An Exact Algorithm for the Single-Depot Multiple Travelling Salesman Problem

Zakir Hussain Ahmed[a] and Ibrahim Al-Dayel[b]

Department of Mathematics and Statistics, College of Science, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Kingdom of Saudi Arabia,

Abstract: We consider the single-depot multiple travelling salesman Problem (MTSP) that is a generalization of the benchmark travelling salesman problem (TSP). The problem outlines that there are multiple salesmen *m* who should visit *n* cities so that each salesman must start from and end at single depot. The objective of the problem is to obtain the lowest total distance covered by all salesmen so that each city is visited only once by one salesman only. It is NP-hard, and it has numerous real-life applications. Though exact solutions can be found for small sized problem instances, yet there are certain circumstances where exact solutions are very essential. Hence, we propose to develop a lexisearch algorithm that uses path representation for a tour to find exact solution to the MTSP. The usefulness of the proposed exact algorithm is shown by comparing with an existing exact algorithm on various sized asymmetric instances from TSPLIB website with various number of salesmen. Experimental study shows the usefulness of the proposed algorithm. Finally, solutions to some symmetric TSPLIB instances are presented.

Keywords: *Multiple travelling salesman problem; NP-hard; Optimal; Exact; Bound; Lexisearch algorithm.*

1. Introduction

The travelling salesman problem (TSP) is a multidisciplinary benchmark problem that aims to obtain a least cost Hamiltonian circuit/cycle in a network. It may be defined as: Given *n* cities and distances among them. Starting from and ending at a single depot city, a salesman should visit all the cities exactly once so that the total distance (cost) covered by the salesman is minimum. Though this problem has been extensively studied by many researchers and proposed many useful algorithms to solve it, yet there are certain circumstances where more than one salesman is required. Hence, the multiple TSP (MTSP) is defined where all salesmen must start from and end at a single depot city. Each city, excluding the depot, is visited only once by one salesman only so that the total distance covered by all salesmen is minimum.

The MTSP has many practical applications, for example, school bus scheduling [1], interview scheduling [2], mission planning [3], crew scheduling [4], job scheduling [5], global navigation satellite surveying system networks [6], and vehicle scheduling and print press scheduling [7].

The problem has many variations such as single-depot, multi-depot, closed or open tours, etc [8]. Also, number of salesmen might be prefixed or permitted to vary, lower and upper bounds can be fixed on the number of salesmen as well as costs can be fixed related to the salesmen. We consider the single-depot closed tour problem that restricts all salesmen to begin from a single depot and finish their tours at the same depot.

Manuscript received September 5, 2020
Manuscript revised September 20, 2020

DOI: 10.22937/IJCSNS.2020.20.09.9

The MTSP is NP-hard [9], and there is no any known polynomial-time algorithm available for the solving the problem. Since, the problem is a TSP variant, the solution methods available for the usual TSP can also be applied to the MTSP. There are two kinds of algorithms available to solve the TSP and the MTSP such as heuristic and exact algorithms. Exact algorithms find exact solutions, whereas, heuristic algorithms obtain near exact solutions very quickly without assuring their optimality. As the problem size increases finding its exact solution using exact algorithm is too tough, if not impossible. However, small and medium sized problem instances can be solved easily, and there are some circumstances where finding exact optimal solutions are required.

In this study, we try to obtain exact solution using an exact method for the MTSP. The well-known exact methods for the TSP and associated problems are branch and bound [10] and lexisearch [11] approaches. The lexisearch approach is found better than the branch and bound approach for the TSP [12]. As MTSP is a generalization of TSP, so, we also propose to develop a lexisearch algorithm using path representation for a tour to obtain exact solution to the MTSP and compare the proposed algorithm against existing exact methods on some instances from TSPLIB [13] with various number of salesmen. Experimental study shows the usefulness of the proposed algorithm. Finally, solutions to some symmetric TSPLIB instances are presented.

This paper is prepared as: Section 2 reports a literature survey for the MTSP. The problem definition and its transformation are presented in Section 3, whereas Section 4 reports a lexisearch algorithm for finding exact solution to the MTSP. Experimental study for the proposed algorithm is presented in Section 5, whereas conclusions, discussions and future work are reported in Section 6.

2. Literature Survey

The MTSP is not well-studied like the usual TSP. Among the literatures for the problem, most of them are for heuristic algorithm. As this study proposes exact algorithm to solve the MTSP, we give a literature survey on exact algorithms. However, gravitational emulation local search [14], two-phase heuristic algorithm [15], genetic algorithms [16] are some well-known heuristics methods proposed for solving the problem.

As mentioned above, very few literatures are available for the exact algorithms. Laporte and Nobert [17] proposed the first exact method by relaxing some constraints of the problem to solve the problem directly, without transforming to the usual TSP.

Exact algorithm based on Branch-and-Bound method to find exact solution of the problem have been proposed in [18]. The algorithm uses a Lagrangean dual inside branch-and-bound algorithm and a sub-gradient technique to solve the dual function. Further, a greedy algorithm is applied to assess the points of the function. The method is tested on randomly created symmetric and Euclidean problem instances of sizes up to 100 and 59 respectively.

Gavish and Srikanth [19] reported a branch-and-bound method, where lower bounds are calculated using Lagrangean relaxation problem. Computational experiences have been reported for non-Euclidean and Euclidean problem instances. The results show that the Euclidean problem instances are harder than non-Euclidean ones. Comparative study show that their algorithm is better than the algorithms in [17, 18].

Husban [20] reported a mathematical model for the MTSP and then developed a branch-and-bound method for finding exact solution to the problem. The computational experience showed that for any problem size, the solution time decreases as the number of salesmen increases.

Gromicho et al. [21] reported a branch-and-bound algorithm using a quasi-assignment relaxation method. An additional bounding technique is also used to find strong lower bounds that improved the lower bound effectively. The proposed algorithm is applied on asymmetric instances up to size n=120 with number of salesmen ranging from 2 to 12. The proposed algorithm is found better than the standard branch-and-bound method.

Vali and Salimifard [22] formulated the problem using a constraint programming (CP) model and then used CP optimizer for finding exact solution to the problem. As reported, the proposed method performed very well compared to some existing algorithms.

Recently, Thenepallea and Singamsettya [23] introduced an open close MTSP with single depot (OCMTSP) in which salesmen are classified into internal and external ones, who are located at depot. The goal is to suggest the optimal tour so that beginning from the depot all salesmen visit the set of cities, each city by only one salesman with the additional condition that only internal salesmen must come back to the depot while the external salesmen are not required to return. A lexisearch algorithm, based on pattern recognition technique, was developed to find exact optimal solutions. As claimed, the algorithm provided sub-optimal and optimal solutions within reasonable solution times.

