ISSN 2278-5612

P-Path Minimum Distance Connectivity from Head Quarter to the Cities

Revathi P[1], Suresh Babu C[2], Purusotham S[3], Sundara Murthy M[4]

[1]Research Scholar, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India.
revati.sai@gmail.com
[2]Academic Consultant, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India.
suresh8044@gmail.com
[3]Asst. Professor, Statistics and OR Division,VIT University, Vellore, Tamil Nadu, India
drpurusotham.or@gmail.com
[4]Professor, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India.
profmurthy@gmail.com

ABSTRACT

Many Combinatorial programming problems are *NP-hard* (Non Linear Polynomial), and we consider one of them called P-path minimum distance connectivity from head quarter to the cities. Let there be n cities and the distance matrix D(i, j, k) is given from i^{th} city to j^{th} city using k^{th} facility. There can be an individual factor which influences the distances/cost and that factor is represented as a facility k. We consider m<n cities are in cluster and to connect all the cities in subgroup (cluster) from others by using same facility k. The problem is to find minimum distance to connect all the cities from head quarter (say 1) threw p-paths subject to the above considerations. For this problem we developed a Pattern Recognition Technique based Lexi Search Algorithm, we programmed the proposed algorithm using C. we compared with the existed models and conclude that it suggested for solving the higher dimensional problems.

KEYWORDS

Lexi Search Algorithm, Pattern Recognition Technique, Partial word, Pattern, Mathematical formation.

Council for Innovative Research

Peer Review Research Publishing System

Journal: International Journal of Management & Information Technology

Vol.4, No.2
editor@cirworld.com
www.cirworld.com, member.cirworld.com

1. INTRODUCTION

In recent years the development of networks in the area of telecommunication and computer has gained much importance. One of the main goals in the design process is to reach total connectivity at minimum cost. The total connections are in some paths (say P). Similar problems arise in the planning of road maps, integrated circuits. The technical restriction that the number of connections at a node is bounded is modelled by introducing constraints that bound the node degrees. Garey al all [2] proved that the resulting degree-constrained minimum. In this paper we studied a variation of Minimum spanning models. For this we developed a Lexi- algorithm based on the “Pattern Recognition Technique” to solve this problem which takes care of simple combinatorial structure of the problem and computational results are reported.Some of the researchers studied variations in the Minimum Spanning Tree (MST) problems. They are Pop, P.C [6], Karger [3] found a linear time randomized algorithm based on a combination of Boruvka”s algorithm and the reverse-delete algorithm. The problem can be solved deterministically in linear by Chazelle [1] Its running time is o (m, α(m,n)) where function α grows extremely slowly. Thus Chazelle’s algorithm takes very close to linear time. Seth Pette [4,5] have found a probably optimal deterministic comparison-based minimum spanning tree algorithm. Sobhan Babu [8] studied a variation of spanning models using pattern recognition technique [9]. Let there be N cities to be connected to the Headquarter City {1}. There is an individual factor which influences the distances/cost and that factor is represented as a facility K. If the city j_1 and j_2 different cities are connected from city i_1 then k_1 and k_2 should be same. Suresh Babu [10] studied another variation of spanning models, which is reverse case of [11]. The problem is to find optimal solution for all the cities connected to head quarter {1} with minimum distance by using k facilities.

2. PROBLEM DESCRIPTION

Let there be n cities and the distance matrix D(i, j, k) is given from i^{th} city to j^{th} city using k^{th} facility. There can be an individual factor which influences the distances/cost and that factor is represented as a facility k. Let D(i, j, k) be the distance/cost from i^{th} city to j^{th} city with facility k. Let there be m<n cities as sub group from given number of n cities. We consider these m cities are in cluster. The restriction of the problem is to use the same facility for connecting all the cities in subgroup (cluster) from others. The problem is to find minimum distance to connect all the cities threw p-paths subject to the above consideration.

