ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

An Alternate Travelling Salesman Problem

P. Madhu Mohan Reddy[*1], E. Sudhakara[1], S. Sreenadh[1], S. V. K. Varma[1]
[*1] Department of Mathematics, Sri Venkateswara University, Tirupati-517 502, Andhra Pradesh, India
mmrphdsv@gmail.com

Abstract

We consider Lexi-Search Approach using Pattern Recognition Technique for a Travelling Sales Man Problem (TSP) in which he wants to visit m cities, where m is even. Let N be the set of n stations defined as N= {1, 2, 3, 4…n} and N1UN2=N. The city '1' taken as the home city and it is in N1. He has to starts from head quarter city {1} which is in N1 from there he visits a city in N2. In this way the salesman visits m cities alternatively and m ≤ n. D (i, j) be the distance or cost matrix. A salesman starts for his tour from a home city (say 1) and come back to it after completing the all the m cities. There is a restriction that he must visit the N1, N2 groups alternatively. An exact algorithm is proposed for this TSP. The algorithm solves the problem on identify the key patterns which optimize the objective of the cost/distance. Hence the objective of the problem is to find a tour with minimum total distance while completing all the m cities alternatively by above considerations.

Keywords: Pattern Recognition, Lexi-Search Approach, Travelling Sales Man Problem.

Introduction

The travelling salesman problem is one of the oldest combinatorial programming problems (Flood[2]-1956 and Croes[3]-1958) and can be stated as follows. There are n cities and the distance between any ordered pair of cities is known. Starting from one of the cities a salesman is to visit the other cities only once and return to the starting city. The objective is to find a tour in such a way that the total length of the tour is minimum. The Traveling Salesman Problem is one of the most intensively studied problems in computational mathematics and is a kind of mathematical puzzle with a long enough history Dantzig, Fulkerson and Johnson-1954 and Tutte-1940. Many solution procedures have been developed such as Flood [2]-1956, Hardgrave & Nembanser[12]-1962 and Little[13] et al-1963 for the travelling salesman problem. One more generalization which is called the "Truncated Travelling Salesman Problem" (Sundara Murthy [9] -1979).

There are many algorithms for usual one man travelling salesman problem developed by researchers from time to time. But the problem has not received much attention in its restricted context. However, literature which is available with regard to the TSP with variations is discussed (Das [7] - 1976, 78, Jaillet [8] - 1985, Pandit[5] - 1961, 63, 64, 65, Murthy & Srivastava et al [10] 1969. The Generalized Travelling Salesman Problem was first addressed by Srivastava[10] et al (1970).Time Dependent Travelling Salesman Problem Was also attempted by Bhavani[11] – 1997, Sobhan Babu[6] – 2000, Naganna[4] – 2001 and Balakrishna[1] – 2006 and the simple combinatorial structure of the TDTSP were taken into account. With sufficient ingenuity, one can also formulate problems of this type as a non trivial Integer programming problem, as a zero–one programming problem more generally (Balas et.al. – 1991 and Glover[14] – 1965). Combinatorial structure may not be clear in some problems because of the NP–Hard nature, but one can always identify the relevant Cartesian product and the feasibility criterion

Problem Description

In this paper we consider Pattern Recognition Based Lexi-Search Approach for A Travelling Sales Man Problem (TSP) in which he wants to visit m cities. The objective is to find a Tour such that total length of the tour minimum. An exact algorithm is proposed for this TSP. The algorithm solves the problem on identify the key patterns which optimize the objective of the cost/distance. The algorithm calculates the solution incrementally for deferent cities have visited by the salesman and the best combination is taken as the solution.

Let N be the set of n stations defined as N= {1, 2, 3, 4…n} and N_1UN_2=N. The city '1' taken as the home city and in N_1 . He has to starts from head quarter city {1} which is in N_1 from there he visits a city in $N_{2.}$ In this way the salesman visits m cities alternatively and m ≤ n. D be the distance or cost matrix associated with the node pair of cities i and j. A salesman starts for his tour from a home city (say 1) and come back to it after completing the all the m cities. There is a restriction that he must visit the N_1, N_2 groups alternatively. Hence the objective of the problem is to find a tour with minimum total distance while completing all the m cities alternatively, where m is even.

Mathematical Formulations

Let N= {1, 2…n} , {1}εN_1, $N_1 U N_2$ =N and $N_1 \cap N_2 = \phi$. For our convenience odd cities can be taken into N_1 and even cities can be taken into N_2 .This is not a necessary condition. Many times we have freedom to name the cities, so we can name the N_1 cities as 1, 3, 5…and N_2 cities 2, 4, 6…as numbers.

$$Minimize\ Z = \sum_{i\in N}\sum_{j\in N} DV(i,j)\,X(i,j) \quad \text{------------------------------} \ (1)$$

$$subject\ to\ \sum_{i\in N}\sum_{j\in N} X(i,j) = m, \ (\text{m is even}),\ m\le n \text{-------------} \ (2)$$

$$X(i,j) = 0\ or\ 1 \quad \text{----------------------} \ (3)$$

$If\ X(i,j)$=1, then, (iε N_1, jε N_2) or (iε N_2, jε N_1) ………………(4)

Equation (1), represent the total length of the tour. Equation (2) represent the salesman travelled total m cities, ≤n. Equation (3) represent the salesman travelled from city i to city j then the value is 1 otherwise it is 0.

Numerical Illustration

The concepts and the algorithm developed will be illustrated by a numerical example for which we have to take number of cities/stations n = 6, and number of stations travelled by the salesman m =6. Let N = {1, 2, 3, 4, 5, 6}. Then the cost/distance DV(i,j) ,i.e ,array DV is given in table-1.

Table – 1

DV(i,j)=

999	4	1	1	2	4
2	999	2	3	1	∞
6	1	999	7	8	2
9	∞	4	999	4	1
3	3	2	8	999	4
4	1	2	1	3	999

In the table-1 numerical example, DV (i, j), where i=j are taken to be very large number 999, because they are irrelevant for finding a tour of the salesman. Though all the DV (i, j)'s taken as non negative integers it can be easily seen that this is not a necessary condition and the cost/distance can be any positive quantity. DV (3, 4) = 7 means that the cost /distance of the connecting the city 3 to 4 is 7.

Table-2

I	1	2	3	4	5	6
INX	1	2	1	2	1	2

Where INX (i) = 1; if $i\in N_1$,
= 2; if $i\in N_2$

Concepts and Definitions

A tour is a feasible trip - schedule for the salesman. Trip-schedule of the salesman can be represented by an appropriate n x n indicator array X = {x (i, j); x (i, j) = 0 or 1} in which x (i, j) = 1 indicates that the salesman visits city j from city i .