In this work, we propose to develop a lexisearch algorithm that uses path representation for a tour to obtain exact solution to the MTSP. The usefulness of the proposed lexisearch algorithm is shown by comparing with an existing exact algorithm [23] and CPLEX on various sized asymmetric instances from TSPLIB website with various number of salesmen. Experimental study shows the usefulness of the proposed algorithm. Finally, solutions to some symmetric TSPLIB instances are presented.

3. Problem Definition and its Transformation to the TSP

The MTSP is an utmost challenging multidisciplinary optimization problem. The problem can be stated as: Given *n* cities (nodes), labelled as *1, 2, …., n,* and *m* salesmen placed at a *single depot* in a network with the specified distance between cities i and j, d_{ij}, *i, j=1,2,...,n,* where the depot is *city 1* and the remaining cities are called *intermediate cities*. Starting from the depot, the problem is to visit all the intermediate cities by exactly one salesman and then return to the depot such that the tours have no common cities (except the depot city) and the total distance covered by all salesmen is minimized. In addition, if the MTSP involves capacity constraints along with each salesman, it converts to the capacitated vehicle routing problem (VRP). Clearly, if $m = 1$, the MTSP converts to the usual TSP. The problem can be expressed as an integer linear programming as below [18].

Define a binary variable, x_{ij}, is equal to 1 (one), when the edge (i, j) belongs to a tour, otherwise 0 (zero).

$$Minimize\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}, \quad d_{ij} = \infty\ for\ i = j$$

Subject to

$$(a)\ \sum_{i=1}^{n} x_{ij} = \begin{cases} m & if\ j = 1 \\ 1 & for\ all\ j\ in\ \{2,3,\ldots,n\} \end{cases}$$

$$(b)\ \sum_{j=1}^{n} x_{ij} = \begin{cases} m & if\ i = 1 \\ 1 & for\ all\ i\ in\ \{2,3,\ldots,n\} \end{cases}$$

$$(c)\ x_{ij} = (0,1)\ for\ all\ i,j\ in\ \{1,2,\ldots,n\}$$

$$(d)\ X = (x_{i,j}) \in S$$

The aim is to minimize the objective function, *z*, which is defined as the total distance covered by the salesmen. The equalities in (a) and (b) represent the number of salesmen allocated to permit multiple departures and arrivals (i.e. visits) to the depot from where the salesmen start, (c) confirms that the variables $x_{i,j}$ are integer, representing whether edge (i, j) is present in the tour. The set S in equality (d) denotes a set of constraints which exclude subtour solutions satisfying the assignment restrictions. The distance matrix may be represented as cost/ time matrix. The TSPs are divided into two types, based on the structure of distance matrix, such as symmetric and asymmetric. If $d_{ij} = d_{ji}$, $\forall\ i, j$, then it is symmetric, otherwise, asymmetric.

For n-city usual TSP, there are likely *(n-1)!* number of routes, and the computational effort is directly proportional to the problem size. So, it is too hard, if not impossible, to solve large sized problems. Additionally, the MTSP requires first to decide the cities allocated to a salesman, then to arrange the optimal order of the cities within each salesman's tour, so, it is harder than the usual TSP. Since, the usual TSP is NP-hard, hence, the MTSP is also NP-hard [9]. To our awareness, there is no polynomial-time algorithm present to solve the MTSP.

3.1. The TSP Transformation for the MTSP

The MTSP may be transformed into the TSP by considering only one salesman. Also, it may be treated as a reduction of the VRP by removing the capacity constraints [24]. The problem with *n* cities and *m* salesmen is transformed into the usual TSP with *n+m*-1 cities by adding *m-1* dummy depots (*n+1, …, n+m-1*), where large (infinite) distances are given to depot-to-depot distances to restrict such travels and zero distances are given between dummy depots and other cities. Solutions for the usual TSP and the MTSP are same [25]. According to research in [26] this conversion is not proper and results in a useless growth in the distance matrix. Further, they proposed a conversion in which the given matrix is augmented with *m−1* dummy columns and rows so that each dummy column and row is a copy of the first column and row of the give matrix. In [27], it is shown that the n-city and m-salesman asymmetric MTSP can be transformed into a usual (*n+m−1)*-city asymmetric TSP. A similar transformation for the symmetric MTSP into a (*n+m+4)-city* symmetric TSP is proposed in [28]. Also, a transformation into a (*n+m-1)-city* symmetric TSP is described in [29]. An improved conversion of a symmetric MTSP into a symmetric TSP is proposed in [30].

We are also going to transform the MTSP to the usual TSP by introducing *m-1* artificial depots. An example of the MTSP with *n = 7, m = 2* is shown in Fig. 1(a) and its transformation to the usual TSP is shown in Fig. 1(b). Also, the original distance matrix and the modified distance matrix with one artificial depot *city 8,* for a 7-city and 2-salesman problem, are showed in Tables 1 and 2 respectively.

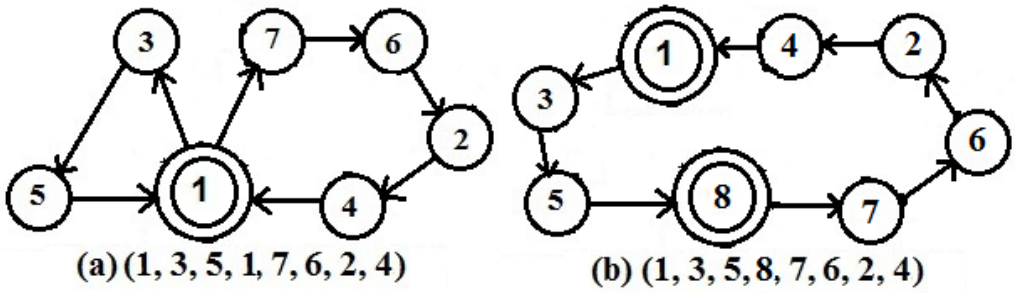

Figure 1. Example of solution of the MTSP and its transformation to the TSP with artificial city 8

Table 1. The distance matrix.