Let there are set of n cities in N = {1,2, . . ., n} and the set of k facilities in K = {1,2, . . ., k}. Then the three dimensional problem is: Let D (i,j,k) be the distance from i^{th} city to j^{th} city using k^{th} facility where i,j Є N and kЄ K. let there are set of m cities in M={1,2,3,...,m} such that M be the cluster and M is sub set of N. Let ‘{1}’ be head quarter city. We want to connection all the (n-1) cities from head quarter city by P-paths. Each city connected from head quarter city {1} either directly or indirectly. The objective of the problem is to find minimum total distance to connecting all the n-1 cities from head quarter {1} under the considerations. For this we developed an algorithm called as Lexi-Search algorithm based on the pattern recognition Technique and it is illustrated with a suitable numerical example for three paths.

3. MATHEMATICAL FORMULATION:

$$Minimize\ Z(X) = \sum_{I=1}^{n} \sum_{J=1}^{n} \sum_{k=1}^{n} D(i,j,k), X(i,j,k) \quad \text{------ (1)}$$

Subjected to the constraints

$$\sum_{k=1}^{k} \sum_{\mathrm{j}=2}^{\mathrm{j}} \mathrm{X}(1,\mathrm{j},\mathrm{k}) = \mathrm{p} \ldots \ldots \ldots \ldots \ldots \ldots (2)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{k} X(i,j,k) = n-1 \ldots \ldots \ldots \ldots .. (3)$$

Let α_{n1}, α_{n2}, α_{n3} ... α_{np} are np cities in p^{th} path

$$\text{Then} \quad \left.\begin{array}{l} \\ \sum_{s=1}^{n-1} x(\alpha_{rs,}\ \ \alpha_{rs+1)}) = n_{p-1} \quad \text{----------------} \\ \\ \end{array}\right\} (4)$$

Let α_1, α_2 Є M,

If X (α_1, β_1, γ_1) = X (α_2, β_2, γ_2), Then $\gamma_1 = \gamma_2$ ---------------- (5)

$$\sum_{i=1}^{p} n_i = n - 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . (6),$$

X (i, j, k) = 0 or 1 ------------------------- (7)

Equation (1) represents that the objective function of the problem. i.e., To find total minimum distance to connect all the cities from head quarter. The equations (2) represent that the total numbers of paths from head quarter to cities are p. Equation (3) represents that the total number of connections in the network are n-1. Equation (4) represent that the total number of connections in p^{th} path is equals to p. Equation (5) represent that the total cities in M using same facility. Equation (6) represent that the total number cities in all p paths are n-1. The constraint (7) describes that if a city *'i'* is connected to city j using facility k then *X*(i,j,k) = *1*. Otherwise it will be equal to 0.

4. NUMERICAL ILLUSTRATION:

The concept and algorithm developed will be illustrated by a numerical example for which total number of cities N={1,2,3,4,5,6,7,8,9}. Among them the cities 2, 5&6 are taken as separate cluster say M={2,5,6}. All the cities in M, when connect from cities should use same facility. Then the distance matrices D(i,j,k) are as follows.

Table -1

D(i,j,1)=

∞	1	4	-	16	3	-	1	-
∞	∞	5	17	14	-	21	-	7
∞	-	∞	4	5	4	-	8	9
∞	19	13	∞	-	18	-	20	-
∞	-	11	15	∞	-	10	-	2
∞	8	-	24	19	∞	3	21	13
∞	-	12	2	-	22	∞	-	6
∞	16	-	7	2	-	15	∞	-
∞	-	9	-	8	12	-	10	∞

Table-2

D(i,j,2)=

∞	1	3	22	-	3	25	-	13
∞	∞	24	-	19	2	-	14	2
∞	15	∞	16	-	4	7	25	-
∞	-	19	∞	17	-	21	-	14
∞	6	-	14	∞	12	8	11	3
∞	-	9	23	-	∞	2	-	18
∞	10	-	12	26	20	∞	8	10
∞	-	4	-	3	-	16	∞	7
∞	15	-	9	-	15	-	11	∞

