Lexicographic optimization of Three Dimensional Travelling Salesman Problem

Dr. Krishan Murari Agrawal
Associate Professor, Mathematics
Bipin Bihari (P.G) College, Jhansi (up)-India
kmagrawala@gmail.com

Abstract

In this paper Lexicographic Approach is used to solve a three Dimensional Travelling salesman problem. Three dimensional travelling salesman problem have different cost between two cities at different time. So a Salesman is to make a tour in such a way that it travels a station once and no station is left unvisited. The tour must be chosen in the manner that Cost must be minimum for the optimum tour, taking consideration of different time. The Travelling salesman problem is different as for n-city problem , n-zones of time are given and the cost between every two pair of cities is different for different time zones. And the tour is so chosen that at one time zone salesman visits only one pair of cities .The Lexicographic algorithm is developed in this paper to solve this type of problem.The Lexicographic algorithm is highly effective in respect of time consumption and calculating steps and size of complexity .The lexicographic algorithm presented in this paper is simpler than the lexicographic algorithm available in literature .A numerical example is also given after the algorithm.

Keywords :Lexicographic approach, Three dimensional Travelling salesman problem.

1.INTRODUCTION

Travelling salesman problem is a classical problem in combinatorial optimization. Since 1950's it has been studied intensively. As a result of it, a large number of techniques were developed to solve the problem. The objective of the problem is to find the shortest route of salesman starting from a given city, visiting all other cities only once and finally come to the same city where he started. There are different approaches for solving travelling salesman problems. Linear programming method, heuristic methods like cutting plan algorithms and branch and bound method, Markov chain, simulated annealing and tabu search methods. Few more algorithms like particle swarm optimization, neural networks, evolutionary computations, ant system, artificial bee colony, etc., are also there.

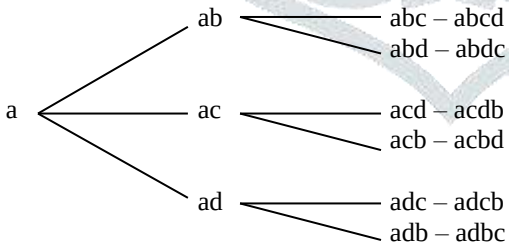
There are certain approaches for the exact solution of TSP and they are

1. Integer Programming.
2. Dynamic Programming
3. Branch and bound algorithm.
4. Lexicographic Search Method.

Vijyalaxmi G.[14] gave Lexicographic approach to travelling salesman problem, Sobhan Babu,K et al[11] gave an algorithm for travelling salesman problem, Venkatadri Naidu et al [13] gave optimization of service operations sequencing a four wheeler at service centre. Liu, G.et al[7] gave, a novel adaptive search strategy of intensification and diversification in tabu search .Our interest is in Lexicographic search. Section 2 of this paper contains definition of Lexicographic search. Section 3 has Mathematical formulation of the Travelling salesman problem ,Definitions of the terms used in the Algorithm developed in the paper and The Lexicographic Algorithm itself. Section 4 is Numerical Illustration and Section 5 is conclusion.

2. LEXICOGRAPHIC METHOD :

Let a,b,c,d are the name of the stations on which the salesman is to visit. Lexicographic method works as following.



It is just like Dictionary method. when a salesman starts from station 'a' it can go the stations 'b', 'c', 'd' so the route becomes 'ab', 'ac', 'ad'. Let 'ab' is the selected route then next route may be 'abc' or 'abd'. Now let 'abc' is the route then 'abcd' is the only way. Rest can be understood by the above figure.

3.1 MATHEMATICAL FORMULATION:

A n-city three dimensional travelling salesman problem can be mathematically stated as

Minimize Z $= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} C\,(i,j,k)X\,(i,j,k)$

Subject to

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

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

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

X (i, j, k) = 1 or 0, represents the salesmen's visit, when salesman visits city j from city i at time k or otherwise.

X will be a feasible solution if it gives a tour for the salesman who visits the n cities with the condition that at a point of time in his tour he will not visit more than one pair of cities.