City	1	2	3	4	5	6	7
1	999	74	20	46	7	90	41
2	23	999	93	67	66	5	56
3	19	46	999	65	82	57	34
4	74	41	59	999	41	61	91
5	48	17	53	7	999	63	84
6	91	79	70	63	62	999	72
7	54	33	92	68	90	84	999

Table 2. The modified distance matrix with row minima (RM) and column minima (CM)

City	1	2	3	4	5	6	7	8	RM
1	999	74	20	46	7	90	41	999	7
2	23	999	93	67	66	5	56	23	5
3	19	46	999	65	82	57	34	19	19
4	74	41	59	999	41	61	91	74	41
5	48	17	53	7	999	63	84	48	7
6	91	79	70	63	62	999	72	91	62
7	54	33	92	68	90	84	999	54	33
8	999	74	20	46	7	90	41	999	7
CM	0	0	8	0	0	0	10	0	199

3.2. Bias Removal

The bias removal phase is found effective for the TSP [31] and clustered TSP [32]. As the MTSP is a generalization of the usual TSP and so, bias removal is supposed to be effective for the MTSP also. The bias removal procedure of the distance matrix can be stated as: first, each row-minimum is subtracted from its associated row elements. Next, apply the same procedure on the resultant matrix column-wise. The sum of the row-minimums and the consequent column-minimums is known as 'bias' of the given matrix [33]. Table 2 shows this bias computation. So, bias of the matrix = row-minimums + column-minimums = 181 + 18 = 199. The resultant distance matrix is a non-negative with minimum one zero in every row and in every column as showed in Table 3. In an assignment problem as well as in a TSP, if a constant is added or subtracted to all elements of a column (or row) in the matrix, then an allocation that reduces the total distance on one matrix also reduces the total distance on the other one. As the MTSP is a generalization of the TSP, thus, it is enough to solve the problem with regard to the reduced distance matrix.

Table 3. The reduced distance matrix.

City	1	2	3	4	5	6	7	8
1	992	67	5	39	0	83	24	992
2	18	994	80	62	61	0	41	18
3	0	27	972	46	63	38	5	0
4	33	0	10	958	0	20	40	33
5	41	10	38	0	992	56	67	41
6	29	17	0	1	0	937	0	29
7	21	0	51	35	57	51	956	21
8	992	67	5	39	0	83	24	992

3.3. Alphabet Table

Alphabet matrix, denoted by *A=[a(i, j)]*, is an *n*×*n* square matrix created by locations of elements of $n \times n$ reduced distance matrix, $D' = [d'_{ij}]$, after arranging the elements in non-decreasing order. Alphabet table, denoted by $[a(i,j) - d'_{i,a(i,j)}]$, is a mixture of elements (cities) of the matrix A and their distances in the reduced distance matrix [34]. Table 4 represents the alphabet table created for the reduced matrix given in Table 3, where C is a city and D is its distance from the corresponding city in the 1st column.

Table 4. The alphabet table.

City	C-D	C-D	C-D	C-D	C-D	C-D	C-D	C-D
1	5-0	3-5	7-24	4-39	2-67	6-83	1-992	8-992
2	6-0	1-18	8-18	7-41	5-61	4-62	3-80	2-994
3	1-0	8-0	7-5	2-27	6-38	4-46	5-63	3-972
4	2-0	5-0	3-10	6-20	1-33	8-33	7-40	4-958
5	4-0	2-10	3-38	1-41	8-41	6-56	7-67	5-992
6	3-0	5-0	7-0	4-1	2-17	1-29	8-29	6-937
7	2-0	1-21	8-21	4-35	3-51	6-51	5-57	7-956
8	5-0	3-5	7-24	4-39	2-67	6-83	1-992	8-992

4. A Lexisearch Algorithm for the MTSP

The lexisearch algorithm is effectively developed for several complex problems [31-37], where all feasible solutions are organized in an order like words in any dictionary, in a way that a partial word denotes a block of words and the block leader. Lower bounds are calculated for the objective function on these blocks which are compared with current 'best solution'. In this block, if no better solution (word) than the present 'best solution' is found, then jump over from the present block into the next block. But, if the bound shows an opportunity to have better solution in this block, then go to its subblock by joining the present leader with the suitable letter and then calculate its lower bound.

4.1. Block Leader

A partial word (incomplete tour) containing some cities is called the block leader of words. For the MTSP, each city, including dummy depot, is treated as a letter in any alphabet. Therefore, all words (solutions) in the dictionary is subdivided into blocks. A word block B having a three-length leader (α_0, α_1, α_2) contains all words starting with the words of these three letters (α_0, α_1, α_2) as a string. The block A having the two-length leader (α_0, α_1) is the next superblock of the block B including the block B as its subblock. Next, the block C having a four-length leader (α_0, α_1, α_2, α_k) is one of the subblocks of block B that contains several four-length subblocks, one for a letter α_k. Block B is the next superblock of the block C [33].

4.2. Lower Bound

Calculating and finding a compact lower bound for a block leader on the objective function for the MTSP is very hard. Therefore, the lower bound used for the TSP [31] and clustered TSP [32] is considered here, that is stated as follows. Let the incomplete tour be ($1=\alpha_0$, α_1, α_2) and *city* α_3 be chosen for joining. Before joining, bound for the block leader (α_0, α_1, α_2, α_3) is calculated. For that, computation starts from second row and continues to the last row of the *alphabet table* and sum up the distances of first legitimate cities (the cities which are not available in the present tour), including *city 1*, in every row, except α_1-th and α_2-th rows. The calculated total distance is the bound for this leader (α_0, α_1, α_2, α_3).

4.3. The Proposed algorithm

In general, there are two methods of expressing salesman's tour in lexisearch algorithm (LSA), which are adjacency and path representations. In [23], for solving an open close MTSP with single depot (OCMTSP) adjacency representation is used. We are using the path representation for solving the MTSP. It is applied on the simple TSP [31] and with precedence constraints [37], and clustered TSP [32], and found very effective results.

Suppose there are n cities and m salesmen in the network, and $D=[d_{ij}]$ is the n-th order distance matrix. After adding $m-1$ dummy depot cities, the number of cities becomes $n+m-1$ and so, new size of the problem becomes, $n = n+m-1$. Accordingly, the given matrix is modified. The proposed LSA for the MTSP is a modification of the LSA for the TSP [31] which is stated as follows.

Step 0: Calculate *bias* of the given distance matrix, construct reduced distance matrix and then the *alphabet table* based on the *reduced matrix*. Fix *best solution value (BS) = L (very large number)*. As *city 1* is the depot city, calculation is started from first row of the *alphabet table*. Set *k=1, tour value (Sol) = 0,* and then go to step 1.

Step 1: Go to k^{th} element of the first row (say *city q*) with distance as *present city distance (Dist)*. If *(Sol + Dist) < BS*, go to step 2 *else,* go to step 9.

Step 2: If the *city q* forms a subtour or the present city and *city q* are both depot cities, drop it, set *k=k+1* and go to step 7 *else,* go to step 3.

Step 3: If all cities of the transformed problem are visited, add the edge connecting the *city q* to *city 1*, compute *Sol* and go to step 4 *else,* go to step 5.