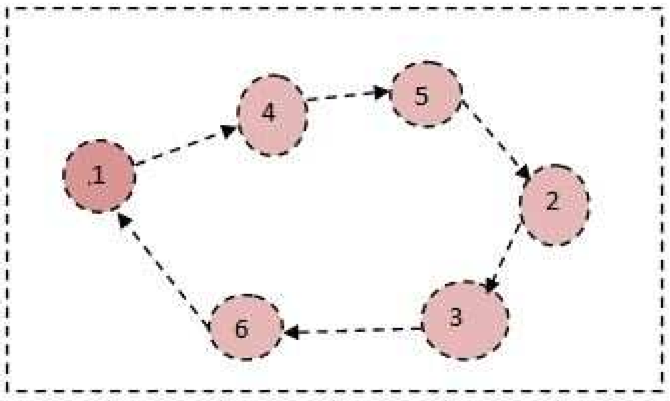
Feasible Trip:{(1,4),(4,5),(5,2),(2,3),(3,6),(6,1)} represents the pattern given in Table - 3, which is feasible .

Table - 3

X(i,j)=

0	0	0	1	0	0
0	0	1	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	1	0	0	0	0
1	0	0	0	0	0

FIGURE-1



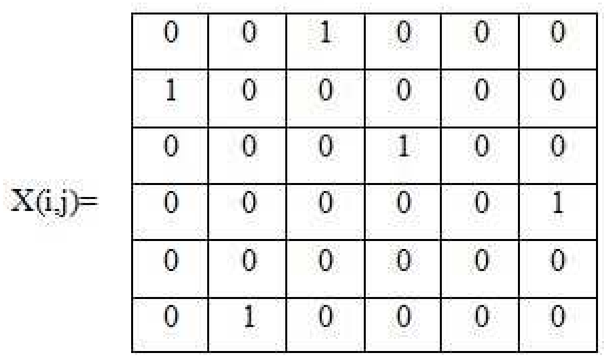
From the above **figure 1** city 1 is connected to the city 4,city 4 is connected to the city 5, city 5 is connected to the city2, city 2 is connected to the city 3, city 3is connected to the city 6 and city 6 is connected to the city1. So it is a feasible solution.

ISSN: 2277-9655
Impact Factor: 1.852

Infeasible Trip

Consider an ordered pair set {(1,3), (3,4), (4,6), (6,2) and (2,1)} represents the pattern given in the **Table - 4**, which is an infeasible solution.

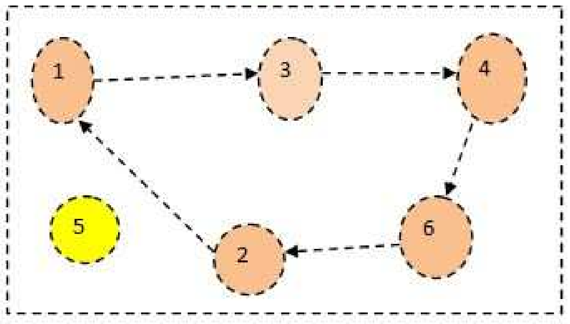
Table – 4



X(i,j)=

0	0	1	0	0	0
1	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	1
0	0	0	0	0	0
0	1	0	0	0	0

FIGURE-2



From the above **figure 2**, it is infeasible due to (i) The salesman visits only 5 cities,<6=m, here city 5 is missed (ii) He visits 4 to 6 and 1 to 3, this is contradiction because the cities 4,6 and 1, 3 belongs to the same group N_2 and N_1 respectively.

5.3 Definition of a Pattern

An indicator two-dimensional array which is associated with an assignment is called a 'pattern'. A Pattern is said to be feasible if X is a solution.

$$V(X) = \sum_{i \in I} \sum_{j \in J} DV(i,j)X(i,j)$$

The value V(x) is gives the total cost/distance of the tour for the solution represented by X. The pattern represented in the table-3 is a feasible pattern. 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 represented by the set of ordered pairs [(i, j)] for which X (i, j) =1, with understanding that the other X (i, j)'s zeros. The ordered pair se{(1,4),(4,5),(5,2),(2,3),(3,6),(6,1)} } represents the pattern given in **table -3,** which is feasible and the ordered pair set {(1,3), (3,4), (4,6), (6,2), (2,1) } represents the pattern given in the **Table –4**, which is a infeasible solution.

There are $M = n^2$ ordered pairs in the two-dimensional array X. For convenience these are arranged in ascending order of their corresponding cost/distance and are indexed from 1 to M (Sundara Murthy-1979). Let SN= [1, 2, 3... n^2] be the set of n^2 indices. Let D be the corresponding array of cost. If a, b∈ SN and a < b then D (a)$\leq$ D (b). Also let the arrays R, C be the array of row and column indices of the ordered pair represented by SN and DC be the array of cumulative sum of the elements of D. The arrays SN, D, DC, R, and C for the numerical example are given in the **table-5**. If p∈ SN then (R (p), C (p)) is the ordered pair and D (a) = DV(R (a), C (a)) is the value of the ordered pair and $DC(a) = \sum_{i=1}^{a} D(i)$.

Table-5

SN	D	DC	R	C
1	1	1	1	3
2	1	2	1	4
3	1	3	2	5
4	1	4	3	2
5	1	5	4	6
6	1	6	6	2
7	1	7	6	4
8	2	9	1	5
9	2	11	2	1
10	2	13	2	3
11	2	15	3	6
12	2	17	5	3
13	2	19	6	3
14	3	22	2	4
15	3	25	5	1
16	3	28	5	2
17	3	31	6	5
18	4	35	1	2
19	4	39	1	6

ISSN: 2277-9655
Impact Factor: 1.852

20	4	43	4	3
21	4	47	4	5
22	4	51	5	6
23	4	55	6	1
24	6	61	3	1
25	7	68	3	4
26	8	76	3	5
27	8	84	5	4
28	9	93	4	1

Let us consider 11∈ SN. It represents the ordered pair (R (11), C (11)) = (3, 6).
Then D (11) = DV (3, 6) = 2, DC (11) =15.

