CONSTRAINED ASYMMETRIC THREE DIMENSIONAL GENERALIZED TRAVELLING SALESMAN PROBLEM::PATTERN RECOGNITION LEXI SEARCH APPROACH

[1]N Vasantha Kumar [2]Dr.U.BALAKRISHNA [3]Dr.E.Keshava Reddy

[1]Research Scholar, JNTUA College of Engineering, Annathapuramu, Andhra Pradesh
[2]Professor, Sreenivasa Institute of Technology & Management studies, Chittor
[3]Professor, Department of Mathematics, JNTUA College of Engineering, Annathapuramu, A.P

Abstract: This paper deals with the constrained asymmetric three dimensional generalized travelling salesman problem (CTGTSP) that concerns the generalization of classical travelling salesman problem. The GTSP is known to be an NP-hard problem and has many interesting applications. In CTGTSP, the cities can be categorized into different grade cities. The objective of this problem is to design optimal route that starts and ends at the same city and visits a subset of cities at a minimum cost such that the salesman has to visit from any grade city to another grade city with at least one city of each grade city and also consecutive visiting times(facilities) should be different. To find optimal solution, an exact pattern recognition technique based Lexi Search algorithm(LSA) is developed.A numerical example is demonstrated in order to understand the search mechanism of the Lexi Search Algorithm.
Further, to test the performance of Lexi Search Algorithm, computational experiments have been carried out on some benchmark as well as randomly generated test instances for CTGTSP, and results are reported. The overall computational results demonstrate that the proposed Lexi Search Algorithm is efficient in providing optimal within the considerable CPU times.

Keywords: Constraint Asymmetric three dimensional Generalized Travelling Salesman Problem (CTGTSP), Lexi Search Algorithm(LSA), Pattern Recognition Technique.

I. INTRODUCTION

The classical traveling salesman problem can be generalized in a natural way by considering a related problem relative to a given partition of the nodes of the graph into node sets (clusters), while the feasibility constraints are expressed in terms of the clusters, i.e. each of the clusters has to be visited once and only once (i.e. exactly once). In the literature, there are considered two versions of the problem: one in which we are interested in finding the shortest closed tour visiting exactly one node from each cluster. This problem is called the generalized traveling salesman problem (GTSP) and has been introduced independently by Henry-Labordere [8], Srivastava et al.[25] and Saskena[22]; the second one is the problem of finding the shortest closed tour including at least one vertex from each cluster. This version of the problem was introduced by Laporte and Nobert[10] and by Noon and Bean [14].
In the present paper we confine ourselves to the problem of choosing atleast one node from each of the clusters. The TSP is a special case of the GTSP where each cluster consists of exactly one node. The GTSP is NP-hard, as it reduces when each cluster consists of exactly one node to the traveling salesman problem which is known to be an NP-hard problem. The GTSP has several applications to location problems, planning, postal routing, logistics, manufacture of microchips, telecommunication problems, railway optimization, etc. More information on the problem and its applications can be found in Fischetti, Salazar and Toth [5,6], Laporte, Asef-Vaziri and Sriskandarajah [10], etc. A lot of attention was payed by the researchers for solving the GTSP: there have been proposed several transformations of the GTSP into TSP, an efficient exact algorithm has been proposed by Fischetti, Salazar and Toth [6] based on a branch-and-bound algorithm that solved the problem to optimality for instances with up to 89 clusters and 442 nodes, etc. The difficulty of obtaining optimal solutions for the GTSP has led to the development of several heuristic and metaheuristic algorithms: an efficient composite heuristic [21], reinforcing ant colony system [16], a random key genetic algorithm [24], variable neighborhood search [9], memetic algorithms [4,7], ant colony algorithms [27], lexi search algorithm[2,3,15,26]etc. Lexi Search algorithm exploit the good properties of different methods by applying them to optimization problems they can efficiently solve. For example, searching the solution space of an optimization problem is efficient when the problem has many solutions. In the case of combinatorial optimization problems the Lexi Search algorithms are designed to achieve a vast exploration of the search space, by escaping from local optima and intensifying at promising solutions regions. In order to solve this CTGTSP optimally, an exact algorithm namely, the pattern recognition technique based Lexi Search Algorithm (LSA) is developed. The problem CTGTSP has several real time applications in transportation and distribution system.