Step 4: If *(Sol ≥ BS)* then go to step 9 *else,* replace *BS = Sol* and go to step 9.

Step 5: Calculate the *lower bound (LB)* of the present leader on the objective function and go to *step 6*.

Step 6: If *(Sol + Dist + LB) ≥ BS* then drop the *city q*, set *k=k+1* and go to step 7; *else,* accept the *city q*, compute *Sol* and go to step 8.

Step 7: If (*k < n)*, go to step 1 *else,* go to step 9.

Step 8: Go to subblock, i.e., go to q^{th} row, put *k = 1* and go to step 1.

Step 9: Jump this block by dropping the *city q* and return to the previous city in the tour (say, *city p*), i.e., go to the p^{th} row of the *alphabet table* and set *k = k + 1* where *k* was the index of the last *visited city* in that row. If city *p = 1* and *k = n*, go to step 10 *else,* go to step 7.

Step 10: Now *BS* is the optimal solution value with regard to the reduced distance matrix, and *BS = BS + bias* is the optimal solution value with regard to the original distance matrix and go to step 11.

Step 11: Current word is the optimal tour sequence with regard to the modified distance matrix. Then tours of the salesmen with regard to the original problem is found and then *stop.*

4.4. Illustration of the algorithm

Steps of the proposed LSA is explained using a 7-city and 2-salesman problem instance with the given distance matrix in Table 1. The modified distance matrix, reduced distance matrix and *alphabet table* are shown in the corresponding Tables 2, 3 and 4. The flow of LSA at several steps is presented in Table 5, where $5\rightarrow4_{(0)}$ denotes present city distance $d_{54} = 0$, and *(0) +5, GS* denote (0) is the tour value

that includes present city distance, 5 is the lower bound and conclusion is GS. The symbols used in the table are as follows:

GS: Go to subblock, i.e., join first *legal* letter to the present leader. GS for *ab* is *abc* as augmented leader.

JB: Jump over the block, i.e., jump over the next block of same length by replacing its last letter by the next letter in the alphabet table. JB for *ab* is *ac*.

JO: Jump out to the immediate higher order block by dropping its last letter and then jumping out the block. JO for *abcd* is *abd*.

Let *BS = 999* and *Sol = 0*. Consider the *alphabet table* and start from its *1st* row. We have *a(1,1) = 5* with *Dist* = $d_{15} = 0$ which does not form subtour and *(Sol + Dist) < BS*. Next, LB for the leader (1, 5) is calculated, which guides us whether *city 5* can be accepted.

$$LB = d_{2,a(2,1)} + d_{3,a(3,1)} + d_{4,a(4,1)} + d_{5,a(5,1)} + d_{6,a(6,1)} + d_{7,a(7,1)} + d_{8,a(8,2)}$$
$$= d_{2,6} + d_{3,1} + d_{4,2} + d_{5,4} + d_{6,3} + d_{7,2} + d_{8,3}$$
$$= 0 + 0 + 0 + 0 + 0 + 0 + 5 = 5.$$

Table 5. Search Table

1(α_1)→α_2	**α_2→α_3**	**α_3→α_4**	**α_4→α_5**	**α_5→α_6**	**α_6→α_7**	**α_7→α_8**	**α_8→1(α_1)**
1→ 5(0)	5→4(0)	4→2(0)	2→6(0)	6→3(0)	3→8(0)	8→7(24)	7→1(21)
(0) +5, GS	(0) +5, GS	(0) +26, GS	(0) +26, GS	(0) +45, GS	(0) +45, GS	(24) +21, GS	BS=45, JO
					3→7(5)		
					(5) +1013, JB, JO		
				6→7(0)	7→8(21)	8→3(5)	3→1(0)
				(0) +26, GS	(21) +5, GS	(26) +0, GS	BS=26, JO
			2→8(18)				
			(18) +26, JB				
			2→7(41), JO				
		4→3(10)					
		(10) +24, JB					
		4→6(20)					
		(20) +23, JB,JO					
	5→2(10)						
	(10) +36, JB						
	5→3(38), JO						
1→3(5)	3→8(0)	8→5(0)	5→4(0)	4→2(0)			
(5) +0, GS	(5) +0, GS	(5) +0, GS	(5) +0, GS	(5) +21, JB			
				4→6(20)			
				(25) +18, JB, JO			
			5→2(10)				
			(15) +41, JB, JO				
		8→7(24), JO					
	3→7(5)	7→2(0)	2→6(0)	6→5(0)			
	(10) +0, GS	(10) +0, GS	(10) +0, GS	(10) +72, JB			
				6→4(1)			
				(11) +41, JB, JO			
	3→2(27), JO						
1→7(24)	7→2(0)	2→6(0)	6→3(0)	3→8(0)	8→5(0)		
(24) +0, GS	(24) +0, GS	(24) +0, GS	(24) +0, GS	(24) +0, GS	(24) +33, JB		
					8→3(5), JO		
			6→5(0)				
			(24) +15, JB				
			6→4(1)				
			(25) +38, JB,JO				
1→4(39), STOP							

As *(Sol + Dist + LB) = 5 + 0 + 0 = 5 < BS*, the *city 5* is accepted and the tour becomes *{1→5}* having *Sol = Sol + Dist = 0 + 0 = 0*. Now go to 5th row of *alphabet table* and choose city '*a(5,1) = 4'* as it is legitimate city. So, *Dist* = $d_{54} = 0$ and *(Sol + Dist) < BS*. Lower bound for the leader *(1, 5, 4)* is computed as follows.

$$LB = d_{2,a(2,1)} + d_{3,a(3,1)} + d_{4,a(4,1)} + d_{6,a(6,1)} + d_{7,a(7,1)} + d_{8,a(8,2)}$$

$$= d_{2,6} + d_{3,1} + d_{4,2} + d_{6,3} + d_{7,2} + d_{8,3}$$
$$= 0 + 0 + 0 + 0 + 0 + 5 = 5.$$

As *(Sol + Dist + LB) = 5 + 0 + 0 = 5 < BS*, the *city 4* is accepted and the tour becomes *{1→5→4}* having *Sol = Sol + Dist = 0 + 0 = 0*. Now go to 4th row of *alphabet table* and choose the city '*a(4,1) = 2'*. So, *Dist* = $d_{42} = 0$ and *(Sol + Dist) < BS*. Lower bound for *(1, 5, 4, 2)* is

$$LB = d_{2,a(2,1)} + d_{3,a(3,1)} + d_{6,a(6,1)} + d_{7,a(7,2)} + d_{8,a(8,2)}$$
$$= d_{2,6} + d_{3,1} + d_{6,3} + d_{7,2} + d_{8,3}$$
$$= 0 + 0 + 0 + 21 + 5 = 26.$$