Definition of an Alphabet – Table and a Word

Let SN = (1, 2…) be the set of indices, D is an array of corresponding costs/distances of the ordered pair. Let arrays R, C be respectively, the row and column indices of the ordered pair. Let $L_k = \{a_1, a_2, - - -- - , a_k\}$, $a_i \in$ SN be an ordered sequence of k indices from SN. The pattern represented by the ordered pair 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 $a_i \leq a_{i+1}$, i = 1, 2, - - - -, k-1. The set SN is defined as the "Alphabet-Table" with alphabetic order as (1, 2, - - - -, n^2) and the ordered sequence L_k is defined as a "word" of length k. A word L_k is called a "sensible word". If $a_i < a_{i+1}$, 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 pattern. 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. Our interest is only in set of words of length at most equation, since the words of length greater than n are necessarily infeasible, as any feasible pattern can have only n unit entries in it. If k < n, L_k is called a partial word and if k = m, it is a full length word or simply a word. A partial word L_k represents, a block of words with L_k as a leader i.e. as its first k letters. A leader is said to be feasible, if the block of word, defined by it has at least one feasible word.

Value of The Word

The value of the (partial) word L_k, V (L_k) is defined recursively as V (L_k) = V (L_{k-1}) + D (a_k) with V (L_o) = 0 where D (a_k) is the cost array arranged such that D (a_k) < D (a_{k+1}). V (L_k) and V(X) the values of the pattern X will be the same. Since X is the (partial) pattern represented by L_k, (Sundara Murthy – 1979).

For example the word L_3 = (1, 5, 9) then value of L_3 is V (L_3) = 1+2+3 = 6 and V(X) = 6

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.

Feasibility criterion of a partial word

An algorithm is developed for checking the feasibility of partial word L_{k+1} =($\alpha_1,\alpha_2,\alpha_3, ...\alpha_k,\alpha_{k+1}$), given that L_k is a feasible partial word. We will introduce more notations which will be useful in the sequel.
IR be an array where IR (i) = 1, i∈ N indicates that the sales man is visiting city from city i. Otherwise R (i) = 0.IC be an array where IC (i) = 1, i∈ N indicates that the sales man is coming to city i from another city, otherwise IC (i) = 0.SW be an array where SW (i) = j indicates that the sales man is visiting the city j from city i. Otherwise SW (i) = 0. L be an array where L[i] = α_i, i∈ N is the letter in the ith position of a word.
Then for a given partial word $L_k = (\alpha_1,\alpha_2,\alpha_3, ...\alpha_k,)$, the values of the arrays IR, IC, SW, and LW are defined as follows.

- IR (IR (α_i)) = 1, i = 1, 2, - - - - - , k and IR (i) = 0 for other elements of i
- IC (C (α_i)) = 1, i = 1, 2, - - - - - , k and IC (j) = 0 for other elements of i
- SW(R (α_i)) = C (α_i), i = 1, 2, - - - , k and SW (j) = 0 for other elements of i
- L (i) = α_i , i = 1, 2, - - - - -, k, and L (j) = 0, for other elements of i.

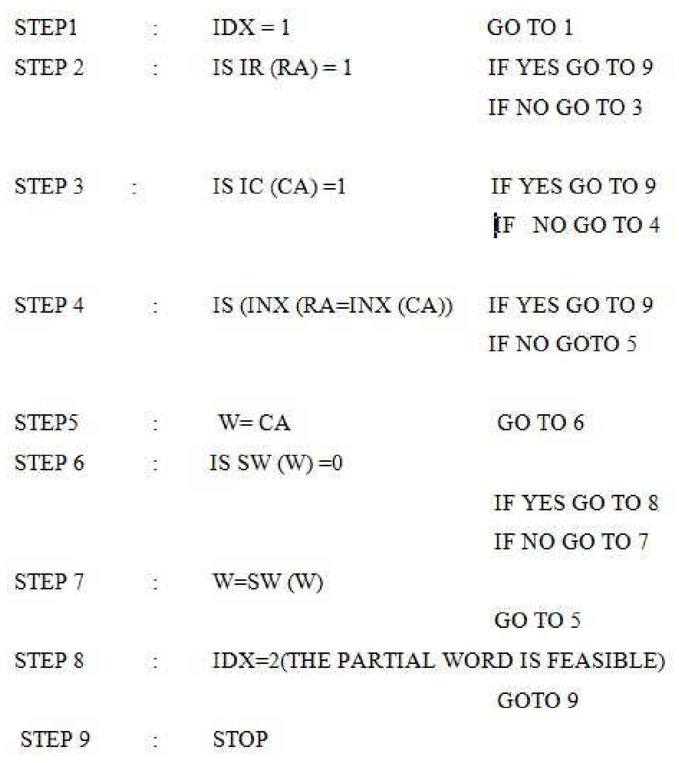
For example consider a sensible partial word L_4 = (1, 4, 5, 9) which is feasible. The array IR, IC, L, SW, takes the values represented in **table - 6** given below.

Table – 6

	1	**2**	**3**	**4**	**5**	**6**
L	1	4	5	9	0	0
IR	1	1	1	1	0	0
IC	1	1	1	0	0	1
SW	3	1	2	6	0	0

The recursive algorithm for checking the feasibility of a partial word L_p is given as follows In the algorithm first we equate IDX =1. At the end if IDX = 2, then the partial word is feasible, otherwise it is infeasible. For this algorithm we have RA = R (a_{p+1}) and CA = C (a_{p+1})

Algorithm-1



```
STEP1    :  IDX = 1                       GO TO 1
STEP 2   :  IS IR (RA) = 1                IF YES GO TO 9
                                          IF NO GO TO 3

STEP 3   :  IS IC (CA) =1                 IF YES GO TO 9
                                          IF  NO GO TO 4

STEP 4   :  IS (INX (RA=INX (CA))         IF YES GO TO 9
                                          IF NO GOTO 5

STEP5    :  W= CA                         GO TO 6
STEP 6   :  IS SW (W) =0
                                          IF YES GO TO 8
                                          IF NO GO TO 7
STEP 7   :  W=SW (W)
                                          GO TO 5
STEP 8   :  IDX=2(THE PARTIAL WORD IS FEASIBLE)
                                          GOTO 9
STEP 9   :  STOP
```

We start with the partial word L_1 = (a_1) = (1). A partial word L_k is constructed as $L_k = L_{k-1}$ * (α_k). Where * indicates chain formulation. We will calculate the values of V (L_k) and LB (L_k) simultaneously. Then two situations arises one for branching and other for continuing the search.

