern; otherwise, it is infeasible. Any one of the indices from SN can take up the prime position in the partial word L_r. In this model, the salesman wish to visit exactly k ($\leq n$) cities that also includes depot city. Therefore, for a feasible solution needs $k-1$ edges that form a path of length $k-1$ edges and visits exactly k cities including depot city. If $r < k-1$, then L_r is said to be a partial/incomplete word, whereas $r = k-1$ represents a full length feasible word or simply a word.

3.3 Setting effective bounds

Let $TS = 9999$ (a sufficiently large value) be the trial solution and is considered as an upper bound. The lower bound (LB) of a word L_r is evaluated using $LB(L_r) = V(L_r) + CC(a_r + B - r) - CC(L_r)$ where $B = k-1$. The value of the word $V(L_r)$ can be determined recursively using $V(L_r) = V(L_{r-1}) + C^*(L_r)$ with $V(L_0) = 0$.

3.4 Lexi-search algorithm

The systematic procedure of Lexi-search algorithm (LSA) is described as follows:

Step 1: Initialize: cost matrix $C = [c_{ij}]$, set $TS = 9999$ (trial solution) and required parameters such as n, k and prespecified city orders.

Step 2: **Construction of alphabet table:**
Generate the alphabet table as described in Section 3.2. Return to Step 3.

Step 3: **Computation of bounds:**

a. Initially, the algorithm starts with $L_r = (a_r) = 1$, where $r = 1$.
b. Compute LB of a partial word (L_r) as discussed in Section 3.3.
c. If $LB < TS$, then go to Step 4, otherwise, drop the partial word L_r, discard the word with L_r as leader, since it does not provide optimal solution and thus, reject all the partial words of order *r* that succeeds L_r, go to Step 6.

Step 4: **Feasibility checking:**

a. If the partial word L_r holds the feasibility criteria, then it is feasible otherwise, it is infeasible.
b. If L_r is feasible, then accept it and continue for the next partial word of order $r+1$ and move to Step 5.
c. If L_r is infeasible, then consider the next partial word of order r by considering a new letter that succeeds a_r in its r^{th} place, go to Step 3b.

Step 5: **Concatenation:**

a. If L_r is a full-length feasible word (i.e. $r = k-1$), then $TS = LB(L_r)$, record TS and L_r for further improvement go to Step 7.
b. If L_r is a partial feasible word, then it can be concatenated by using $L_{r+1} = L_r * (a_{r+1})$ where * specifies the string operation and go to Step 3b.

Step 6: If all the words of order *r* are exhausted and the length of the word L_r is 1, then go to Step 8. Otherwise, go to Step 7.

Step 7: **Backtracking:**

a. To explore the solution space, backtracking is performed by assuming current TS as the upper bound and continues the search with the next letter of the partial word of order $r-1$ and go to Step 3b.

b. Repeat the steps 3b to 7 and continue this process until TS has no further improvement and go to Step 8.

Step 8: Stop.

Finally, at the end of the search, TS provide the optimal solution, the latest complete word L_r give the position of the letters andone can find the optimal schedule for connectivity of given cities with the help of L_r.

4. Numerical Illustration

To validate the above developed algorithm, a 7 city numerical illustration is considered for *k*OTSPO, for which $N = \{1,2,3,4,5,6,7\}$, depot city=1, $n = 7, k = 6$ and prespecified ordered cites $(2,5)$. This means the salesman has to start his tour from the depot city and visit 5 out of 7 cities, need not return to the depot city. Further, the salesman has to visit city 5 immediately after visiting the city 2. The asymmetric cost matrix C assumes the non-negative values and is given in Table 1.

Table 1: Cost matrix (C)

$C(i,j)$	1	2	3	4	5	6	7
1	*999*	*10*	*15*	*5*	*16*	*15*	*29*
2	-	*999*	*19*	*22*	*25*	*12*	*1*
3	-	-	*999*	*29*	*7*	*19*	*22*
4	-	-	-	*999*	*15*	*17*	*14*
5	-	-	-	-	*999*	*10*	*11*
6	-	-	-	-	-	*999*	*17*
7	-	-	-	-	-	-	*999*

Alphabet table

Alphabet table construction for the numerical example (given in Table 1) is provided in Table 2. The arrays SN, C^*, CC, R and CL in Table 2 represents serial number, ascending arrangement of values of cost matrix $C = [c_{ij}]$, cumulative time, row, and column indices, respectively.