In the above tables (Table 1&2) the value ∞ indicates the non-connectivity of the cities directly. Here the entire D(i, j, k)'s are taken as non-negative integers it can be easily seen that this is not a necessary condition and the distance cannot taken as negative quantities. In table 2.2 Suppose D(5,7,2)=8means the distance of the connecting the city 5 to 7 by using facility 2 is 8.For our convenience the total cities in cluster M can be identified by the array B as follows in Table-3

Table – 3

B(i)=	1	2	3	4	5	6	7	8	9
	0	1	0	0	1	1	0	0	0

In the above numerical example given in Table – 3, B (i) = 1 represent that the the city i is belongs to cluster M. Otherwise B(i) = 0. Suppose B(2)=1 means city 2 is in cluster M.

5. CONCEPT AND DEFINITIONS:

5.1. Definition of a pattern:

An indicator three-dimensional array which is associated with connection is called a ’pattern’. A Pattern is said to be feasible if X is a solution. $V(X) = \sum_{i\epsilon N}\sum_{j\epsilon N}\sum_{k\epsilon K} D(i,j,k)X(i,j,k)$ The value V(x) is gives the total cost of the tour for the solution represented by X. The value V(X) gives the total distance of the network for the solution represented by X. Thus X is the feasible pattern gives the total distance 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 triple (i, j, k) for which X (i, j, k)=1, with understanding that the other X(i, j, k)’s are zeros.

5.2. Alphabet table:

There are M=n×n×k ordered triples in the three-dimensional array X. For convenience these are arranged in ascending order of their corresponding cost and are indexed from 1 to M (Sundara Murthy-1979). Let SN= [1, 2, 3,…. M] be the set of M indices. Let D be the corresponding array of cost. If a, b$\in$SN and a < b then D(a)$\leq$ D (b). Also let the arrays R, C, K be the array of row, column and facility indices of the ordered triples represented by SN and CD be the array of cumulative sum of the elements of D. The arrays SN, D, CD, R, C, K for the numerical example are given in the table-4. If p$\in$SN then (R(p),C(p),K(p)) is the ordered triple and D(a)=T(R(a),C(a),K(a)) is the value of the ordered triple and CD (a) $= \sum_{i=1}^{a} D(i)$

Table – 4

SN	D	CD	R	C	K
1	1	1	1	2	1
2	1	2	1	8	1
3	1	3	1	2	2
4	2	5	5	9	1
5	2	7	7	4	1
6	2	9	8	5	1
7	2	11	2	6	2
8	2	13	2	9	2
9	2	15	6	7	2
10	3	18	1	6	1
11	3	21	6	7	1
12	3	24	1	3	2
13	3	27	1	6	2
14	3	30	5	9	2
15	3	33	8	5	2
16	4	37	1	3	1
17	4	41	3	4	1
18	4	45	3	6	1
19	4	49	3	6	2
20	4	53	8	3	2
21	5	58	2	3	1
22	5	63	3	5	1
23	6	69	7	9	1
24	6	75	5	2	2
25	7	82	2	9	1
26	7	89	8	4	1
27	7	96	3	7	2

28	*7*	*103*	*8*	*9*	*2*
29	8	111	3	8	1
30	8	119	6	2	1
31	8	127	9	5	1
32	8	135	5	7	2
33	8	143	7	8	2
34	9	152	3	9	1
35	9	161	9	3	1
36	9	170	6	3	2
37	9	179	9	4	2
38	10	189	5	7	1
39	10	199	9	8	1
40	10	209	7	2	2
41	10	219	7	9	2
42	11	230	5	3	1
43	11	241	5	8	2
44	11	252	9	8	2
45	12	264	7	3	1
46	12	276	9	6	1
47	12	288	5	6	2
48	12	300	7	4	2
49	13	313	4	3	1
50	13	326	6	9	1
51	13	339	1	9	2
52	14	353	2	5	1
53	14	367	2	8	2
54	14	381	4	9	2
55	14	395	5	4	2
56	15	410	5	4	1
57	15	425	8	7	1
58	15	440	3	2	2
59	15	455	9	2	2
60	15	470	9	6	2
61	16	486	1	5	1
62	16	502	8	2	1
63	16	518	3	4	2
64	16	534	8	7	2
65	17	551	2	4	1
66	17	568	4	5	2
67	18	586	4	6	1
68	18	604	6	9	2
69	19	623	4	2	1
70	19	642	6	5	1
71	19	661	2	5	2
72	19	680	4	3	2
73	20	700	4	8	1
74	20	720	7	6	2
75	21	741	2	7	1
76	21	762	6	8	1
77	21	783	4	7	2
78	22	805	7	6	1
79	22	827	1	4	2
80	23	850	6	4	2
81	24	874	6	4	1
82	24	898	2	3	2
83	25	923	1	7	2
84	25	948	3	8	2
85	26	974	7	5	2