3.2 Concepts and Definitions

3.2 (a) Definition of a Pattern

An indicator three-dimensionally 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 pattern represented in the table -4.2 is a feasible pattern.

It can be noticed that at a given time k the matrix X (i,j,k) (i,j, = 1,2,3,4) has for any one element the value 1 and for all the elements are zeros. This indicates that at time k if x (i,j,k)=1 then he visits city j from city i.

Now V (X) the value of the pattern x is defined as

$$V(X) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} C(i,j,k)\, X(i,j,k)$$

The value V (x) gives the total cost of the tour for the solution represented by X. Thus the value of the feasible pattern gives the total cost represented by it. In the algorithm, which is developed in the sequel, a search is made for a feasible pattern with the least value. Each pattern of the solution x is represents the set of ordered triples [(i,j,k)] for which x (i,j,k) = 1 with the understanding that the other x (i,j,k)'s are zeros. The ordered triple set [(2,4,1), (1,3,2), (4,1,3), (3,2,4)] represents the pattern given in the Table -4.2, which is a feasible solution.

There are n^3 ordered triple in the three dimension array x, for convenience these are arranged in ascending order of their corresponding costs and are indexed from 1 to n^3. Let SN = [1.2...n^3] be the set of n^3 indices. Let D be the corresponding array of distances. If $\alpha, \beta \in$ SN and $\alpha < \beta$ then D (α) $\leq$ D (β). Also let the arrays R, C and T be the array of row, column and time indices of the ordered triples represented by SN and DC be the array of cumulative sums of the elements of D.

The arrays SN, D, DC, R, C, T for the numerical example are given in the Table 4.3. If P ϵ SN then [R(P), C (P), T (P),] is the ordered triple and D (a) = [R(a),C(a),T(a)] is the value of the ordered triple and DC (a) = $\sum_{i=1}^{a} D(i)$

3.2 (b) Definition of an alphabet - Table and a word

Let L_k = {a_1, a_2, ... a_k, } aj ϵ SN be an ordered sequence of k indices from SN. The pattern represented by the ordered triples whose indices are given by L_k is independent of the order of a_i in the sequence. Hence for uniqueness the indices are arranged in the increasing order such that $a_j \leq a_{j+1}$, i = 1,2, ... n-1. The set SN is defined as the "Alphabet - Table" with alphabetic order as (1,2,3 ... n^3) and the ordered sequence L_k is defined as a "word" of length k. A word L_k is called a "sensible word", if $a_i < a_{j+1}$ for i = 1,2,... k-1 and if this condition is not met it is called a "insensible word".

A word L_k is said to be feasible if the corresponding pattern x is feasible and same is with the case of infeasible and partial feasible Therefore a partial feasible word is said to be feasible if k = n. A partial word L_k is said to be feasible if the block of words represented by L_k has at least one feasible word or equivalently the partial pattern represented by L_k should not have any inconsistency.

Any of the letters in SN can occupy the first place in the partial word L_k. Consider L_{k-1} = (a_1,a_2...a_{k-1}). The alphabet for the k^{th} position is SN a_{k-1} = (a_{k-1}+1, a_{k-1}+2,……. N^3)

SN_P is defined as SN_p = (p +1, p+2, …..n^3). Thus for example consider a word with two letters as (a_1, a_2) = (1,3). Then SNa_2 = SN_3 = (4, 5, 6 ...64) is the alphabet for the third position. We concentrate on the set of words of length almost n (for the numerical example it is 4). A leader L_k (k < n) is said to be feasible, if the block of words defined by it contains at least one feasible word or equivalently there should not be inconsistency in the partial pattern defined by the partial word. The value of the L_k, V(L_k) is defined recursively as

V (L_k) = V (L_{k-1}) + D (a_k) with V (L_o) = 0 obviously this V (L_k) and V (X) the values of the pattern X represented by L_k will be same. For example consider L_3 = (1,2,5). Then V (L_3) will be V (L_3) = 0+0+0= 0.