Table 2: Alphabet Table

$S.N$	C^*	CC	R	CL		$S.N$	C^*	CC	R	CL
1	2	2	1	4		22	12	200	1	5
2	3	5	3	2		23	12	212	3	7
3	4	9	3	6		24	13	225	1	3
4	5	14	6	1		25	13	238	6	3
5	5	19	4	3		26	13	251	1	6
6	8	27	5	7		27	14	265	3	4
7	9	36	6	4		28	14	279	2	6
8	9	45	2	5		29	14	293	5	6
9	10	55	1	2		30	15	308	4	2
10	10	65	6	2		31	15	323	7	3
11	10	75	5	4		32	16	339	3	5
12	10	85	4	7		33	16	355	6	5
13	11	96	3	1		34	16	371	6	7
14	11	107	7	1		35	17	388	5	1
15	11	118	5	2		36	18	406	1	7
16	11	129	2	4		37	19	425	7	4

17	11	140	2	7	38	19	444	4	6
18	12	152	2	1	39	20	464	2	3
19	12	164	4	1	40	20	484	4	5
20	12	176	7	2	41	20	504	7	5
21	12	188	5	3	42	26	530	7	6

Search Table

The search mechanism of the proposed LSA using alphabet table (given in Table 2) is shown in Table 3. In Table 3, SN indicates the serial number, the subsequent columns indexed by 1, 2, 3, 4 and 5(since the total number of cities to be visited $k = 6$ (i.e. $k - 1 = 5$), thus the length of the word L_r becomes 5 represent the letters in respective positions of a word. The next two columns respectively represent the value of the cost (Vc) and the lower bound (LB) for a partial word L_r . The subsequent columns indexed by R and CL represents the row and column indices of the letter respectively. Finally, the last column represented by *Remark* states that, if the partial word is said to be feasible then it is marked as *A* i.e. accept; otherwise *R* i.e. reject.

In Table 3, for each SN, a lower bound (LB) of a partial word isdetermined and checked whether it satisfies the bounds or not. A lower bound of a partial word is less than or equal to a trail value or IBFS and if it satisfies the feasibility then the partial word is called feasible, accepted and marked as *A*, otherwise the partial word is called infeasible, rejected and marked as *R* under *Remark* column. Similarly, a lower bound of a partial word is greater than a trail value or IBFS, rejected and is denoted by >*VT,R*. The feasible pattern of the initial solution $(TS = 27)$ found at 8th row of Table 3 is observed as $L_5 = (1, 2, 5, 6, 8)$. In order to find the improved solution backtracking is performed. On proceeding further for the next improvement, the solution is not improved, and thus it converged at the 18th row of Table 3. Therefore, the current solution becomes the final optimal solution i.e. 27 for the numerical example. Clearly, it is seen that the proposed LSA is capable of finding optimal solutions efficiently. An arbitrary feasible and optimal solution for the considered example is demonstrated in Fig.2.

Table 3: Search Table

S.N	1	2	3	4	5	Vc	LB	R	C	Rem
1	1					2	19	1	4	A
2		2				5	19	3	2	A
3			3			9	19	3	6	R
4			4			10	23	6	1	R
5			5			10	27	4	3	A
6				6		18	27	5	7	A
7					7	27	27	6	4	R
8					8	27	27	2	5	A, TS=27
9				7		19	29	6	4	>TS, R
10			6			13	31	5	7	>TS, R
11		3				6	20	3	6	A
12			4			11	20	6	1	R
13			5			11	28	4	3	>TS, R
14		4				7	29	6	1	>TS, R
15	2					3	25			A
16		3				7	25			R
17		4				8	30			>TS, R
18	3					4	31			>TS, R