The paper is arranged as follows: The subsequent section 2 define the proposed problem and a zero-one integer programming mathematical formulation. Section 3 describes the preliminaries connected to the solution procedure. The proposed Lexi Search Algorithm (LSA) is presented in Section 4, whereas Section 5 provides a numerical illustration for CTGTSP. Computational details are reported in Section 6. Finally, concluding remarks are summarized in Section 7.

II. Problem description and Mathematical formulation

The CTGTSP can be defined as follows:

Let $G = (N,E)$ be a directed connected graph, where $N = \{1,2,...,n\}$ be the given set of n cities/nodes and E be an edge/arc set. A non-negative asymmetric cost C(i,j,k) [i, j=1,2,3,…..,n; k=1,2,3,…..,m] is associated with each edge $(i, j) \in E$ *at time (facility)k* and indicates the travel cost from ith city to jth city at time(facility) k. Consider the cities as different grades and atleast one city to be visited in each grade. For each edge $(i, j) \in E$, X(i,j,k) = 1, if the salesman visits city j from city i at time k and X(i,j,k) = 0 otherwise .The problem CTGTSP is to find a feasible tour for m(<n) cities for the salesman with minimum total cost by visiting any grade city to another grade city with atleast one city from each grade and consecutive visiting time(facility) should be different(i.e., he visits city j from city i at time k and visits city t from city j at time p(p≠k))

The following assumptions are used to formulate the model CTGTSP.

- The Total number of cities visiting is predefined
- The cities are divided into known number of different grade cities
- The cities in each grade is predefined
- Each city is to be visited exactly once
- The entries in the cost matrix assume arbitrary units.

Under the above assumptions, the model CTGTSP is formulated as a zero-one integer programming problem as follows:

Let N ={1,2,3,..,n} and let $N=N_1 \cup N_2 \cup N_3 \cup ... \cup N_m$, $N_i \cap N_j=\Phi$, $i \neq j$, and $i,j \in \{1, ..., m\}$.

$$\text{Min} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} C(i,j,k)X(i,j,k) \qquad \text{--------(1)}$$

Subjects to the constraints

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} X(i,j,k) = m \qquad \text{--------(2)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X(i,j,k)=1 \; for\, k=1,2,3,........, m \qquad \text{--------(3)}$$

If X(i,j,k) =1, then, ($i \varepsilon N_i$, $j \varepsilon N_j$) or ($i \varepsilon N_j$, $j \varepsilon N_i$) --------(4)
$|N_p| \geq 1$ for all p={1,2,3,…,m} --------(5)
If X(i,j,k)=1 and X(j,l,t)=1 then $k \neq t$ --------(6)
X(i,j,k) = {0 ,1}; $i,j \in N$ --------(7)

In the above model, (1) represents the objective function that minimizes the overall cost.
The constraint (2) ensures from the fact that any feasible solution consists of m cities. The constraint (3) represents that a salesman enters into each city exactly once. The constraint (4) represents a city is in N_i grade city and the other is in N_j grade city . The constraint (5) represents atleast one city to be visit in each grade. The constraint (6) represents, if the salesman visits city j from city i and then city l from city j then times(facilities) should be different. Finally, the constraint (7) represents the X(i,j,k) =1, the salesman visits city j from city i at time k and otherwise X(i, j, k) = 0.

III. PRELIMINARIES OF LSA

The components associated to the Lexi Search Algorithm are described as follows:

3.1 Feasible solution

A solution to the CTGTSP is said to be a feasible, if it satisfies all the constraints (2) to (7) in the problem.

3.2 Pattern

An indicator three-dimensional array which is associated with a tour is called a 'pattern'. A pattern is said to be feasible if X is a feasible solution. The value of the pattern *X* is determined using (8), provides the overall travel cost and this is equal to the value

of the objective function $V(X) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} X(i,j,k)\, C(i,j,k)$ --------(8)

3.3 Alphabet – Table

An alphabet table is the arrangement of the elements of the cost matrix C(i, j, k) in non decreasing order and indexed from 1 to $n\times n\times m$. Let SN = {1,2,..., $n\times n\times m$} be the set of $n\times n\times m$ ordered indices, arrays D and DC represent the costs and cumulative sums of the elements in D, respectively. Let the arrays R, C, T be the array of row, column and time(facility) indicies of the ordered triples represented by SN . The table comprises the set of ordered indices such as SN, D, DC, R ,C and T is referred as alphabet table. Let L_s= {s_1,s_2,s_3,…..,s_r} be an ordered string of r indices from the set SN , where s_i is a member of SN . The pattern L_s indicated by an ordered indices and these indices are independent of the order s_i in the sequence. For uniqueness, the indices from SN are organized in non-decreasing order such that $s_i < s_{i+1,\ i=1,2,3,\ldots,r-1}$

3.4 Word and partial word

An ordered sequence L_s= {s_1,s_2,s_3,…..,s_r}is represented as a word of length *r* . A feasible word L_s is said to be a partial feasible word if $r < m$ and if r = m, then it represents the full length feasible word or simply a word. Any one of the indices from *SN* can take up the prime position in the partial word L_s . A partial word L_s defines a block of words with L_s as a leader. If the block of word characterized by it has at least one feasible word then the leader is said to be feasible, otherwise infeasible.