3.2 (c) Lower bound of a partial word LB (L_k)

A lower bound LB (L_k) for the values of the block of words represented by L_k can be defined as follows:

LB (L_k) = V (L_k) + $\sum_{j=1}^{n-k} D(a_k + j)$

= V (L_k) + DC (a_k + n-k)- DC (a_k)

Consider the partial word L_3 = (1,2,8)

V(L_3) = 0+0+0=0

LB (L_3) = V (L_3) + DC (a_3 + 4-3) - DC (a_3)

= 0 + DC (8+4-3) - DC (8)

= DC (9) - DC (8)

= 1 - 0 = 1

3.2 (d) Algorithm for - feasibility criterion of a partial word

A recursive algorithm is developed for checking the feasibility of a partial word L_{k+1} = (a_1, a_2, .. a_k, a_{k+1}) given that L_k is a feasible partial word. We will introduce some more notations which will be useful in the sequence

RI Be an array where RI (i) = 1, i ϵ N represents that the salesman is visiting some city from city i, otherwise zero. (It represents that salesman is visiting or not visiting from city i, for that RI (i)=1 or 0)

CI be an array where CI (i) = 1, i ϵ N represents that the salesman is coming to city i from some city; otherwise CI (i) =0.

TI be an array where TI (i) =1, i ϵ N represents that the salesman at time i travels one pair of cities.

SW be an array where SW (i) is the city that the salesman is visiting from city i and SW (i) = 0 indicates that the salesman is not visiting any city from city i.

L Be an array where L (i) is the letter in the i^{th} position of a word. Then for a given partial word L_k (a_1, a_2, .. a_k) the values of the arrays RI, CI, TI, SW, L are as follows. RI [R (a_i)] = 1, i = 1,2,...k and RI (j) = 0 for other elements of j. CI [C (a_i)] = 1, i = 1,2,...k and CI (j) = 0 for other elements of i. TI (T (a_i)] = 1, i = 1,2,....k and TI (j) = 0 for other elements of i. SW [R(a_i)] = C (a_i) i = 1,2, ….. k and SW (j)=0 for other

elements of j. $L(i) = a_i$, $i = 1,2, .. k$, $L(i) = 0$ $i > k$

For example consider a sensible partial word $L_3 = (1,6.8)$ which is feasible. The arrays RI, CI, TI, SW and L takes the values represented in the Table -4.4

The recursive algorithm for checking the feasibility of a partial word L_{P-1} is given as follows. In this algorithm L_p is a feasible word. We test the following tips for introducing a_{p+1} in the L_p

In the algorithm, first we equate XI = 0. At the end if XI = 1 then the partial word L_{p+1} is feasible. Otherwise it is infeasible.

THE ALGORITHM

STEP 1 XI = 0
GOTO 2

STEP 2 IS $[RI\{R(a_p + 1)\}] = 1$ IF YES GOTO 8
IF NO GOTO 3

STEP 3 IS $[CI\{C(a_p + 1)\}] = 1$ IF YES GOTO 8
IF NO GOTO 4

STEP 4 IS $[TI\{T(a_p + 1)\}] = 1$ IF YES GOTO 8
IF NO GOT0 5

STEP 5 IS $[SW\{C(a_p + 1)\}] = 0$ IF YES XI =1 GOTO 8
IF NO
$K = SW\{C(a_{P+1})\}$ GOTO 6

STEP 6 IS $\{K = R(a_p + 1)\}$ IF YES GOTO 7
IF NO, Assume $C(a_{p+1}) = K$
GOTO 5

STEP 7 IS (I = N) IF YES XI =1 GOTO 8
IF NO GOTO 8

STEP 8 STOP

This recursive algorithm can also be used as a subroutine in the Lexi - search algorithm. We start the algorithm with a very large value say 999 as a trial value of VT. If the value of a feasible word is known we can as well start with that value as VT. During the search the value of VT is improved. At the end of the search the currént value of VT gives the optimal feasible word. We start with the partial world $L_1(a_1) = (1)$. A partial word L_p is constructed as $L_p = L_{p-1} * (a_p)$, where * indicates chain form or concatenation. We will calculate the values of V (L_p) and LB (L_p) simultaneously. Then two cases arises one for branching and other for continuing the search.