1. LB (L_k) < VT. Then we check whether L_k is feasible or not. If it is feasible we proceed to consider a partial word of under (k+1). Which represents a sub-block of the block of words represented by L_k? If L_k is not feasible then consider the next partial word k by taking another letter which succeeds a_k in the position. If all the words of order k are exhausted then we consider the next partial word of order (k-1).
2. LB (L_k) $\geq$ VT. In this case we reject the partial word L_k. We reject the block of word with L_k as leader as not having optimum feasible solution and also reject all partial words of order k that succeeds L_k

Algorithm – 2

The following algorithm gives an optimal feasible word.

```
STEP 0  :   Initialization
                The arrays SN, D, DC, R, C and INX are made available. K (=which represents position in a word)
                =1. J (=which takes the letters in SN) =0, $L_0$=Φ (A null sequence), the arrays IR, IC, SW and L are
                initialized to zero. VT=9999, N=6, H=$N^2$-N+K. V ($L_0$) =0.
STEP 1  :   J = J+1
                IS J≤H,                                IF YES, GO TO 2
                                                       IF NO GO TO 7.

STEP 2  :   $L_k = L_{k-1}$*(J), V ($L_k$) =V ($L_{k-1}$) +D (J)
                LB ($L_k$) =V ($L_k$) +DC (J+N-K)-DC (J)
                IS LB ($L_k$) ≥VT,                     IF YES, GO TO 7
                                                       IF NO GO TO 3
STEP 3  :   check the feasibility of $L_k$(using algorithm 1)
```

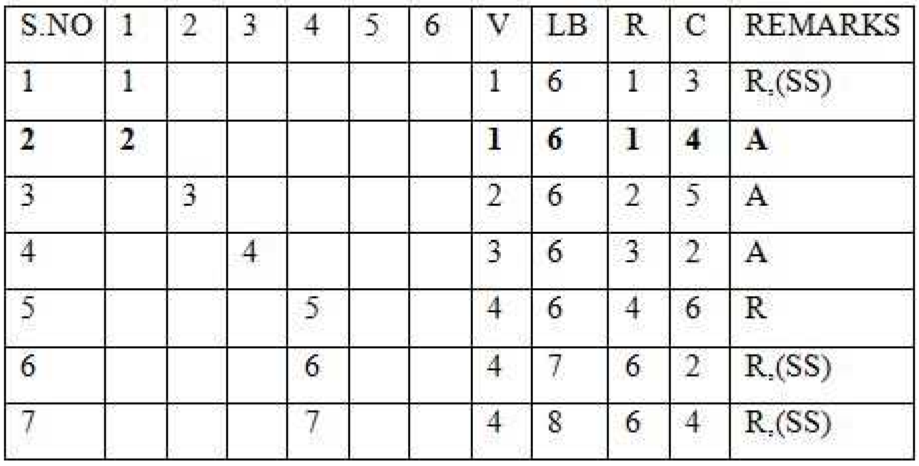
```
                IS IDX=0,                                 IF YES GO TO 1
                                                          IF NO GO TO 4
STEP 4  :   IS K=N                                        IF YES GO TO 6
                                                          IF NO GO TO 5
STEP 5  :   IR (R(J))=1, IC(C(J))=1, SW(R(A))=C(J), LW(K)=J, K=K+1,
                $H=N^2-N+K$
                                                          GO TO 1
STEP 6  :   $L_k$ is a full length word (i.e. k=6). VT=V ($L_k$), record $L_k$ and
                VT; GO TO 8
STEP 7  :   IS K = 1                                      IF YES GO TO 9 (search
                                                          is over).
                                                          IF NO GOTO 8
STEP 8: IS K=K-1, J=LW(k),IR(R(J))=0, IC (C(J))=0,LW(K)=0,  SW(R(J))=0, $H=N^2-N+k$,      GO TO 1

STEP 9  :   STOP.   (The current value of VT, when the search terminate
                        give the value of an optimal feasible word).
```

Search-Table

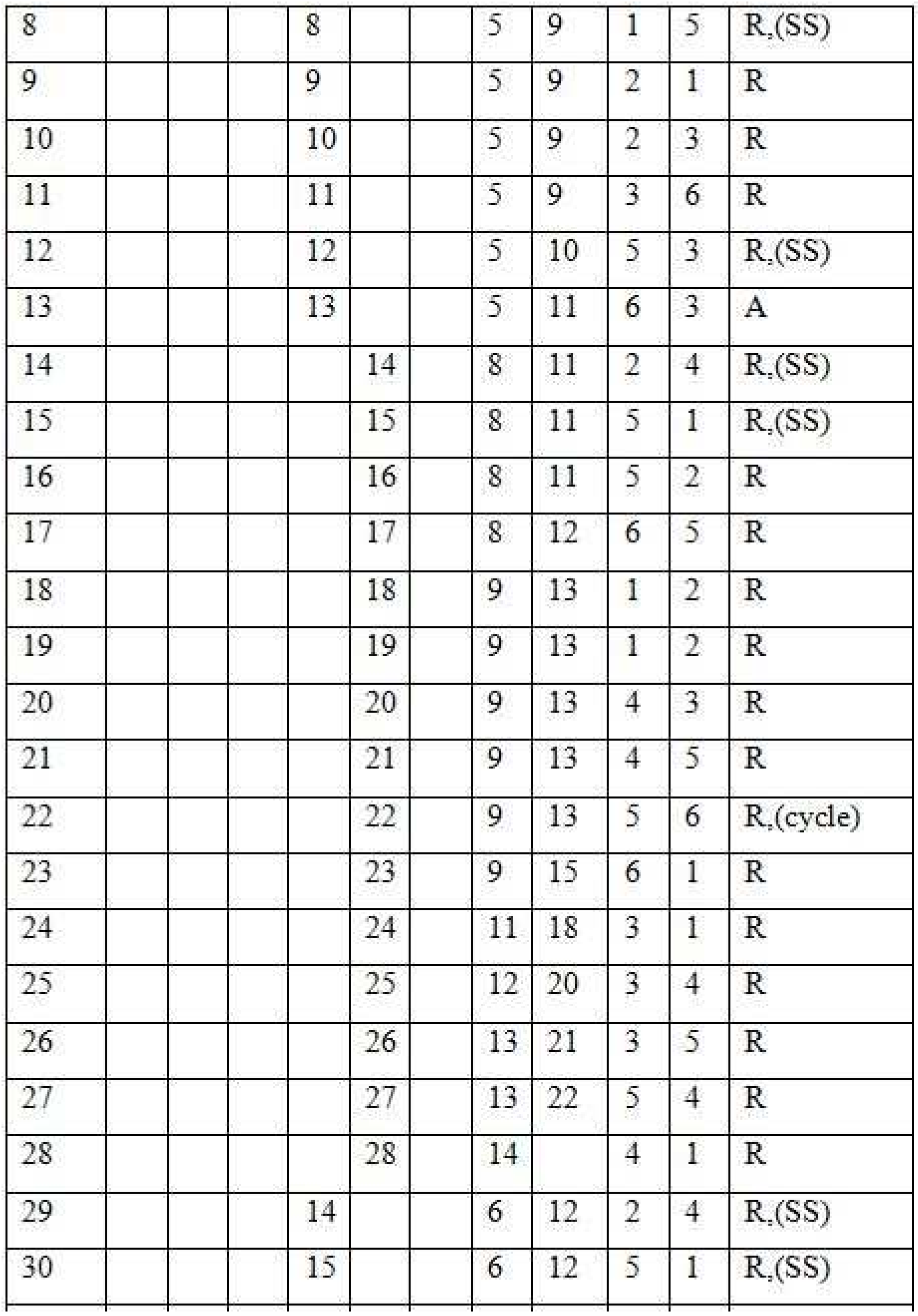
The working details of getting an optimal word using the above algorithm for the illustrative numerical example is given in the **Table - 7**. The columns named (1), (2), (3),…, gives the letters in the first, second, third and so on places respectively. The columns R, C and T give the row, column and facility indices of the letter. The last column gives the remarks regarding the acceptability of the partial words. In the following table A indicates ACCEPT and R indicates REJECT.SS indicates same set or same group.