3.5 Value of a word

The value of the word L_s denoted by $V(L_s)$ is determined by using $V(L_s)=V(L_{s-1})+D(s_r)$ with $V(L_0)=0$ and obviously $D(s_r)$ be the distance array which is organized in such a way that $D(s_r) \le D(s_{r-1})$ for r = 1,2,3,..., $n\times n\times m$. $V(L_s)$ is similar to the value of V(X) (Sundara Murthy[26]).

3.6 Computation of bounds

The effective setting of lower and upper bounds are more challenging to the class of NP-hard problems to control the search space. Initially, the upper bound of L_s is assumed to be a high value (*UB* = *VT*= 9999) (for minimization objective functions) as a trial solution. The lower bound A lower bound LB (L_s) for the values of the block of words represented by L_s can be defined as follows: $LB(L_s) = V(L_s)+\sum_{j=1}^{m-k} D(s_k + j) = V(L_s)+DC(s_k + m-k) - DC(s_k)$

Lexi Searh Method:

Optimal solutions obtained by exact search methods have grown into more attractive in the context of solving combinatorial optimization problems in order to make effective decisions. The exact approaches can be observed as exhaustive and implicit search methods. One of the prominent implicit search technique is Branch and Bound method (Little et al., 1963[27]). Lexi search algorithm is one such implicit enumeration procedure, due to effective bound settings, only a fractional part of a solution space is investigated and converges to optimal solution systematically (Pandit, 1962[25]), which was developed to tackle the loading problem. Infact, Branch & Bound can be seen as a special case of Lexi search algorithm. The Lexi search algorithm takes care of all the components of Branch & Bound such as the development of feasible solutions, feasibility checking and determining the bounds for the partial feasible solution. The entire search process is done in a precise manner and resembles to the search for an essence of a word in a dictionary, thus, the name is given as "Lexi-search". Moreover, this systematic search defends stack overflow and search time. The main difficulty of any problem utilizing implicit enumeration methods is (i) checking the feasibility (ii) setting effective bounds. There is a difficulty in testing the feasibility for few problems. To overcome this, a pattern recognition technique based Lexi-search approach (Murthy, 1976[26]) 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 is 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".

IV. *Proposed* Lexi-*search Algorithm*

The procedure of Lexi-search algorithm is described as follows:

Step 1: ***Initialization***
Initialize the cost matrix C=[C(i,j,k)],the required parameters *m*,*n* and *UB* = *VT* = 9999 (large value) and go to Step 2.
Step 2: Construct an alphabet table using the given cost matrix C as discussed in the Section 3.3 and move to Step 3.
Step 3: Bound Settings
The algorithm starts with a partial word $L_s=(s_s)=1$, $s_s \in SN$, where the length of the partial word is unity, i.e. *s=1*.
Determine the lower bound of a partial word $LB(L_s)$.If $LB(L_s)<VT$, then go to Step 5, else go to Step 4.
Step 4: If $LB(L_s) \ge VT$, then drop the partial word *Ls* and dismiss the block of words with *Ls* as leader. Since it does not yield an optimal solution and thus, reject all the partial words of the order s that succeeds L_s and go to Step 7.

Step 5: ***Feasibility Checking***

If the partial word L_s satisfies the constraints then it is said to be feasible, otherwise, it is infeasible. If L_s is feasible, then accept it and continue for next partial word of order s+1 and go to Step 6, else proceed with the next partial word of order s by considering another letter that succeeds S_s in its s^{th} position and go to Step 3.

Step 6: ***Concatenation***

If L_s is a full length feasible word of length s (i.e. s=m) , then replace *VT* by the value of $LB(L_s)$ and then go to Step 8. If L_s is a partial word, then it can be concatenated by using $L_{s+1} = L_s *(S_{s+1})$,where * indicates the concatenation operation and go to step 3.