1- LB (L_p) < VT. Then we check whether L_p is feasible or not. If it is feasible we proceed to consider a partial word of order (P+1), which represents a sub block of the block of words represented by L_p. If L_p is not feasible then consider the next partial word of order p by taking another letter which succeeds a_p in the P^{th} position. If all the words of order P are exhausted then we consider the next partial word of order (P-1).
2- LB (L_p) ≥ VT. In this case we reject the partial word, meaning that the block of words with L_p as leader is rejected for not having an optimal word and we also reject all partial words of order P that succeeds L_p.

Now we are in a position to develop a lexi-search algorithm to find an optimal feasible word.

3.3 Lexi - Search Algorithm For Optimal Feasible Word

The following algorithm gives an optimal feasible word.

STEP 1: (Initialization) The arrays SN, D, DC, R, C, T and N are made available RI, CI, TI, SW, L, V, LB are initialized to zero. The values I = 1, J = 0, VT = 999, NZ = N*N*N-N, MAX = NZ-I. (I stands for i^{th} letter of the word)

STEP 2 : J = J+1 IS (J > MAX) IF YES GOTO 10
IF NO GOTO 3

STEP 3 : L (I) = J JA = J + N - I
V (1) = V (I-1) + D (J)
LB (I) = V(I) + DC (JA) - DC (J)
GOTO 4

STEP 4 : IS (LB (I) ≥ VT) IF YES GOTO 11
IF NO GOTO 5

STEP 5 : CHECK THE FEASIBILITY OF L (USING ALGORITHM) IF YES GOTO 2 IS (XI = 0)
IF NO GOTO 6

STEP 6 : IS (I = N) IF YES GOTO 9
IF NO GOTO 7

STEP 7 : L (I) = J RI {R(J)} = 1
CI {C(J)} =1
TI {T(J)}=1
SW {R (J)} = C (J)
GOTO 8

STEP 8 : I = I+1 MAX = MAX -1
GOTO 2

STEP 9 : L (I) = J L (I) is full length word and is feasible VT = V (I),
Record L (I), VT
GOTO 11

STEP 10 : IS (I=1) IF YES GOTO 13
IF NO GOTO 1
STEP 11 : I = I-1
MAX = MAX +1 GOTO 12
STEP 12 : J = L (I)
RI {R (J)} =0 CI {C(J)} =0
TI {T (J)} =0
SW {R(J)} =0
GOTO 2
STEP 13 : STOP
END

The current value of VT at the end of the search is the value of the Optimal feasible word. At the end if VT=999 it indicates that there is no feasible solution.

4. NUMERICAL ILLUSTRATION :

The concepts and the algorithm development will be illustrated by a numerical example for which n = 4 let N = [1,2,3,4] the cost array C (i, j, k) is given in the table-4.1.

Table -4.1 Cost Matrix.

$$C(i,j,1) = \begin{pmatrix} 999 & 0 & 3 & 28 \\ 16 & 999 & 41 & 0 \\ 40 & 0 & 999 & 5 \\ 27 & 0 & 21 & 999 \end{pmatrix} \qquad C(i,j,3) = \begin{pmatrix} 999 & 26 & 24 & 34 \\ 15 & 999 & 29 & 20 \\ 37 & 2 & 999 & 0 \\ 32 & 0 & 31 & 999 \end{pmatrix}$$

$$C(i,j,2) = \begin{pmatrix} 999 & 12 & 20 & 29 \\ 44 & 999 & 14 & 46 \\ 0 & 27 & 999 & 42 \\ 8 & 6 & 13 & 999 \end{pmatrix} \qquad C(i,j,4) = \begin{pmatrix} 999 & 30 & 36 & 42 \\ 8 & 999 & 0 & 25 \\ 17 & 7 & 999 & 50 \\ 19 & 5 & 38 & 999 \end{pmatrix}$$

In the numerical example given in Table -4.1, C (i,j,k) for i = j, i, j, k = 1,2,3,4 are taken to be very high number 999, because they are Irrelevant for finding tours of the salesman.