From the above table – 4, Let us consider 75 $\in$ SN. It represents the ordered pair D (R(75),C(75), K(75))=(2,7, 1). Then D (75) = D (2, 7, 1) = 21 and CD (75) = 741.

5.3. Definition of alphabet table and a word:

Let SN= (1,2,...) be the set of indices, let D be an array of distances in between respective cities. CD be an array of cumulative sums of elements in D. Let arrays R,C and K be respectively, the row, column and facility indices of the ordered triples. Let L $_k$ ={a_1, a_2,......,a_k}, a_i €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_i ≤a_{i+1}, i=1,2,........k-1. The set SN is defined as the "Alphabet-Table" with alphabet order as (1,2,....,n^2p) 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 has at least one feasible word or, equivalently the partial pattern represented by L_k has at least one feasible word or equivalently the partial pattern represented by 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 then 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=n, 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 partial word 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

5.4. 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_i) with V(L_0)=0, where D(a_i) is the distance array arranged such that D(a_i)≤D(ai_{+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).

Consider the partial word L_4 = (1,2,4,5) Then V(L_4) =1+1+2+2=6

5.5. 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=(a_i,a_2, ...a_k) can be defined as follows.

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

Consider the partial word L_4 = (1, 2, 4, 5) and V (L_4) =1+1+2+2=6 Then

LB(L_k)=V(L_k)+CD(a_4+n-1-k)-CD(a_k)=6+CD(5+8-4)-CD(5)=6+CD(9)-CD(5) =6+15-7 = 14

5.6. Feasibility criterion of partial word

An algorithm was developed, in order to check the feasibility of a partial word L_{k+1} = {a_1,a_1,............a_k,a_{k+1}} given that L_k is a feasible word. We will introduce some more notations which will be useful in the sequel.

- IR be an array where IR(i)=1, i€N indicates that the i^{th} city is connected to some city j otherwise IR(i)=0
- **IC** be an array where IC (i) = 1, i∈N indicates that the i^{th} city is connected from some city j otherwise IC(i)=0
- **SW** be an array where SW (i) = j indicates that the i^{th} city is connected to city j otherwise SW (i) = 0
- **L** be an array where L[i] = α $_i$, i∈N is the letter in the i^{th} position of a word

  Then the values of the arrays IR, IC, SW, L are as follows

  - IR (R (a_i)) = 1, i = 1, 2, - - - - - , k and IR (j) = 0 for other elements of j
  - IC (C (a_i)) = 1, i = 1, 2, - - - - - , k and IC (j) = 0 for other elements of j
  - 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, and L(j) = 0, for other elements of j
  - For example consider a sensible partial word L_4 = (1, 2, 4, 5) which is feasible. The array IR, IC, L, SW takes the values represented in table - 5 given below.