Step 7: If all the words of order s are exhausted and length of the word L_s is 1, then the Search mechanism is terminated and go to Step 9, else move to Step 8.

Step 8: ***Backtracking***

Backtracking is adopted to explore the search space; the current *VT* is assumed as an upper bound and continues the search with next letter of the partial word of order s -1, go to Step 3. Repeat the Steps 3 to 8 until *VT* has no further improvement and ignore the feasible/infeasible solutions which are not constitute in the optimal solution. Go to Step 9.

Step 9: Record the latest *VT* and the corresponding word L_s . Go to Step 10.

Step 10: *Stop*

Finally, at the end of the search, *VT* provides the optimal solution and the word L_s give the position of the letters and one can find the optimal schedule for connectivity of given cities with the help of L_s.

V. NUMERICAL ILLUSTRATION

A numerical example with 6 cities is considered to explain the concepts and the Lexi Search Algorithm for CTGTSP, for which N=[1,2,3,4,5,6], m=4, the cities 1&2 as first grade; cities 3&4 as second grade and the remaining cities 5&6 as third grade i.e., N_1=1,2; N_2=3,4 and N_3=5,6 and atleast one city to be visited in each group.

The cost between each pair of cities assumes a non-negative quantity, can be asymmetric, represented as a cost matrix C and is given in Table 1, where '–'indicates the disconnectivity or self-loop between the pair of cities. The problem is to find the best route plan for the salesman to cover 4 cities with atleast one city from each grade such that the overall cost is minimum. The asymmetric cost matrix C assumes the non-negative values (arbitrary units) and is given in Table 1.

Table –1

$C(i,j,1)=\begin{bmatrix} - & 14 & 27 & 2 & 10 & 26 \\ 17 & - & 15 & 22 & 4 & 8 \\ 22 & 7 & - & 16 & 71 & 54 \\ 1 & 7 & 17 & - & 5 & 29 \\ 51 & 31 & 41 & 5 & - & 21 \\ 61 & 71 & 14 & 1 & 7 & - \end{bmatrix}$	$C(i,j,2)=\begin{bmatrix} - & 7 & 61 & 1 & 17 & 21 \\ 64 & - & 4 & 5 & 17 & 29 \\ 9 & 21 & - & 31 & 67 & 15 \\ 31 & 2 & 0 & - & 16 & 49 \\ 1 & 74 & 14 & 91 & - & 29 \\ 29 & 6 & 67 & 7 & 1 & - \end{bmatrix}$	$C(i,j,3)=\begin{bmatrix} - & 41 & 7 & 41 & 1 & 9 \\ 71 & - & 29 & 75 & 41 & 7 \\ 4 & 17 & - & 7 & 3 & 20 \\ 71 & 6 & 15 & - & 41 & 71 \\ 1 & 29 & 21 & 21 & - & 49 \\ 5 & 29 & 2 & 49 & 7 & - \end{bmatrix}$

5.1 *Alphabet table*

Table 2 concerns the construction of alphabet table as discussed in Section 3.3 for the cost matrix C. The first three columns report that the serial number (*SN*) , cost (D) and cumulative cost (DC) , respectively. The subsequent three columns provide the details about row (*R*) , column (*C*) and time(T)indices, respectively. For convenience, a partial alphabet table is considered and given in Table 2.

Table-2(Alphabet - Table)

SN	D	DC	R	C	T		SN	D	DC	R	C	T		SN	D	DC	R	C	T
1	0	0	4	3	2		31	8	131	2	6	1		61	29	701	6	1	2
2	1	1	4	1	1		32	9	140	3	1	2		62	29	730	2	3	3
3	1	2	6	4	1		33	9	149	1	6	3		63	29	759	5	2	3
4	1	3	1	4	2		34	10	159	1	5	1		64	29	788	6	2	3
5	1	4	5	1	2		35	14	173	1	2	1		65	31	819	5	2	1
6	1	5	6	5	2		36	14	187	6	3	1		66	31	850	3	4	2
7	1	6	1	5	3		37	14	201	5	3	2		67	31	881	4	1	2
8	1	7	5	1	3		38	15	216	2	3	1		68	41	922	5	3	1
9	2	9	1	4	1		39	15	231	3	6	2		69	41	963	1	2	3
10	2	11	4	2	2		40	15	246	4	3	3		70	41	1004	1	4	3
11	2	13	6	3	3		41	16	262	3	4	1		71	41	1045	2	5	3
12	3	16	3	5	3		42	16	278	4	5	2		72	41	1086	4	5	3
13	4	20	2	5	1		43	17	295	2	1	1		73	49	1135	4	6	2
14	4	24	2	3	2		44	17	312	4	3	1		74	49	1184	5	6	3
15	4	28	3	1	3		45	17	329	1	5	2		75	49	1233	6	4	3