A 'tour' is a feasible trip- schedule for the salesman. Trip Schedule of the salesman can be represented by an appropriate $(n \times n \times n)$ indicator array. $X = \{x(i,j,k)\} \mid x(i,j,k) = 0 \text{ or } 1\}$ in which $x(i,j,k) = 1$ indicates that the salesman visits city j from city i at time (facility) k and if there is not such trip it is indicated by $x(i,j,k) = 0$. The indicator X given in Table -4.2 where the 4x4x4 of X is presented as four matrices for different values of $k\,\mathrm{I,e.,k} = 1,2,3,4$ to the numerical example.

Table -4.2 - Indicator Matrix

$$X(i,j,1) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad X(i,j,3) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$X(i,j,2) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad X(i,j,4) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The representation of the solution X to the problem is that the salesman visits city 4 from city 2 at time 1, visits city 3 from city 1 at time 2, visits city 1 from city 4 at time 3 and visits city 2 from city 3 at time 4. This solution is a feasible solution.

The tour corresponding to Table 3.2 can be presented as

$2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 2$

k = 1 ,k= 3 ,k = 2, k=4

Table 4.3 : Alphabet Table

S.No.	D	DC	R	C	T
1	0	0	1	2	1
2	0	0	2	4	1
3	0	0	3	2	1
4	0	0	4	2	1
5	0	0	3	1	2
6	0	0	3	4	3
7	0	0	4	2	3
8	0	0	4	2	3
9	2	2	3	2	3
10	3	5	1	3	1
11	5	10	3	4	1
12	5	15	4	2	4
13	6	21	4	2	2
14	7	28	3	2	4
15	8	36	4	1	2
16	8	44	2	1	4
17	12	56	1	2	2
18	13	69	4	3	2
19	14	83	2	3	2
20	15	98	2	1	3
21	16	114	2	1	1
22	17	131	3	1	4
23	19	150	4	1	4
24	20	170	1	3	2
25	20	190	2	4	3
26	21	211	4	3	1
27	24	235	1	3	3
28	25	260	2	4	4
29	26	286	1	2	3
30	27	313	3	2	2
31	27	340	4	1	1
32	28	368	1	4	1
33	29	397	1	4	2
34	29	426	2	3	3
35	30	456	1	2	4
36	31	487	4	3	3
37	32	519	4	1	3
38	34	553	1	4	3
39	36	589	1	3	4
40	37	626	3	1	3
41	38	664	4	3	4
42	40	704	3	1	1
S.No.	**D**	**DC**	**R**	**C**	**T**
43	41	745	2	3	1
44	42	787	3	4	2
45	42	829	1	4	4
46	44	873	2	1	2
47	46	919	2	4	2
48	50	969	3	4	4
49	999	1968	1	1	1
50	999	2967	2	2	1
51	999	3966	3	3	1
52	999	4965	4	4	1
53	999	5964	1	1	2
54	999	6963	2	2	2
55	999	7962	3	3	2
56	999	8961	4	4	2
57	999	9960	1	1	3
58	999	10959	2	2	3
59	999	11958	3	3	3
60	999	12957	4	4	3
61	999	13956	1	1	4

62	999	14955	2	2	4
63	999	15954	3	3	4
64	999	16953	4	4	4

Let us consider 22 ϵ SN. It represents the ordered triple.
[R (22), C (22), T (22)] = (3,1,4) then D (22) = C(3,1,4) = 17 and DC (22) = 122

Table -4.4 The arrays RI, CI, TI, SW and L for Partial Word L3 = (1. 6, 8)

	1	2	3	4
RI	1	1	1	0
CI	0	1	1	1
TI	1	0	1	1
SW	2	3	4	0
L	1	6	8	0

Table 4.5 Search - Table

The working details of getting an optimal word, using the algorithm 3.3 for the illustrated numerical example is given in the following table. The columns (1), (2), (3) and (4) gives the letters in the first, second, third and fourth places respectively. The corresponding V (I) and LB (I) are also indicated. Last column gives whether Rejected (R) or Accepted (A).