Search Table-7

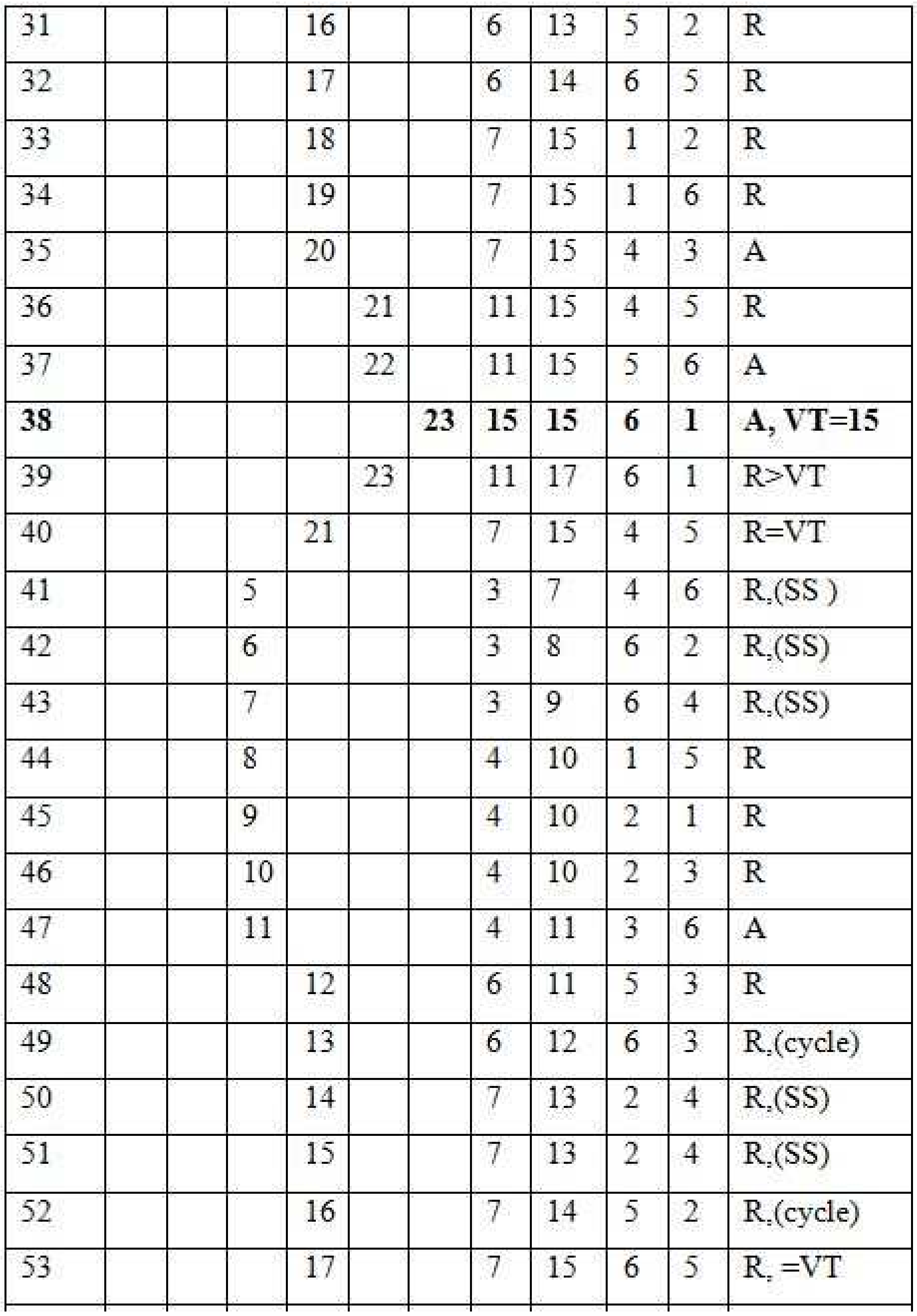


S.NO	1	2	3	4	5	6	V	LB	R	C	REMARKS
1	1						1	6	1	3	R.(SS)
2	**2**						**1**	**6**	**1**	**4**	**A**
3		3					2	6	2	5	A
4			4				3	6	3	2	A
5				5			4	6	4	6	R
6				6			4	7	6	2	R.(SS)
7				7			4	8	6	4	R.(SS)