16	5	33	4	5	1		46	17	346	2	5	2		76	51	1284	5	1	1
17	5	38	5	4	1		47	17	363	3	2	3		77	54	1338	3	6	1
18	5	43	2	4	2		48	20	383	3	6	3		78	61	1399	6	1	1
19	5	48	6	1	3		49	21	404	5	6	1		79	61	1460	1	3	2
20	6	54	6	2	2		50	21	425	1	6	2		80	64	1524	2	1	2
21	6	60	4	2	3		51	21	446	3	2	2		81	67	1591	3	5	2
22	7	67	3	2	1		52	21	467	5	3	3		82	67	1658	6	3	2
23	7	74	4	2	1		53	21	488	5	4	3		83	71	1729	3	5	1
24	7	81	6	5	1		54	22	510	2	4	1		84	71	1800	6	2	1
25	7	88	1	2	2		55	22	532	3	1	1		85	71	1871	2	1	3
26	7	95	6	4	2		56	26	558	1	6	1		86	71	1942	4	1	3
27	7	102	1	3	3		57	27	585	1	3	1		87	71	2013	4	6	3
28	7	109	2	6	3		58	29	614	4	6	1		88	74	2087	5	2	2
29	7	116	3	4	3		59	29	643	2	6	2		89	75	2162	2	4	3
30	7	123	6	5	3		60	29	672	5	6	2		90	91	2253	5	4	2

5.2 *Search table*

The logical flow of the developed Lexi search algorithm(LSA) (presented in Section 4) is given through a numerical example in Table 3.

Table-3(Search Table)

SN	1	2	3	4	V (I)	LB (I)	R	C	T	REM		SN	1	2	3	4	V (I)	LB (I)	R	C	T	REM
1	1				0	1	4	3	2	A		66			8		3	5	5	1	3	R
2		2			1	2	4	1	1	R		67			9		4	6	1	4	1	R=VT
3		3			1	2	6	4	1	A		68		8			2	4	5	1	3	R
4			4		2	3	1	4	2	R		69		9			3	5	1	4	1	R
5			5		2	3	5	1	2	R		70		10			3	6	4	2	2	R=VT
6			6		2	3	6	5	2	R		71	3				1	2	6	4	1	A
7			7		2	3	1	5	3	R		72		4			2	3	1	4	2	R
8			8		2	4	5	1	3	R		73		5			2	3	5	1	2	A
9			9		3	5	1	4	1	R		74			6		3	4	6	5	2	R
10			10		3	5	4	2	2	R		75			7		3	4	1	5	3	R
11			11		3	6	6	3	3	R		76			8		3	5	5	1	3	R
12			12		4	8	3	5	3	R		77			9		4	6	1	4	1	R=VT
13			13		5	9	2	5	1	R		78		6			2	3	6	5	2	R
14			14		5	9	2	3	2	R		79		7			2	4	1	5	3	A
15			15		5	10	3	1	3	R		80			8		3	5	5	1	3	R
16			16		6	11	4	5	1	R		81			9		4	6	1	4	1	R=VT
17			17		6	11	5	4	1	R		82		8			2	4	5	1	3	A
18			18		6	11	2	4	2	R		83			9		4	6	1	4	1	R=VT
19			19		6	12	6	1	3	R		84		9			3	5	1	4	1	R
20			20		7	13	6	2	2	R		85		10			3	6	4	2	2	R=VT
21			21		7	14	3	2	1	A		86	4				1	2	1	4	2	A
22				22	14	14	4	2	1	R		87		5			2	3	5	1	2	R
23				23	14	14	6	5	1	R		88		6			2	3	6	5	2	A
24				24	14	14	1	2	2	R		89			7		3	4	1	5	3	R
25				25	14	14	6	4	2	R		90			8		3	5	5	1	3	A
26				26	14	14	1	3	3	R		91				9	5	5	1	4	1	R
27				27	14	14	2	6	3	A=VT =14		92				10	5	5	4	2	2	R
28			22		8	15	3	2	1	R>VT		93				11	5	5	6	3	3	R
29		4			1	2	1	4	2	R		94				12	6	6	3	5	3	R=VT
30		5			1	2	5	1	2	A		95			9		4	6	1	4	1	R=VT
31			6		2	3	6	5	2	R		96		7			2	4	1	5	3	R
32			7		2	3	1	5	3	R		97		8			2	4	5	1	3	A
33			8		2	4	5	1	3	R		98			9		4	6	1	4	1	R=VT
34			9		3	5	1	4	1	A		99		9			3	5	1	4	1	A
35				10	5	5	4	2	2	R		100		10			3	6	4	2	2	R=VT
36				11	5	5	6	3	3	R		101	5				1	2	5	1	2	A
37				12	6	6	3	5	3	A=VT =6		102		6			2	3	6	5	2	A
38			10		3	5	4	2	2	R		103			7		3	4	1	5	3	R
39			11		3	6	6	3	3	R=VT		104			8		3	5	5	1	3	R
40		6			1	2	6	5	2	R		105			9		4	6	1	4	1	R=VT