As *(Sol + Dist + LB) = 26 + 0 + 0 = 26 < BS*, the *city 2* is accepted and the tour becomes *{1→5→4→2}* having *Sol = Sol + Dist = 0 + 0 = 0*. Now go to 2rd row of *alphabet table* and choose the city '*a(2,1) = 6'* having *Dist* = $d_{26} = 0$. As *(Sol + Dist) < BS*, then bound for *(1, 5, 4, 2, 6)* is computed.

$$LB = d_{3,a(3,1)} + d_{6,a(6,1)} + d_{7,a(7,2)} + d_{8,a(8,2)}$$
$$= d_{3,1} + d_{6,3} + d_{7,1} + d_{8,3}$$
$$= 0 + 0 + 21 + 5 = 26.$$

Since *(Sol + Dist + LB) = 26 + 0 + 0 = 26 < BS*, the tour becomes *{1→5→4→2→6}* having *Sol = Sol + Dist = 0 + 0 = 0*. Then choose the city '*a(6,1) = 3'* having *Dist* = $d_{63} = 0$. As *(Sol + Dist) < BS*, bound for *(1, 5, 4, 2, 6, 3)* is computed.

$$LB = d_{3,a(3,1)} + d_{7,a(7,2)} + d_{8,a(8,3)} = d_{3,1} + d_{7,1} + d_{8,7}$$
$$= 0 + 21 + 24 = 45$$

As *(Sol + Dist + LB) = 45 + 0 + 0 = 45 < BS*, the tour becomes *{1→5→4→2→6→3}* having *Sol = Sol + Dist = 0 + 0 = 0*. Next, choose the city '*a(3,2) = 8'* having *Dist* = $d_{38} = 0$. As *(Sol + Dist) < BS*, bound for *(1, 5, 4, 2, 6, 3, 8)* is computed.

$$LB = d_{7,a(7,2)} + d_{8,a(8,3)} = d_{7,1} + d_{8,7} = 21 + 24 = 45.$$

Since *(Sol + Dist + LB) = 45 + 0 + 0 = 45 < BS*, the tour becomes *{1→5→4→2→6→3→8}* having *Sol = Sol + Dist = 0 + 0 = 0*. Next, choose the city '*a(8,3) = 7'* having *Dist* = $d_{87} = 24$. As *(Sol + Dist) < BS*, bound for *(1, 5, 4, 2, 6, 3, 8, 7)* is computed.

$$LB = c_{7,a(7,2)} = c_{7,1} = 21.$$

Since *(Sol + Dist + LB) = 21 + 0 + 24 = 45 < BS*, the tour becomes *{1→5→4→2→6→3→8→7}* having *Sol = Sol + Dist = 0 + 24 = 24*. As the present city is the final city in this tour, thus, return to the depot (*city 1*) having *Dist* = $d_{71} = 21$. As *(Sol + Dist) < BS*, the first complete tour becomes *{1→5→4→2→6→3→8→7→1}* having *Sol = 45*. Now replace *BS = 45* and jump out to the next super block, i.e., *{1→5→4→2→6→3}* having *Sol = 0*. Now try to find another complete tour with cheaper tour value.

Continuing this way, the optimal tour *{1→5→4→2→6→7→8→3→1}* having *BS = 26* is found. Therefore, the optimal tour value with regard to the given matrix, BS = *bias + BS = 199 + 26 = 255*. Our preliminary study shows that our algorithm obtains final solution quickly, but for verifying its optimality, it takes long time. So, if there is no any improvement of the solution is found within 180 seconds then jump out to the next super block.

5. Computational Experience

Our LSA is encoded in Visual C++. In order to establish the efficiency of LSA, computational experience is carried out on a set of benchmark TSPLIB instances [13] with different number of salesmen and run on a Laptop with i3-3217U CPU@1.80 GHz and 4 GB RAM under MS Windows 7. Four asymmetric TSPLIB instances, that is, br17, ftv33, ftv35 and ftv38, each one with 2, 3 and 4 salesmen, are considered for the comparative study. A comparative study of our proposed algorithm (LSA) against CPLEX and Benders method reported in [38], and an existing lexisearch algorithm that used adjacency representation (LSA2) [23] is showed in Table 6. Solutions obtained by these algorithms as well as best known solution reported (within brackets) for 1-salesman instances in TSPLIB and computational time (in seconds) whenever the final/best solution is obtained for first time (FTime) and the complete time (CTime) by our proposed LSA, are reported.

It is seen from the Table 6 that the obtained solutions for br17 and ftv33 (with 2, 3 and 4 salesman), and ftv35 (with 2 salesmen) using all four algorithms are same. For ftv35 (with 3 salesmen), obtained solutions by Benders method, LSA2 and our proposed LSA are same 1511, while solution obtained by CPLEX is 1541, a higher value. So, the proposed algorithm (LSA) finds the exact solution.

For ftv35 (with 4 salesmen), solutions obtained by Benders method and our LSA are same 1551, whereas, solution by LSA2 is 1532. Now, the question is that whether 1551 or 1532 is optimal. The answer is as follows. The differences between optimal solutions for ftv35 with 2 salesmen and 1 salesman is (1489-1473=) 16, and for ftv35 with 3 salesman and 2 salesmen is (1511-1489=) 22. It seems that as number of salesmen increases these differences also increase. So, the optimal solution for the ftv35 with 4 salesmen must be at least 1511 + 22 = 1533, and so, 1532 cannot be optimal solution. Hence, our solution 1551 is optimal.

For ftv38 with 2 salesmen, solutions obtained by Benders method and LSA2 are same 1505, whereas, solutions by CPLEX and our proposed LSA are 1551 and 1546 respectively. Now, as the solution of the instance ftv38 with one salesman is 1530, so the solution of ftv38 with 2 salesmen cannot be less than 1530, so, 1505 cannot be optimal. Also, between 1551 and 1546, the solution 1546 is better. Hence, our solution is the optimal.

For ftv38 with 3 salesmen, solutions obtained by Benders method and LSA2 are same 1521, whereas, solutions by CPLEX and our proposed LSA are same 1567. For this case also, as the solution of the instance ftv38 with one salesman is 1530, so the solution of ftv38 with 3 salesmen cannot be less than 1530, so, 1521 cannot be optimal. Hence, our solution 1567 is the optimal.

For ftv38 with 4 salesmen, solutions obtained by Benders method, LSA2 and our LSA are same 1546, 1532 and 1608 respectively. For this case also, as the solution of the instance ftv38 with 3 salesmen is 1567, so the solution of ftv38 with 4 salesmen cannot be less than 1567, so, solutions 1532 and 1546 cannot be optimal. Hence, our solution 1608 is the optimal.