S.No.	1	2	3	4	V(I)	LB (I)	R	C	T	R/A
1	1				0	0	1	2	1	A
2		2			0	0	2	4	1	R
3		3			0	0	3	2	1	R
4		4			0	0	4	2	1	R
5		5			0	0	3	1	2	A
6			6		0	0	3	4	3	R
7			7		0	0	4	2	3	R
8			8		0	2	2	3	4	R
9			9		2	5	3	2	3	R
10			10		3	8	1	3	1	R
11			11		5	10	3	4	1	R
12			12		5	11	4	2	4	R
13			13		6	13	4	2	2	R
14			14		7	15	3	2	4	R
15			15		8	16	4	1	2	R
16			16		8	20	2	1	4	R
17			17		12	25	1	2	2	R
18			18		13	27	4	3	2	R
19			19		14	29	2	3	2	R
20			20		15	31	2	1	3	R
21			21		16	33	2	1	1	R
22			22		17	36	3	1	4	R
S.No.	**1**	**2**	**3**	**4**	**V(I)**	**LB (I)**	**R**	**C**	**T**	**R/A**
23			23		19	39	4	1	4	R
24			24		20	40	1	3	2	R
25			25		20	41	2	4	3	A
26				26	41	41	4	3	1	R
27				27	44	44	1	3	3	R
28				28	45	45	2	4	4	R
29				29	46	46	1	2	3	R
30				30	47	47	3	2	2	R
31				31	47	47	4	1	2	R
32				32	48	48	1	4	1	R
33				33	49	49	1	4	2	R
34				34	49	49	2	3	3	R
35				35	50	50	1	2	4	R
36				36	51	51	4	3	3	R
37				37	52	52	4	1	3	R
38				38	54	54	1	4	3	R
39				39	56	56	1	3	4	R
40				40	57	57	3	1	3	R
41				41	58	58	4	3	4	A=VT=58
42			26		21	45	4	3	1	R
43			27		24	49	1	3	3	R

44			28		25	51	2	4	4	A
45				29	51	51	1	2	3	R
46				30	52	52	4	1	1	R
47				31	52	52	3	2	2	R
48				32	53	53	1	4	1	R
49				33	54	54	1	4	2	R
50				34	54	54	2	3	3	R
51				35	55	55	1	2	4	R
52				36	56	56	4	3	3	A=VT=56
53			29		26	53	1	2	3	R
54			30		27	54	4	1	1	R
55			31		27	55	3	2	2	R
56			32		28	57	1	4	1	R>VT
57		6			0	0	3	4	3	A
58			7		0	0	4	2	3	R
59			8		0	2	2	3	4	A
60				9	2	2	3	2	3	R
61				10	3	3	1	3	1	R
62				11	5	5	3	4	1	R
63				12	5	5	4	2	4	R
64				13	6	6	4	2	2	R
65				14	7	7	3	2	4	R
66				15	8	8	4	1	2	A=VT=8
67			9		2	5	3	2	3	R
68			10		3	8	1	3	1	R=VT
69			11		5	10	3	4	1	R>VT
70		7			0	2	4	2	3	R
71		8			0	5	2	3	4	A
72			9		2	5	3	2	3	R
73			10		3	8	1	3	1	R
74			11		5	10	3	4	1	R>VT
75		9			2	10	3	2	3	A
76	2				0	0	2	4	1	R
77		3			0	0	3	2	1	R
78		4			0	0	4	2	1	R
79		5			0	0	3	1	2	A
80			6		0	0	3	4	3	R
81			7		0	0	4	2	3	R
82			8		0	2	2	3	4	R
83			9		2	5	3	2	3	R
84			10		3	8	1	3	1	R
85			11		5	10	3	4	1	R>VT
86		6			0	0	3	4	3	R
S.No.	**1**	**2**	**3**	**4**	**V(I)**	**LB (I)**	**R**	**C**	**T**	**R/A**
87		7			0	2	4	2	3	R
88		8			0	5	2	3	4	R
89		9			2	10	3	2	3	R>VT
90	3				0	0	3	2	1	A
91		4			0	0	4	2	1	R
92		5			0	0	3	1	2	R
93		6			0	0	3	4	3	R
94		7			0	2	4	2	3	R
95		8			0	5	2	3	4	R
96		9			2	10	3	2	3	R>VT
97	4				0	0	4	2	1	A
98		5			0	0	3	1	2	A
99			6		0	0	3	4	3	R
100			7		0	0	4	2	3	R
101			8		0	2	2	3	4	R
102				9	2	2	3	2	3	R
103				10	3	3	1	3	1	R
104				11	5	5	3	4	1	R
105				12	5	5	4	2	4	R
106				13	6	6	4	2	2	R
107				14	7	7	3	2	4	R
108				15	8	8	4	1	2	R=VT