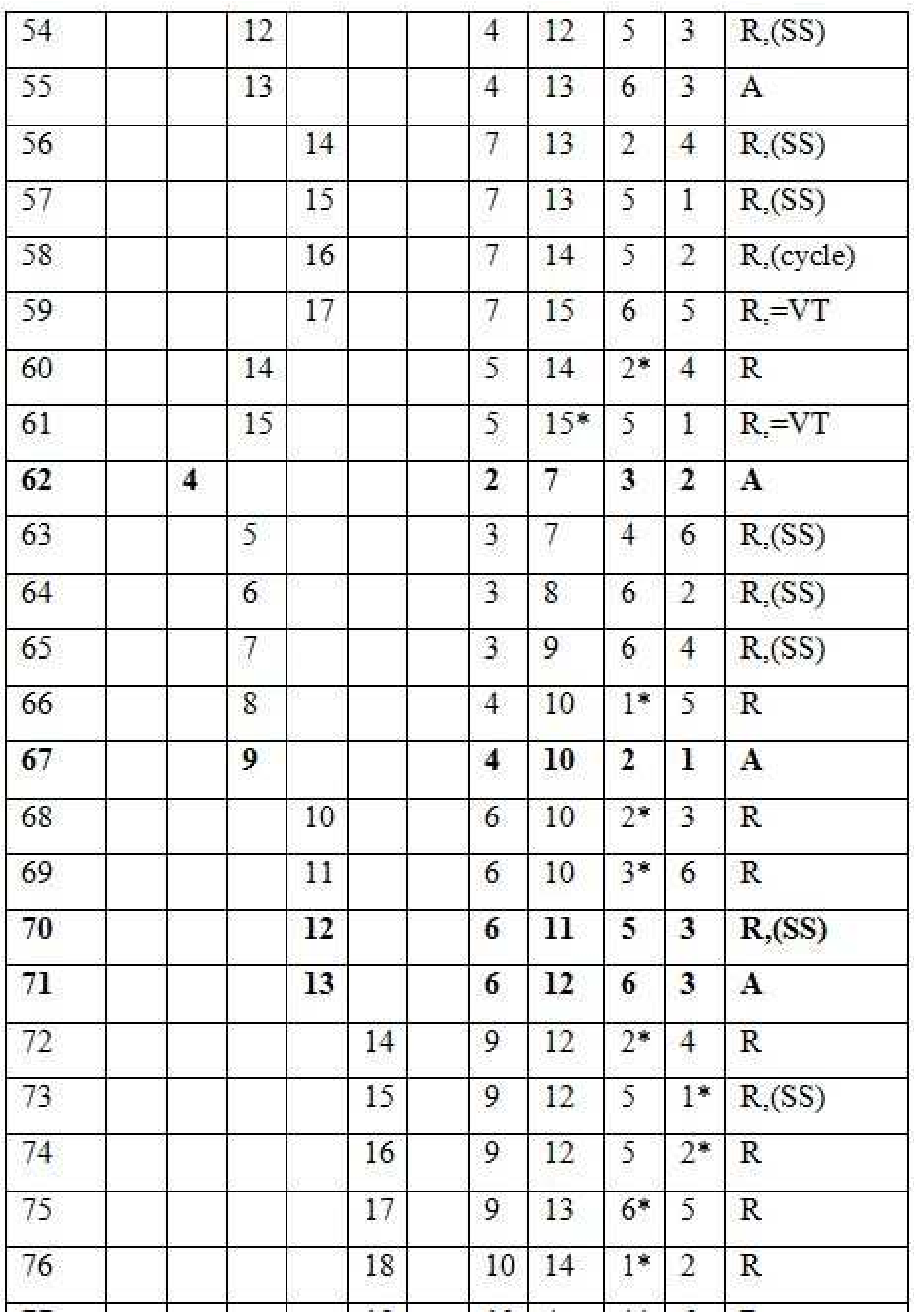
ISSN: 2277-9655
Impact Factor: 1.852



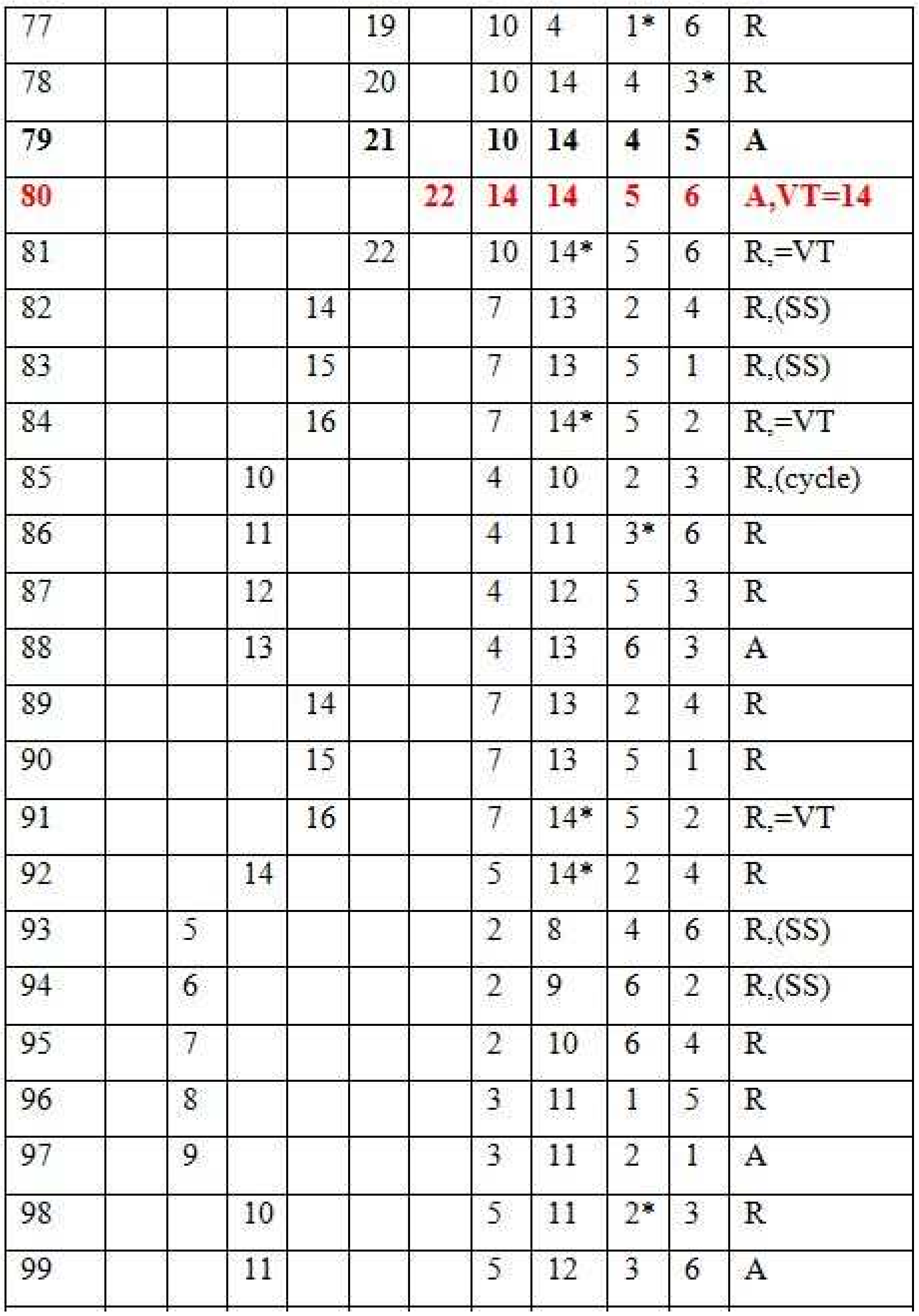
8				8			5	9	1	5	R,(SS)
9				9			5	9	2	1	R
10				10			5	9	2	3	R
11				11			5	9	3	6	R
12				12			5	10	5	3	R,(SS)
13				13			5	11	6	3	A
14					14		8	11	2	4	R,(SS)
15					15		8	11	5	1	R,(SS)
16					16		8	11	5	2	R
17					17		8	12	6	5	R
18					18		9	13	1	2	R
19					19		9	13	1	2	R
20					20		9	13	4	3	R
21					21		9	13	4	5	R
22					22		9	13	5	6	R,(cycle)
23					23		9	15	6	1	R
24					24		11	18	3	1	R
25					25		12	20	3	4	R
26					26		13	21	3	5	R
27					27		13	22	5	4	R
28					28		14		4	1	R
29				14			6	12	2	4	R,(SS)
30				15			6	12	5	1	R,(SS)