Table 6: A comparative study of different algorithms on asymmetric TSPLIB instances

Instance	n	m	Solution by			Results by our proposed LSA		
			CPLEX	Benders method	LSA2	Solution	FTime	CTime
br17	17	2	39	39	39	39	6.94	48.94
(39)		3	42	42	42	42	0.00	45.01
		4	47	47	47	47	0.13	45.13
ftv33	34	2	1302	1302	1302	1302	171.83	211.37
(1286)		3	1328	1328	1328	1328	288.14	557.89
		4	1367	1367	1367	1367	93.41	768.85
ftv35	36	2	1489	1489	1489	1489	88.69	345.46
(1473)		3	1541	1511	1511	1511	297.66	864.74
		4	---	1551	1532*	1551	519.32	1189.49
ftv38	39	2	1551	1505*	1505*	1546	369.72	908.37
(1530)		3	1567	1521*	1521*	1567	340.86	1751.81
		4	--	1546*	1532*	1608	607.37	2230.53
Average							**232.01**	**747.30**

*Note: Solutions are not exact optimal

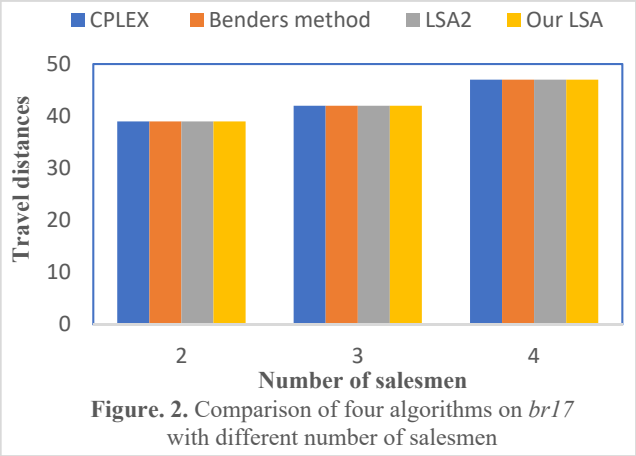

Figure. 2. Comparison of four algorithms on *br17* with different number of salesmen

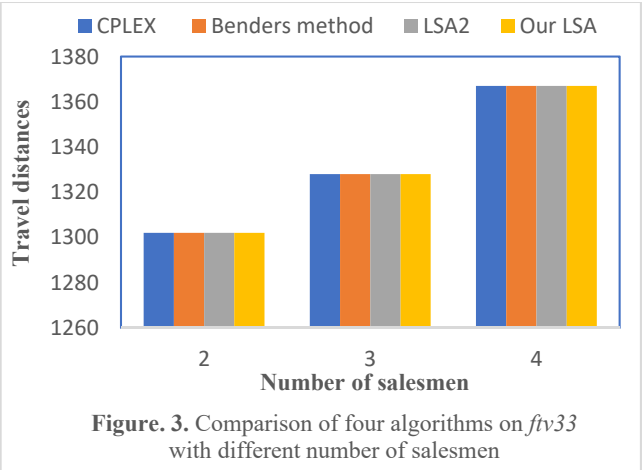

Figure. 3. Comparison of four algorithms on *ftv33* with different number of salesmen

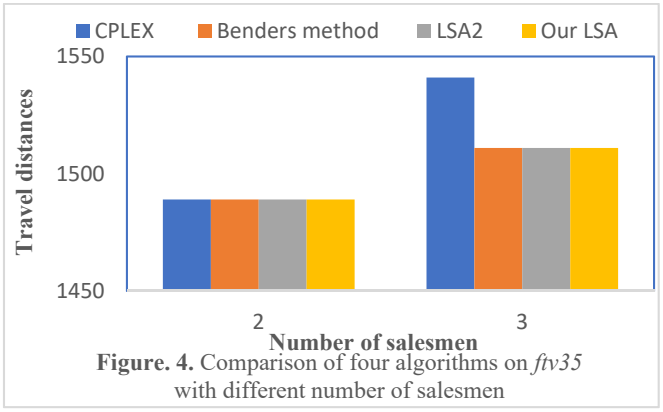

Figure. 4. Comparison of four algorithms on *ftv35* with different number of salesmen

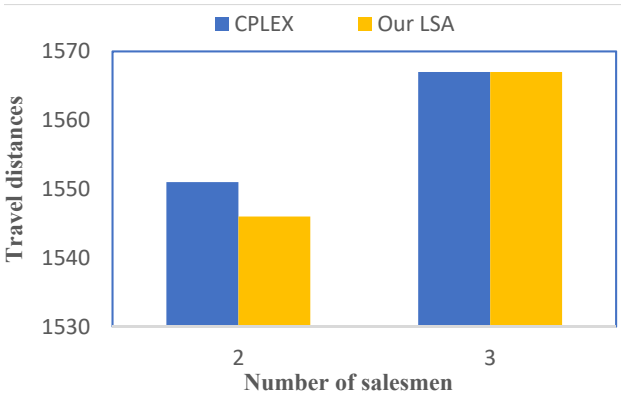

Figure. 5. Comparison of two algorithms on *ftv38* with different number of salesmen

Additionally, to visualize effectiveness of our proposed algorithm, LSA, against CPLEX, Benders and LSA2 algorithms on the above considered four TSPLIB instances, results are depicted in line diagrams. The total travel distances for br17, ftv33, ftv35, and ftv38 with different number of salesmen are showed in Figures 2, 3, 4 and 5 respectively. The Figures 2 and 3 show that all the four algorithms obtained same solutions on br17 and ftv33 with 2, 3, and 4 salesmen respectively. The Figure 4 shows that all algorithms obtained same solutions on ftv35 with 2 salesmen**,** but for ftv35 with 3 salesmen, except for CPLEX, all other three algorithms obtained same solution, and CPLEX could not obtain optimal solution. It is seen from Figure 5 that for ftv38 with 2 salesmen, the proposed LSA obtained better solution than that by CPLEX. However, for ftv38 with 3 salesmen, our LSA and CPLEX obtained same solution. It is already found that for ftv38 with 2, 3 and 4 salesmen, Benders method and LSA2 obtained solutions less than optimal solutions, hence, their results are not depicted in Figure 5. It is clear from these figures that our proposed LSA obtained better solution than solutions by CPLEX, Benders method and LSA2.