41		7			1	3	1	5	3	A		106		7			2	4	1	5	3	R
42			8		2	4	5	1	3	R		107		8			2	4	5	1	3	A
43			9		3	5	1	4	1	R		108			9		4	6	1	4	1	R=VT
44			10		3	5	4	2	2	R		109		9			3	5	1	4	1	A
45			11		3	6	6	3	3	R=VT		110			10		5	7	4	2	2	R>VT
46		8			1	3	5	1	3	A		111		10			3	6	4	2	2	R=VT
47			9		3	5	1	4	1	A		112	6				1	3	6	5	2	A
48				10	5	5	4	2	2	R		113		7			2	4	1	5	3	R
49				11	5	5	6	3	3	R		114		8			2	4	5	1	3	A
50				12	6	6	3	5	3	R=VT		115			9		4	6	1	4	1	R=VT
51			10		3	5	4	2	2	R		116		9			3	5	1	4	1	A
52			11		3	6	6	3	3	R=VT		117			10		5	7	4	2	2	R>VT
53		9			2	4	1	4	1	A		118		10			3	6	4	2	2	R=VT
54			10		4	6	4	2	2	R=VT		119	7				1	3	1	5	3	A
55		10			2	5	4	2	2	R		120		8			2	4	5	1	3	R
56		11			2	6	6	3	3	R=VT		121		9			3	5	1	4	1	R
57	2				1	2	4	1	1	A		122		10			3	6	4	2	2	R=VT
58		3			2	3	6	4	1	R		123	8				1	3	5	1	3	A
59		4			2	3	1	4	2	R		124		9			3	5	1	4	1	A
60		5			2	3	5	1	2	R		125			10		5	7	4	2	2	R>VT
61		6			2	3	6	5	2	A		126		10			3	6	4	2	2	R=VT
62			7		3	4	1	5	3	R		127	9				2	5	1	4	1	A
63			8		3	5	5	1	3	R		128		10			4	7	4	2	2	R>VT
64			9		4	6	1	4	1	R=VT		129	10				2	6	4	2	2	R=VT
65		7			2	4	1	5	3	A												

Table 3 explains the details that how the algorithm enumerates the solutions as well as converges to the optimal solution. The column indexed by *SN* represents the serial number. Since *n =6,m=4*, therefore the total number of arcs required for the optimal schedule of CTGTSP is 4 . Thus, the length of optimal feasible word becomes 4 .The columns 1, 2, 3, 4 of Table 3 represents the respective positions of the letters of a word L_s . The subsequent columns labelled as *V*, *LB*,*R*, *C and T* respectively represent the value, lower bound, row ,column and time indices of the partial word. Finally, the column indexed by *REM* represents the remarks of a partial word i.e. if a partial word is feasible then it is accepted and denoted by 'A', otherwise rejected and indicated by 'R'. Here, serial number *SN* indicates the iteration count.

Optimal and sub-optimal solutions

The set of solutions, which are observed from the search table are given in Table 4. Table 4 reports the details of feasible patterns, feasible (sub-optimal) and optimal solutions. The initial found pattern L_4 = {1,3,21,27} gives the objective function value *VT* =14 units that is noticed at 27th row of the Table 3. In order to improve this solution backtracking is performed. After performing the backtracking by considering the initial found solution (i.e. 14 units) as current upper bound, the best objective function value as *VT=6* units and whose feasible pattern L_4 = {1,5,9,12} is found at 37th row of the Table 3. Table 3 clearly shows that the objective function value *VT=6* units dominates all the other solutions, and hence the current solution (i.e.*VT =6* units) become the optimal solution. This clearly shows the developed Lexi search is capable to enumerate the possible solutions that assist the decision maker to construct viable decisions with preferred solutions also. The graphical representation of respective feasible and optimal solutions is given in Fig 2 and Fig 3.