31				16			6	13	5	2	R
32				17			6	14	6	5	R
33				18			7	15	1	2	R
34				19			7	15	1	6	R
35				20			7	15	4	3	A
36					21		11	15	4	5	R
37					22		11	15	5	6	A
38						**23**	**15**	**15**	**6**	**1**	**A, VT=15**
39					23		11	17	6	1	R>VT
40				21			7	15	4	5	R=VT
41			5				3	7	4	6	R,(SS)
42			6				3	8	6	2	R,(SS)
43			7				3	9	6	4	R,(SS)
44			8				4	10	1	5	R
45			9				4	10	2	1	R
46			10				4	10	2	3	R
47			11				4	11	3	6	A
48				12			6	11	5	3	R
49				13			6	12	6	3	R,(cycle)
50				14			7	13	2	4	R,(SS)
51				15			7	13	2	4	R,(SS)
52				16			7	14	5	2	R,(cycle)
53				17			7	15	6	5	R, =VT

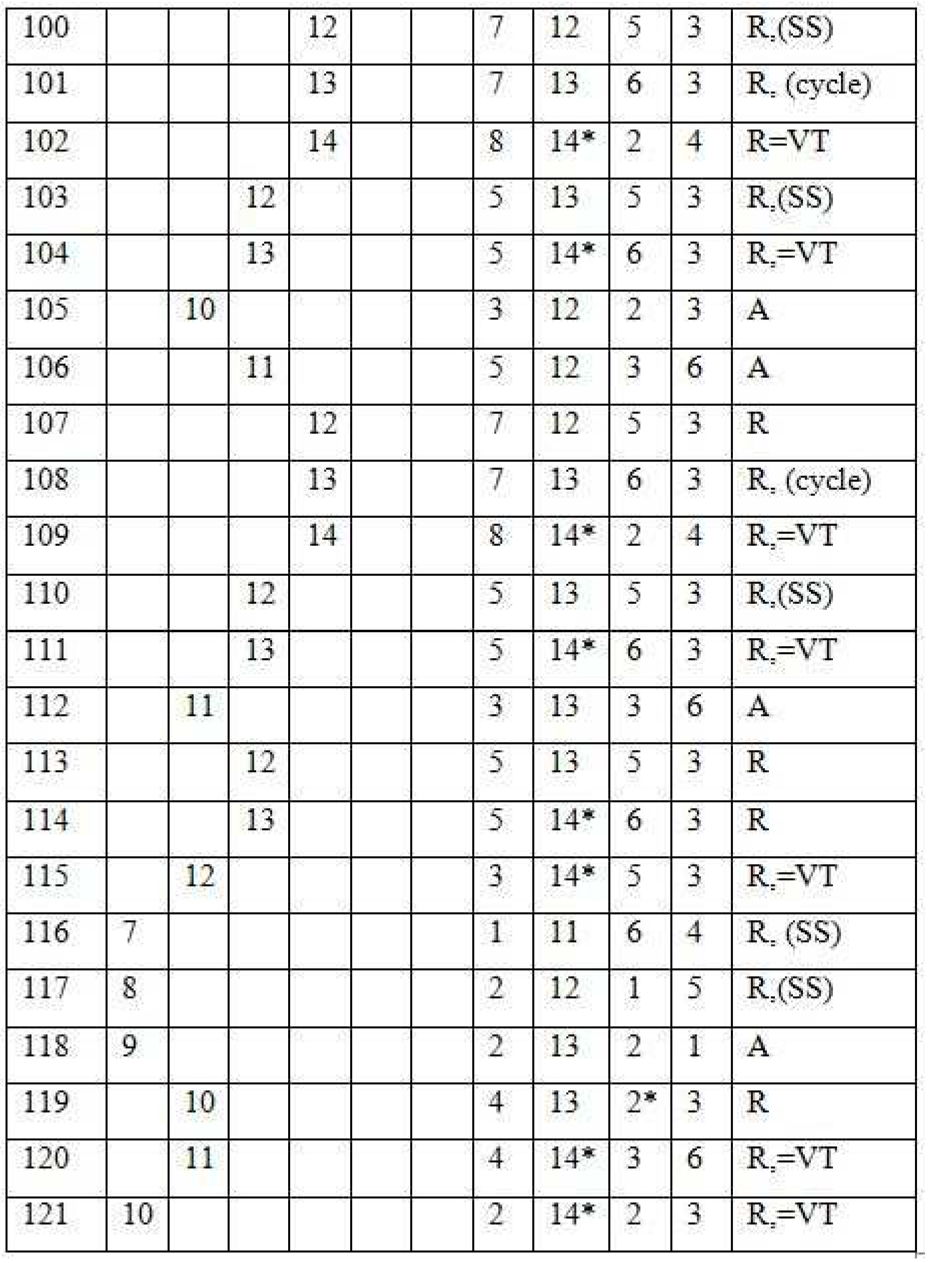


54			12				4	12	5	3	R,(SS)
55			13				4	13	6	3	A
56				14			7	13	2	4	R,(SS)
57				15			7	13	5	1	R,(SS)
58				16			7	14	5	2	R,(cycle)
59				17			7	15	6	5	R,=VT
60			14				5	14	2*	4	R
61			15				5	15*	5	1	R,=VT
62		**4**					**2**	**7**	**3**	**2**	**A**
63			5				3	7	4	6	R,(SS)
64			6				3	8	6	2	R,(SS)
65			7				3	9	6	4	R,(SS)
66			8				4	10	1*	5	R
67			**9**				**4**	**10**	**2**	**1**	**A**
68				10			6	10	2*	3	R
69				11			6	10	3*	6	R
70				**12**			**6**	**11**	**5**	**3**	**R,(SS)**
71				**13**			**6**	**12**	**6**	**3**	**A**
72					14		9	12	2*	4	R
73					15		9	12	5	1*	R,(SS)
74					16		9	12	5	2*	R
75					17		9	13	6*	5	R
76					18		10	14	1*	2	R