On the other hand, as regard the CTime, using the proposed LSA, it is observed that for any instance when the number of

salesmen increases the computational time increases. It shows that the structure of multi-salesman problem instance is getting harder as the number of salesmen increases. However, for the instance br17 with 2-salesman computational time is 48.94 seconds while with three and four salesmen computational times are 45.01 and 45.13 seconds respectively. It means that the structures of br17 with 3 and 4 salesmen are less complex than br17 with 2-salesmen. Generally, a LSA first obtains a solution and then confirms its optimality, meaning that the remaining subproblems are rejected [31, 34]. On average computational time, the Table 6 reports that LSA obtains a solution within a maximum 31% of complete computational time. So, LSA spends minimum 69% of complete computational time on verifying the solutions. Thus, for these asymmetric TSPLIB instances, LSA spends less time on obtaining solution, and hence, a large amount of subproblems are thrown.

Table 7 reports solutions for few symmetric instances from TSPLIB of sizes varies from 14 to 42 with 1, 2, 3 and 4 salesmen. For dantzig42, solutions obtained within two hours of computational time are reported. The table shows that for any instance when the number of salesmen increases solutions as well as computational time (CTime) also increase. For most of the instances, it is found that computational time for solving an one salesman instance is lesser than its corresponding multi-salesmen instances. Thus, for symmetric instances, the MTSP is harder than the usual TSP. However, for ulysses22 and bayg29 with single salesman LSA spend more time than with two salesmen. Generally, for the MTSP instances when number of salesmen increases time also increases. On average, LSA obtains a best solution within maximum 57% of the complete time for the symmetric instances. That means, LSA spends minimum 43% of complete computational time on verifying the solutions. Overall, LSA expends less time on obtaining solution while many subproblems are rejected.

Table 7. Results by LSA on few symmetric TSPLIB instances

Instance	n	m	Sol	FTime	CTime	Instance	n	m	Sol	FTime	CTime
burma14	14	1	3323	0.00	0.01	ulysses16	16	1	6859	0.19	0.67
(3323)		2	3372	0.01	0.02	(6859)		2	6960	0.67	2.01
		3	3547	0.00	0.12			3	7112	1.03	8.63
		4	3731	0.00	0.67			4	7335	3.97	33.28
gr17	17	1	2085	0.00	0.60	gr21	21	1	2707	0.01	0.04
(2085)		2	2188	0.00	1.21	(2707)		2	2839	0.02	0.09
		3	2322	0.63	7.92			3	3022	0.21	1.01
		4	2504	5.85	56.95			4	3233	2.18	14.65
ulysses22	22	1	7013	81.88	123.99	gr24	24	1	1272	30.56	44.95
(7013)		2	7107	60.76	111.74	(1272)		2	1389	68.77	116.61
		3	7259	95.15	173.62			3	1523	358.28	441.59
		4	7519	42.82	417.26			4	1695	491.78	548.46
fri26	26	1	937	28.35	76.93	bayg29	29	1	1610	72.64	105.94
(937)		2	1070	152.62	194.07	(1610)		2	1652	12.55	64.26
		3	1214	320.41	335.32			3	1718	45.53	252.99
		4	1375	701.56	775.02			4	1792	205.07	492.76
bays29	29	1	2020	0.20	77.06	*dantzig42	42	1	699	446.49	---
(2020)		2	2074	199.79	283.56	(699)		2	701	518.94	---
		3	2139	143.88	455.43			3	703	632.17	---
		4	2244	323.94	824.49			4	710	895.71	---
Average FTime			**= 95.87 (*except dantzig42)**			**Average CTime**			**= 167.89**		

6. Conclusions and Future Research

In this work, a lexisearch algorithm (LSA) using path representation is presented to obtain exact solution to the single-depot MTSP. Steps of LSA are illustrated through an example of 7-city and 2-salesman problem instance. Next, a comparative study of the proposed LSA is carried with three existing exact algorithms, namely, CPLEX and Benders method reported in [38], and another lexisearch algorithm (LSA2) in [23] on four asymmetric TSPLIB instances with 2, 3, and 4 salesmen. The proposed LSA could obtain exact solution for all instances, other algorithms could not obtain exact solutions for all instances. Finally, solutions to few symmetric instances from TSPLIB of sizes varies from 14 to 42 with 1, 2, 3 and 4 salesmen are reported.

Computational experience shows that for most of the instances, computational efforts for solving a single salesman instance is less than its corresponding multi-salesman problem instances. For any problem instance with multi-salesman, when the number of salesmen increases the solutions as well the computational times also increase. It is found that for large sized problem instances, proposed LSA is not effective. However, this study suggests an alternate exact algorithm to apply on other complex optimization problems.

It is also found that LSA first obtains a solution very quickly while spends long computational time to verify its optimality. So, there must be some technique to verify the obtained solution for its optimality quickly. Hence, an effective data-guided technique could be suggested that might decrease computational effort for verifying the optimality of the quickly obtained solution and might obtain exact solutions

for other instances as shown for the TSP [31], bottleneck TSP [39] and quadratic assignment problem [40], which is under next investigation.

Acknowledgment

The authors are very grateful to the honorable anonymous reviewers for their constructive comments and suggestions which helped the authors to improve this paper.

References

[1]. R.D. Angel, W.L. Caudle, R. Noonan, and A. Whinston, Computer assisted school bus scheduling, Management Science 18 (1972) pp. 279–288.

[2]. K.C. Gilbert, and R.B. Hofstra, A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem, Decision Sciences 23(1992) pp. 250–259.

[3]. B.L. Brumitt, and A. Stentz, Dynamic mission planning for multiple mobile robots, Proceedings of IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, vol. 3 (1996) pp. 2396-2401.

[4]. T. Zhang, W.A. Gruver, and M.H. Smith, Team scheduling by genetic search, Proceedings of the second international conference on intelligent processing and manufacturing of materials, vol. 2 (1999) pp. 839–844.

[5]. A.E. Carter, and C.T. Ragsdale, Scheduling pre-printed newspaper advertising inserts using genetic algorithms, Omega 30 (2002) pp. 415–421.

[6]. H.A. Saleh, and R. Chelouah, The design of the global navigation satellite system surveying networks using genetic algorithms, Engineering Applications of Artificial Intelligence 17 (2004) pp. 111–122.

[7]. 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 175 (2006) pp. 245–257.

[8]. T. Bektas, The multiple traveling salesman problem: An overview of formulations and solution procedures, Omega 34 (2006) pp. 209–219.

[9]. M. R. Garey, and D.S. Johnson, A guide to the theory of NP-completeness, computers and intractability, W. H. Freeman & Co., New York, NY, USA, 1990.

[10]. J.D.C. Little, K.G. Murthy, D.W. Sweeny, and C. Karel, An algorithm for the travelling salesman problem, Operations Research 11 (1963) pp. 972-989.