Table - 5

	1	2	3	4	5	6	7	8	9

L	1	2	4	5					
IR	2				1		1		
IC		1		1				1	1
SW	2,8				9		4		

The recursive algorithm for checking the feasibility of a partial word L_p is given as follows In the algorithm first we equate IX = 0. At the end if IX = 1 then the partial word is feasible, otherwise it is infeasible. For this algorithm we have TR=R(a_i), TC=C(a_i),TK=K(a_i)

6. ALGORITHMS:

6.1. Algorithm 1: (Algorithm For Feasible Checking)

```
STEP0      :   IX=0              GO TO 1
STEP1      :   IS TR=HC          IF YES GO TO 2
                                 IF NO GO TO 3

STEP2      :   IS IR (HC) <P           IF YES GO TO 3
                                       IF NO GO TO 15
STEP3      :   IS IR (TR)=1            IF YES GO TO15
                                 IF NO GO TO 4
STEP4      :   IS IC (TC)=1            IF YES GO TO 15
                                       IF NO GO TO 5
STEP4A     :   Z1=P-NP, RP=N-1-I
               IS (RP>=Z1)             IF YES GOTO 5
                                       IF NO GOTO 15
STEP5      :   W = TC                  GOTO 6
STEP6      :   IS SW (W) = 0            IF YES GO TO 9
                                 IF NOGO TO 7
STEP7      :   IS W=TR                 IF YES GO TO16
                                 IF NO GO TO 8
STEP8      :   W=SW (W)            GO TO 6
 STEP9     :   IS B (TC) =1        IF YES GOTO10
                                 IF NO GOTO 14
STEP10     :   M=B (TC)
               IS NB = 0                IF YES GOTO 11
                                 IF NO GOTO 12
STEP11 :    F(M)=TK                    GOTO13
STEP12 :    IS F(M) =TK                 IF YES GOTO13
                                 IF NO GOTO   16
STEP13 :    NB=NB+1                    GOTO14
STEP14 :     IX=1
STEP15 :     STOP
```

$L_k = L_{k-1}$ * Where * indicates chain formulation. We will calculate the values of V(Lp) and LB (Lp) simultaneously. Then two situations 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 processed to consider a partial word of under (P+1). Which represents a sub-block of the block of Words represented by LP
2. LB (Lp) ≥VT . In this case we reject the partial word LP. We reject the block of word with L_p as leader as not having optimum feasible solution and also reject all partial words of order p that succeeds L_p.

6.2. Algorithm 2: (Lexi-Search Algorithm)

```
STEP-1:  (initialization):
         The arrays SN, D, CD, R, C, K, P, M, N and B are made available IR, IC, F, L, V, LB and SW are initialized to
zero.  The values I=1, J=0, VT =999, HC=1, NZ=0, MAX=n*n*m
STEP-2  :      J=J+1
               IS (J≥MAX)                          IF YES GOTO 13
                                                   IF NO GO TO 3

STEP-3   :     L( I)=J, TR=R(J), TC=C(J), TK=K(J)
               V(I)=V(I-1)+D(J),  LB(I)=V(I)+DC(J+N-1-I)-DC(J)
               IS (LB(I)≥VT)                                      IF YES GOTO 15
                                                                  IF NO GOTO 4
STEP-4  :      (CHECK THE FEASIBILLITY BY USING ALGORITHM-1)
               IS (IX=0)                                          IF YES GO TO 2
                                                                  IF NO GO TO 5
STEP 5  :      IS (I = N-1)                                       IF YES GO TO 6
                                                                  IF NO GO TO 11
STEP 6  :      IS HC=TR                                           IF YES GOTO 7
                                                                  IF NO GOTO 8
STEP 7  :      IS IR(TR)=P-1                                      IF YES GOTO 9
                                                                  IF NO GOTO 2
STEP 8  :      IS IR(TR)=P                                        IF YES GOTO 9
                                                                  IF NO GOTO 2
STEP 9  :      VT = V (I), L (I) = J, L (I) is full length word and is feasible
               Record L (I) , VT
               IS B(TC)=1                                 IF YES NB=NB-1 GOTO 13
                                                          IF NO GOTO 13
STEP 11        :        L (I) = J
               IC (TC) = 1, SW (TR) = TC
               IS TR = HC                                 IF YES IR(TR)=IR(TR)+1GOTO 12
                                                          IF NO IR(TR)=1GOTO 12
STEP 12        :        I = I+1                                    GOTO 2
STEP-13        :        I=I-1, J=L(I), TR=R(J), TC=C(J), IC(TC)=0, SW(TR)=0
               IS TR = HC                                 IF YES IR(TR)=IR(TR)-1GOTO 14
                                                          IF NO IR(TR)=0 GOTO 14
STEP14 :       IS B(TC)=1                                 IF YES NB=NB-1GOTO 2
                                                          IF NO GO TO 2
STEP 15        :        IS (I = 1)                                IF YES GO TO 16
                                                          IF NO GO TO 13
STEP-16        :        STOP
```