77					19		10	4	1*	6	R
78					20		10	14	4	3*	R
79					**21**		**10**	**14**	**4**	**5**	**A**
80						**22**	**14**	**14**	**5**	**6**	**A,VT=14**
81					22		10	14*	5	6	R,=VT
82				14			7	13	2	4	R,(SS)
83				15			7	13	5	1	R,(SS)
84				16			7	14*	5	2	R,=VT
85			10				4	10	2	3	R,(cycle)
86			11				4	11	3*	6	R
87			12				4	12	5	3	R
88			13				4	13	6	3	A
89				14			7	13	2	4	R
90				15			7	13	5	1	R
91				16			7	14*	5	2	R,=VT
92			14				5	14*	2	4	R
93		5					2	8	4	6	R,(SS)
94		6					2	9	6	2	R,(SS)
95		7					2	10	6	4	R
96		8					3	11	1	5	R
97		9					3	11	2	1	A
98			10				5	11	2*	3	R
99			11				5	12	3	6	A

ISSN: 2277-9655
Impact Factor: 1.852



100				12			7	12	5	3	R_2(SS)
101				13			7	13	6	3	R_2 (cycle)
102				14			8	14*	2	4	R=VT
103			12				5	13	5	3	R_2(SS)
104			13				5	14*	6	3	R_2=VT
105		10					3	12	2	3	A
106			11				5	12	3	6	A
107				12			7	12	5	3	R
108				13			7	13	6	3	R_2 (cycle)
109				14			8	14*	2	4	R_2=VT
110			12				5	13	5	3	R_2(SS)
111			13				5	14*	6	3	R_2=VT
112		11					3	13	3	6	A
113			12				5	13	5	3	R
114			13				5	14*	6	3	R
115		12					3	14*	5	3	R_2=VT
116	7						1	11	6	4	R_2 (SS)
117	8						2	12	1	5	R_2(SS)
118	9						2	13	2	1	A
119		10					4	13	2*	3	R
120		11					4	14*	3	6	R_2=VT
121	10						2	14*	2	3	R_2=VT

At the end of the search the current value of VT is 14 and it is the value of the optimal feasible word **L_6= {2, 4, 9, 13, 21, 22}** it is given in 80th row of the search table. The array IR, IC, SW and L takes the values represented in the **table-7**. The pattern represented by the above optimal feasible word is represented in the **table-8.**

Table-7

	1	**2**	**3**	**4**	**5**	**6**
IR	1	1	1	1	1	1
IC	1	1	1	1	1	1
SW	4	1	2	5	6	3
L	2	62	67	71	79	80

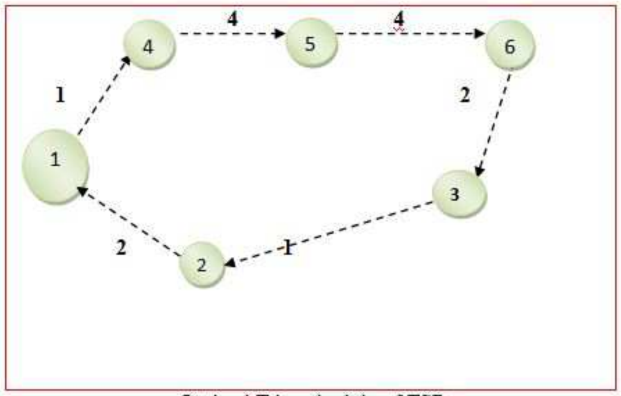
Table-8

X(i,j)=

0	0	0	1	0	0
1	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	1	0	0	0

The tour represented by the above pattern is **(1,4), (3,2),(2,1),(6,3),(4,5),(5,6).** In this tour the optimal solution for the considerable numerical example is 14.It also can be represented by **1→ 4→ 5→ 6 →3 →2 →1.** The diagramed representation of this arrangement can also see in the **figure-3**.

FIGURE-3



Optimal Trip schedule of TSP

Conclusion

In this paper, we have developed a Lexi-Search algorithm based on pattern recognition technique to solve. The model is then formulated as a zero one programming problem. A Lexi- Search Algorithm based on pattern recognition technique is developed for getting an optimal solution. A suitable numerical example is discussed for better understanding of the concepts and the steps involved in the algorithm.

Acknowledgements

One of the authors Prof. S.Sreenadh expresses thanks to UGC for providing financial support through the UGC SAP DRS-I Project to undertake this work. The authors would gratefully acknowledge the valuable advice from Prof. M. Sundara Murthy, Sri Venkateswara University, Tirupati.

References

[1] Balakrishna, U (2006), “Truncated Time – Dependent TSP” M.Phil dissertation, S.V. University, Tirupati , India.

[2] Flood, M.M, 1956 The TSP.Opns.Res.,41,pp. 61-75.

[3] Crores G.A., 1958“ A method for solving TSP” Opns. Res.Vol.6,,No.3, p 791.

[4] Naganna, B, 2007, Operations research, Ph.D, Thesis, S.V University, Tirupati,India.

[5] Pandit,S.N.N, 1963, Some Quantitative combinatorial Search Problems-Ph.D., Thesis,IIT Kharagpur.

[6] Sobhan Babu K, 2000, TDTSP&P- Median problems. M.Phil, Disertation, S.V.U.C.E. Tirupati, India

[7] Das. S, 1978, the most economical Route for a TSP” paper presented in ORSI.

[8] Jaillet, P. 1985“probabilisticTSP” Phd thesis Technical report no.185, Opns.Res., Center, Massachusetts Institute of Technology , Cambridgr, M.A.

[9] Sundara Murthy, M, 1979, Combinatorial programming-A Pattern Recognition Approach. PhD, Thesis REC, Warangal,India.

[10] Srivastava,S.S, Santosh Kumar,Garg,R.C& Sen P 1969,Generalized Travelling Sales man Problem through n sets of nodes. Opns.Res. 7, No2, pp 97- 101

[11] Bhavani, V, 1997 Combinatorial programming problems (TSP) M. Phil thesis, A.P., India.

[12] Hardgrave, W.W.&G.L. Nambhauser1962,“On the relation between the travelling salesman and the Longest Path Problems” Opns. Res.Vol10, p.647.

[13] Little,J.D.C. Murthy K.G. Sweeny, D.W, 1963 An Algorithm for the TSP. Opns. Res., 11, pp. 972-982.

[14] Glover.F, 1965, a multi phase- dual algorithm for the zero –one integer programming problems. Opns. 13. pp 879-919.