[11]. S.N.N. Pandit, The Loading Problem. Operations Research 11 (1962) pp. 639-646.

[12]. S.N.N. Pandit and K. Srinivas, A lexisearch algorithm for the traveling salesman problem, Proceedings of the IEEE International joint conference on neural networks 3 (1991) pp. 2521-2527.

[13]. G. Reinelt, 1991, TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[14]. R.A. Shokouhi, M. Farahnaz, K. Hengameh, and A.R. Hosseinabadi, Solving multiple traveling salesman problem using the gravitational emulation local search algorithm, Applied Mathematics 9(2) (2015) pp. 699–709.

[15]. X. Xu, H. Yuan, M. Liptrott, and M. Trovati, Two phase heuristic algorithm for the multiple-travelling salesman problem, Soft Comput 22 (2018) pp. 6567–6581.

[16]. M.A. Al-Furhud, and Z.H. Ahmed, Genetic algorithms for the multiple travelling salesman problem, International Journal of Advanced Computer Science and Applications (IJACSA) 11(7) (2020) pp. 553–560.

[17]. G. Laporte, Y. Nobert, A cutting planes algorithm for the *m*-salesmen problem, Journal of the Operational Research Society 31 (1980) pp. 1017–1023.

[18]. A.I. Ali, and J.L. Kennington, The asymmetric *m*-traveling salesmen problem: a duality based branch-and-bound algorithm, Discrete Applied Mathematics 13 (1986) pp. 259–276.

[19]. B. Gavish, and K. Srikanth, An optimal solution method for large-scale multiple traveling salesman problems, Operations Research 34(5) (1986) pp. 698–717.

[20]. A. Husban, An exact solution method for the MTSP, The Journal of the Operational Research Society 40(5) (1989) pp. 461-469.

[21]. J. Gromicho, J. Paixão, and I. Branco, Exact solution of multiple traveling salesman problems. In: Mustafa Akgül, et al., editors. Combinatorial optimization. NATO ASI Series, vol. F82. Berlin: Springer; 1992. pp. 291–92.

[22]. M. Vali, and K. Salimifard, A constraint programming approach for solving multiple traveling salesman problem, In: The Sixteenth International Workshop on Constraint Modelling and Reformulation, 28 August 2017, pp. 1-17.

[23]. J.K. Thenepallea, and P. Singamsettya, An open close multiple travelling salesman problem with single depot, Decision Science Letters 8 (2019) pp. 121–136.

[24]. H.C. Lau, T.M. Chan, and W.T. Tsui, Application of genetic algorithms to solve the multi-depot vehicle routing problem, IEEE Transaction Automatic Science and Engineering 7 (2010) pp. 383–392.

[25]. S. Gorenstein, Printing press scheduling for multi-edition periodicals, Management Science 16(6):B (1970) pp. 373–83.

[26]. J.A. Sveska, and V.E. Huckfeldt, Computational experience with an *m*-salesman traveling salesman algorithm, Management Science 19 (1973) pp.790-799.

[27]. M. Bellmore, and S. Hong, Transformation of multisalesmen problem to the standard traveling salesman problem, Journal of Association for Computing Machinery 21 (1974) pp. 500–504.

[28]. S. Hong, and M.W. Padberg, A note on the symmetric multiple traveling salesman problem with fixed charges, Operations Research 25 (1977) pp. 871–874.

[29]. M.R. Rao, A note on multiple travelling salesmen problem, Operations Research 28 (1980) pp. 628–632.

[30]. R. Jonker, and T. Volgenant, Technical Note—An Improved Transformation of the Symmetric Multiple Traveling Salesman Problem, Operations Research 36(1) (1988) pp. 163-167.

[31]. Z.H. Ahmed, A data-guided lexisearch algorithm for the asymmetric traveling salesman problem, Mathematical Problems in Engineering (2011) Article ID 750968, 18 pages.

[32]. Z.H. Ahmed, An exact algorithm for the clustered traveling salesman problem, Opsearch 50(2) (2013) pp. 215-228.

[33]. Z.H. Ahmed, A sequential constructive sampling and related approaches to combinatorial optimization, PhD Thesis, Tezpur University, Assam, India, 2000.

[34]. Z.H. Ahmed, A lexisearch algorithm for the bottleneck traveling salesman problem, International Journal of Computer Science and Security 3(6) (2010) pp. 569-577.

[35]. Z.H. Ahmed, A new reformulation and an exact algorithm for the quadratic assignment problem, Indian Journal of Science and Technology 6(4) (2013) pp. 4368-4377.

[36]. Z.H. Ahmed, A lexisearch algorithm for the distance-constrained vehicle routing problem, International Journal of Mathematical and Computational Methods 1 (2016) pp. 165-174.

[37]. Z.H. Ahmed, and S.N.N. Pandit, The travelling salesman problem with precedence constraints, Opsearch 38 (2001) pp. 299-318.

[38]. C. Changdar, R.K. Pal, and G.S. Mahapatra, A genetic ant colony optimization based algorithm for solid multiple travelling salesmen problem in fuzzy rough environment, Soft Computing 21(16) (2017) pp. 4661-4675.

[39]. Z.H. Ahmed, A data-guided lexisearch algorithm for the bottleneck traveling salesman problem, International Journal Operations Research 12 (2011) pp. 20-33.

[40]. Z.H. Ahmed A data-guided lexisearch algorithm for the quadratic assignment problem, Indian Journal of Science and Technology 7(4) (2014) pp. 480–490.

About the authors:

Zakir Hussain Ahmed is a Full Professor in the Department of Mathematics and Statistics at Imam Mohammad Ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia. From 2004 to 2019, he was in the Department of Computer Science as Professor at the same University. He obtained MSc in Mathematics (Gold Medalist), Diploma in Computer Application, MTech in Information Technology and PhD in Mathematical Sciences (Artificial Intelligence/Combinatorial Optimization) from Tezpur University (Central), Assam, India. Before joining the current institution, he served in Tezpur University, Sikkim Manipal Institute of Technology, Asansol Engineering College, and Jaypee Institute of Engineering and Technology, India. His research interests include artificial intelligence, combinatorial optimization, digital image processing and pattern recognition. He has several publications in the fields of artificial intelligence, combinatorial optimization and image processing.

Ibrahim Al-Dayel is the Head and Assistant Professor in the Department of Mathematics and Statistics at Imam Mohammad Ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia. He obtained PhD in Mathematics from King Saud University, Riyadh, KSA. His research interests include manifold, submanifold, Killing and conformal vector field, Jacobi field and geometry of hypersurfaces of spheres. He has several publications in these fields.