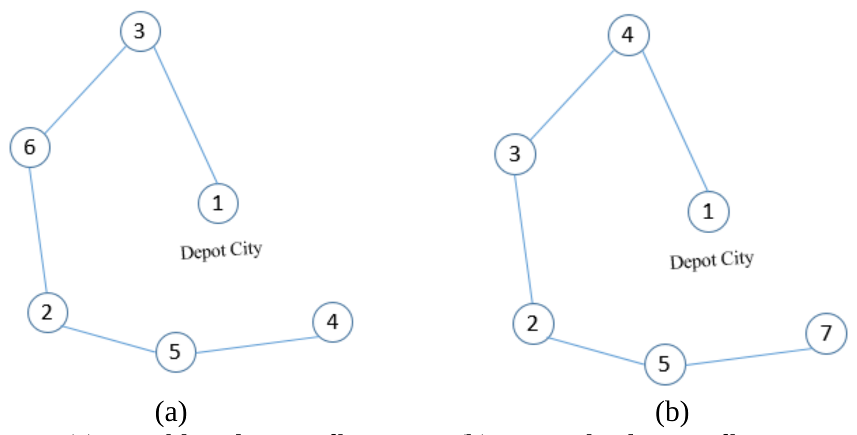


Fig.2(a) Feasible solution of*k*OTSPO, (b) Optimal solution of*k*OTSPO

5. Conclusions

In this paper, we considered a variant of open travelling salesman problemcalled an open *k*-city travelling salesman problem with ordered constraint (*k*OTSPO), motivated by the real world outsourcing scenarios in human resource allocation and routing problems. The *k*OTSPO has been formulatedas a zero-one integer programming. An efficient exact algorithm, the pattern recognition technique based Lexi-search algorithm (LSA) is developed for *k*OTSPO.To validate the proposed algorithm, a numerical example is illustrated. For the future consideration, one can extend the model *k*OTSPO with time windows, multiple depots and other practical variants etc.However, developing an efficient exact algorithm for such variants is still a challenging problem.

References

1. Chauhan, C., Gupta, R., & Pathak, K. (2012). Survey of methods of solving tsp along with its implementation using dynamic programming approach. *International journal of computer applications*, *52*(4).
2. Akhand, M. A. H., Ayon, S. I., Shahriyar, S. A., Siddique, N., & Adeli, H. (2020). Discrete spider monkey optimization for travelling salesman problem. *Applied Soft Computing*, *86*, 105887.
3. Halim, A. H., & Ismail, I. (2019). Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. *Archives of Computational Methods in Engineering*, *26*(2), 367-380.
4. Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(2), 231-247.
5. Battarra, M., Pessoa, A. A., Subramanian, A., & Uchoa, E. (2014). Exact algorithms for the traveling salesman problem with draft limits. *European Journal of Operational Research*, *235*(1), 115-128.
6. Hatamlou, A. (2018). Solving travelling salesman problem using black hole algorithm. *Soft Computing*, *22*(24), 8167-8175.
7. Bertsimas, D. J., &Simchi-Levi, D. (1996). A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations research*, *44*(2), 286-304.
8. Hacizade, U., & Kaya, I. (2018). Ga based traveling salesman problem solution and its application to transport routes optimization. *IFAC-Papers On Line*, *51*(30), 620-625.
9. Bharati, T. P., &Kalshetty, Y. R. (2016). A hybrid method to solve travelling salesman problem. *IJIRCCE*, *4*(8), 15148-15152.
10. Sathyan, A., Boone, N., & Cohen, K. (2015). Comparison of approximate approaches to solving the travelling salesman problem and its application to UAV swarming. *International Journal of Unmanned Systems Engineering.*,*3*(1), 1.
11. PS, A., & VASHISHT, V. (2013). Open Loop Travelling Salesman Problem using Genetic Algorithm. *Population*, *1*(1).
12. Wang, X., Liu, D., & Hou, M. (2013). A novel method for multiple depot and open paths, Multiple Traveling Salesmen Problem. In *2013 IEEE 11th international symposium on applied machine intelligence and informatics (SAMI)* (pp. 187-192). IEEE.

13. Sengupta, L., Mariescu-Istodor, R., &Fränti, P. (2019). Which Local Search Operator Works Best for the Open-Loop TSP?.*Applied Sciences*, *9*(19), 3985.
14. Abdulrazaq, M. B., Tahir, Y. S., Sha'aban, S., &Jibia, M. S. (2019, October). Polynomial Reduction of TSP to Freely Open-loop TSP. In *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)* (pp. 1-4). IEEE.
15. Chieng, H. H., & Wahid, N. (2014). A performance comparison of genetic algorithm's mutation operators in n-cities open loop travelling salesman problem. In *Recent Advances on Soft Computing and Data Mining* (pp. 89-97). Springer, Cham.
16. Thenepalle, J., & Singamsetty, P. (2019). An open close multiple travelling salesman problem with single depot. *Decision Science Letters*, *8*(2), 121-136.
17. Thenepalle, J. K., & Singamsetty, P. (2018). Bi-criteria travelling salesman subtour problem with time threshold. *The European Physical Journal Plus*, *133*(3), 1-15.
18. Singamsetty, P., Thenepalle, J., & Uruturu, B. (2021). Solving open travelling salesman subset-tour problem through a hybrid genetic algorithm. *Journal of Project Management*, *6*(4), 209-222.
19. Singamsetty, P., & Thenepalle, J. (2021). Designing optimal route for the distribution chain of a rural LPG delivery system. *International Journal of Industrial Engineering Computations*, *12*(2), 221-234.
20. Murthy M.S. (1976). A bulk transportation problem. *Opsearch,*13,143–155.