Table-4(Optimal and Sub-optimal Solutions)

SN	**Feasible Pattern**	**Corresponding tour**	**Solution**
1.	L_4 = {1,3,21,27}	**(4,3,2),(6,4,1),(3,2,1),(2,6,3)**	**14(Sub-optimal)**
2.	L_4 = {1,5,9,12}	**(4,3,2),(5,1,2),(1,4,1),(3,5,3)**	**6(Optimal)**

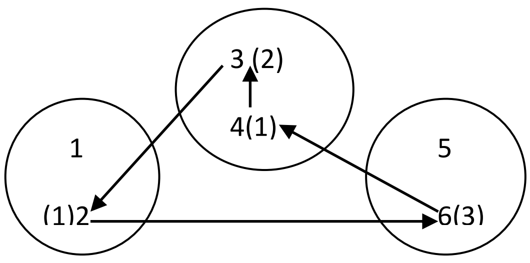

Fig.2. SUB OPTIMAL SOLUTION

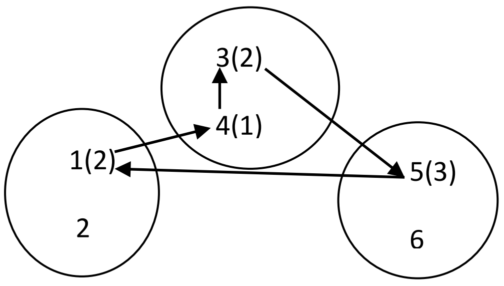

Fig.3. OPTIMAL SOLUTION

VI. COMPUTATIONAL ANALYSIS

The performance of the proposed Lexi Search Algorithm for solving the CTGTSP was tested on some benchmark problems. All the experiments were conducted by implementing the LSA in Matlab 2017a and then running on PC with 2.0 GHz, Intel(R) core i3 processor, 4 GB of RAM running Microsoft Windows 10 Operating System. Analyzing the computational results, it results that overall the proposed Lexi Search Algorithm performs well in terms of solution quality. The results obtained using our Lexi Search Algorithm are facilitated by reducing considerably the solutions space and a better exploration of the search space.Table-5 gives the list of the problems tried along with the average CPU time in seconds required for solving them.

Table 5
Descriptive statistics of CPU runtime of LSA on random instances

SN	|N|	|G|	|S|	M	NPT	CPU runtime (In Seconds)		
						Min	Max	Avg
1.	15	3	5	9	5	0.0534	0.0743	0.0638
2.	40	5	8	20	5	2.2346	3.5473	2.8909
3.	60	10	6	32	5	5.3245	7.2314	6.2779
4.	84	12	7	44	5	13.2317	16.3243	14.778
5.	98	14	7	68	5	17.3246	19.4374	18.381

SN–Serial Number; |*N*| – Number of cities; |G| - Number of grade cities; |S| - Number of cities in each grade cities; *m* – Number of cities to be visited; NPT–Number of problems tried; Min.–Minimum CPU runtime required for finding best solution; Max.– Maximum CPU runtime required for finding best solution; Avg.– Average CPU runtime required for finding best solution;

VII. CONCLUSION

In this paper, we considered an exceptional combinatorial optimization problem called Constrained Asymmetric three dimensional generalized travelling salesman problem (CTGTSP), motivated by the real world outsourcing scenarios in human resource allocation and routing problems. The model CTGTSP has been presented as a zero-one integer programming. An efficient exact algorithm, the pattern recognition technique based Lexi Search Algorithm (LSA) is developed for CTGTSP. The Lexi Search Algorithm performance of CTGTSP is tested over some benchmark as well as randomly generated test instances and the results are reported. The extensive computational results showed that the LSA performs well in yielding exact solutions within practically considerable CPU runtimes. Furthermore, an interesting observation is that the key parameters $|N|,|G|,|S|$ *and* m judge the performance of the LSA for solving CTGTSP. The model CTGTSP finds good number of applications in transportation, vehicle routing and logistics distributions etc. For the future consideration, one can extend the model CTGTSP with multiple salesmans and other practical variants etc. However, developing an efficient exact algorithm for such variants is still a challenging problem.

VIII. ACKNOWLEDGEMENT

The authors would like to thank the anonymous referees for constructive comments on earlier version of this paper.