109			9		2	5	3	2	3	R
110			10		3	8	1	3	1	R
111			11		5	10	3	4	1	R>VT
112		6			0	0	3	4	3	A
113			7		0	0	4	2	3	R
114			8		0	2	2	3	4	R
115			9		2	5	3	2	3	R
116			10		3	8	1	3	1	R
117			11		5	10	3	4	1	R>VT
118		7			0	2	4	2	3	R
119		8			0	5	2	3	4	A
120			9		2	5	3	2	3	R
121			10		3	8	1	3	1	R
122			11		5	10	3	4	1	R>VT
123		9			2	10	3	2	3	R>VT
124	5				0	0	3	1	2	A
125		6			0	0	3	4	3	R
126		7			0	2	4	2	3	A
127			8		0	2	2	3	4	A
128				9	2	2	3	2	3	R
129				10	3	3	1	3	1	R
130				11	5	5	3	4	1	R
131				12	5	5	4	2	4	R
132				13	6	6	4	2	2	R
133				14	7	7	3	2	4	R
134				15	8	8	4	1	2	R=VT
135			9		2	5	3	2	3	R
136			10		3	8	1	3	1	R
137			11		5	10	3	4	1	R>VT
138		8			0	5	2	3	4	A
139			9		2	5	3	2	3	R
140			10		3	8	1	3	1	R
141			11		5	10	3	4	1	R>VT
142		9			2	10	3	2	3	R>VT
143	6				0	2	3	4	3	A
144		7			0	2	4	2	3	R
145		8			0	5	2	3	4	A
146			9		2	5	3	2	3	R
147			10		3	8	1	3	1	R
148			11		5	10	3	2	3	R>VT
149		9			2	10	3	2	3	R>VT
150	7				0	5	4	2	3	A
S.No.	**1**	**2**	**3**	**4**	**V(I)**	**LB (I)**	**R**	**C**	**T**	**R/A**
151		8			0	5	2	3	4	A
152			9		2	5	3	2	3	R
153			10		3	8	1	3	1	R
154			11		5	10	3	4	1	R>VT
155		9			2	10	3	2	3	R>VT
156	8				0	10	2	3	4	R>VT

The partial word in 25th row is L_3 = (1,5,25) which is partially feasible. For this partial word the arrays RI, CI, TI, SW and L are given as the following Table -4.6

Table -4.6 The arrays RI, CI, TI, SW and L for feasible Partial Word L_3=(1, 5, 25)

	1	2	3	4
RI	1	1	1	0
CI	1	1	0	1
TI	1	1	1	0
SW	1	4	1	0
L	1	5	25	0

At the end of the search the current value of VT is 8 and it is the value of the optimal feasible word. The optimal feasible word is L_4 = (1,6,8,15), it is given in 66th row of the search table. The pattern represented by the above optimal feasible word is represented in the following Table 4.7.

Table 4.7 Optimal Feasible Solution

$$X(i,j,1) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad X(i,j,2) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$X(i,j,3) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad X(i,j,4) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The Tour represented by the above pattern is [(1,2,1), (4,1,2), (3.4,3), (2,3,4)] where the salesman from city 1 goes to city 2 at time 1, from city 2 goes to city 3 at time 4, from city 3 goes to city 4 at time 3 and from city 4 goes to city 1 at time 2. It can be seen that all cities and all times are involved in the optimal solution for the tour. It also can be represented by

$$1 \underset{k=1}{\rightarrow} 2 \underset{k=4}{\rightarrow} 3 \underset{k=3}{\rightarrow} 4 \underset{k=2}{\rightarrow} 1$$

5. CONCLUSION :

In the Lexicographic search the structure of search algorithm does not require huge dynamic memory, so for large problems this method appears to be relatively smaller than to branch and bound method. The lexicographic algorithm thus developed in this paper , surely gives optimal tour of the salesman. As compared to the exponential space for branch and bound algorithm the lexicographic search takes a linear space .The procedure developed by others requires more variable and more constraint, to solve time dependent traveling salesman problem. The procedure developed by author takes care of the simple combinatorial structure of traveling salesman problem. So have lesser calculations. A computer program may be developed according to algorithm and the algorithm may be extended for Assignment problem also.

REFERENCES

[1] Affenzeller, M., Wanger, S., (2003) "A self-adaptive model for selective pressure handling within the theory of genetic algorithms", EUROCAST 2003, Las Palmas de Gran Canaria, Spain, Lect. Notes Comp. Sci. 2809 (1), 384–393.

[2] Bianchi, L., Knowles, J., Bowler, J., (2005)" Local search for the probabilistic traveling salesman problem: Correction to the 2- p-opt and 1-shift algorithms", Eur. J. of Oper. Res. 162(1), 206 219.

[3] Bianco L.,Mingozzi S., (1994)"Exact and heuristic procedures for the TSP with precedence constraints, based on dynamic programming" Information Systems &Operational Research,19-32,.

[4] Chu, S.C., Roddick, J.F., Pan, J.S., (2004)" Ant colony system with communication strategies", Inform. Sci. 167 (1–4), 63– 76.

[5] Das S.,Ahmed N. (2001),"A travelling salesman problem with multiple job facilities" Opsearch 38(4), 394-406

[6] Gutin, G., Punnen, A.P. (Eds.) (2002) "The Traveling Salesman Problem and its Variations". Kluwer, Dordrecht, 169–194 (Chapter 9).

[7] Liu, G., He, Y., Fang, Y., Oiu, Y., (2003)" A novel adaptive search strategy of intensification and diversification in tabu search", in: Proceedings of Neural Networks and Signal Processing, Nanjing, China.

[8] Onwubolu, G.C. and Clerc, M. (2004)," Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization", Int.J. Prod. Res. 42 (3), 473–491.

[9] Shinano, Y.,Inui, N., Fukagawa, Y. and Takakura, N., (2008) "An Application of Traveling- Salesman Models to Shot Sequence Generation for Scan Lithography", 5th European Congress on Computational Methods in Applied Sciences and Engineering, 30 , Venice, Italy.

[10] Sobhan Babu,K & Sundara Murthy.M, (2010), "An efficient algorithm for Variant Bulk Transportation Problem", International Journal of Engineering Science and Technology, Vol.2(7), pp.2595-2600.

[11] Sobhan Babu,K ,Keshava redid.E & Sundara Murthy.M(2012)"an exact algorithm for travelling salesman problem" Vol.3(1),pp.317-321.

[12] Turkensteen, M., Ghosh, D., Goldengorin, B., Sierksma, G(2007) "Tolerance-based Branch and Bound algorithms for the ATSP", European Journal of Operational Research 189: 775–788.

[13] Venkatadri Naidu Ch.,Mudar Valli P.,Sita Ram Raju, (2010),"Optimization of service operations sequencing a four wheeler service centre",Int. Jour. Of Engg. & Sc. And Technology; vol.2(6), 2321-2327.

[14] Vijyalaxmi G. (2013)"Lexicographic approach to travelling salesman problem"IOSR Journal of Mathematics Vol 6(4), 01-08.