REFERENCES

[1]Bäck, T., Hoffmeister, F., Schwefel, H.: A survey of evolution strategies. In: Proc. of the 4th International Conference on Genetic Algorithms, San Diego, CA (July 1991)
[2]Balakrishna, U and Sundara Murthy, M. (2009). Three dimensional TSP and ASP models,PhD Thesis,SV University,Tirupati.
[3]Bhavani, V. and Sundara Murthy, M. (2005). Time-Dependent Travelling Salesman Problem – OPSEARCH 42, PP. 199-227
[4]Bontoux, B., Artigues, C., Feillet, D.: A Memetic Algorithm with a Large Neighborhood Crossover Operator for the Generalized Traveling Salesman Problem. Computers & Operations Research (2009)
[5]Fischetti, M., Salazar, J.J., Toth, P.: The symmetric generalized traveling salesman polytope. Networks 26, 113–123 (1995)
[6]Fischetti, M., Salazar, J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research 45, 378–394 (1997)
[7]Gutin, G., Karapetyan, D.: A memetic algorithm for the generalized traveling salesman problem. Natural Computing 9, 47–60 (2009)
[8]Henry-Labordere: The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. RAIRO Operations Research B2, 43-49 (1969)
[9]Hu, B., Raidl, G.: Effective neighborhood structures for the generalized traveling salesman problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 36–47. Springer, Heidelberg (2008)
[10]Laporte, G., Nobert, Y.: Generalized Traveling Salesman through n sets of nodes: an integer programming approach. INFOR 21, 61–75 (1983)
[11]Laporte, G., Asef-Vaziri, A., Sriskandarajah, C.: Some applications of the generalized traveling salesman problem. Journal of Operational Research Society 47, 1461–1467 (1996)
[12]Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, *11*(6), 972-989
[13]Matei, O.: Evolutionary Computation: Principles and Practices, Risoprint (2008)
[14]Noon, C.E., Bean, J.C.: A Lagrangian based approach for the asymmetric generalized traveling salesman problem. Operations Research 39, 623–632 (1991)
[15]Pandit, S. N. N and Rajbhugshi (1976). Restricted Travelling Salesman Problem through n sets of nodes. Papers presented at the 9th annual convention of ORSI, Calcutta.

[16]Pintea, C., Pop, P.C., Chira, C.: Reinforcing Ant Colony System for the Generalized Traveling Salesman Problem. In: Proc. of International Conference Bio-Inspired Computing-Theory and Applications (BIC-TA), Wuhan, China. Evolutionary Computing Section, pp. 245–252 (2006)
[17]Pop, P.C.: The generalized minimum spanning tree problem. Twente University Press, The Netherlands (2002)
[18]Pop, P.C., Kern, W., Still, G.J.: A new relaxation method for the generalized minimum spanning tree problem. European Journal of Operational Research 170, 900–908 (2006)
[19]Pop, P.C.: A survey of different integer programming formulations of the generalized minimum spanning tree problem. Carpathian J. of Math. 25(1), 104–118 (2009)
[20]Pop, P.C., Matei, O., Pop Sitar, C., Chira, C.: A Genetic Algorithm for Solving the Generalized Vehicle Routing Problem. In: Corchado, E., Graña Romay, M., Manhaes Savio, A. (eds.) HAIS 2010, Part II. LNCS, vol. 6077, pp. 119–126. Springer, Heidelberg (2010)
[21]Renaud, J., Boctor, F.F.: An efficient composite heuristic for the Symmetric Generalized Traveling Salesman Problem. European Journal of Operational Research 108(3), 571–584 (1998)
[22]Saskena, J.P.: Mathematical model of scheduling clients through welfare agencies. Journal of the Canadian Operational Research Society 8, 185–200 (1970)
[23]Schwefel, H.P.: Collective phenomena in evolutionary systems. In: Proc. of 31st Annual Meetting of the International Society for General System Research, pp. 1025–1033 (1987)
[24]Snyder,L.V.,Daskin,M.S.:A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operations Research 174, 38–53 (2006)
[25]Srivastava, S.S., Kumar, S., Garg, R.C., Sen, P.: Generalized traveling salesman problem through n sets of nodes. CORS Journal 7, 97–101 (1969)
[26]Sundara Murthy, M. (1979). Combinational Programming. A Pattern Recognition Approach – Ph.D. Thesis, REC, Warangal.
[27]Yanga, J., Shi, X., Marchese, M., Liang, Y.: An ant colony optimization method for generalized TSP problem. Progress in Natural Science 18(11), 1417–1422 (2008)