6.3. Search table

The working detail of getting an optimal word using the above algorithm for the illustrative numerical example is given in the following table-6. The columns named (1), (2),(3)............gives the letters in the first second thirds and so on places respectively. The columns R,C gives the row, column and facility indicates the letter. The last columns give the remarks regarding the acceptability of the partial word. In the following table 'A' indicates ACCEPT and 'R' indicates REJECT.

Table-6

SN	1	2	3	4	5	6	7	8	V	LB	R	C	K	REMARK
1	1								1	13	1	2	1	A
2		2							2	13	1	8	1	A
3			3						3	13	1	2	2	R
4			4						4	14	5	9	1	A
5				5					6	14	7	4	1	A
6					6				8	14	8	5	1	A
7						7			10	14	2	6	2	R
8						8			10	15	2	9	2	R
9						9			10	16	6	7	2	A
10							10		13	16	1	6	1	A
11								11	16	16	6	7	1	R
12								12	16	16	1	3	2	R
13								13	16	16	1	6	2	R
14								14	16	16	5	9	2	R
15								15	16	16	8	5	2	R
16								16	17	17	1	3	1	R
17								17	17	17	3	4	1	R
18								18	17	17	3	6	1	R
19								19	17	17	3	6	2	R
20								20	17	17	8	3	2	R
21								21	18	18VT	2	3	1	A
22							11		13	16	6	7	1	R
23							12		13	16	1	3	2	A
24								13	16	16	1	6	2	R
25								14	16	16	5	9	2	R
26								15	16	16	8	5	2	R
27								16	17	17	1	3	1	R
28								17	17	17	3	4	1	R
29								18	17	17=VT	3	6	1	A
30							13		13	16	1	6	2	R
31							14		13	16	5	9	2	R
32							15		13	17	8	5	2	R=VT
33						10			11	17	1	6	1	R=VT
34					7				08	15	2	6	2	R
35					8				08	16	2	9	2	R
36					9				08	17	6	7	2	R=VT
37				6					06	15	8	5	1	A
38					7				08	15	2	6	2	R
39					8				08	16	2	9	2	R
40					9				08	17	6	7	2	R=VT
41				7					06	16	2	6	2	R
42				8					06	17	2	9	2	R=VT
43			5						04	15	7	4	1	A
44				6					06	15	8	5	1	A
45					7				08	15	2	6	2	R
46					8				08	16	2	9	2	A
47						9			10	16	6	7	2	A

48							10		13	16	1	6	1	A
49								11	16	16	6	7	1	R
50								12	16	16	1	3	2	R
51								13	16	16	1	6	2	R
52								14	16	16	5	9	2	R
53								15	16	16	8	5	2	R
54								16	17	17	1	3	1	R=VT
55							11		13	16	6	7	1	R
56							12		13	16	1	3	2	A
57								13	16	16	1	6	2	R
58								14	16	16	5	9	2	R
59								15	16	16	8	5	2	R
60								16	17	17	1	3	1	R=VT
61							13		13	16	1	6	2	R
62							14		13	16	5	9	2	R
63							15		13	17	8	5	2	R=VT
64						10			11	17	1	6	1	R=VT
65					9				08	17	6	7	2	R=VT
66				7					06	16	2	6	2	R
67				8					06	17	2	9	2	R=VT
68			6						04	16	8	5	1	A
69				7					06	16	2	6	2	R
70				8					06	17	2	9	2	R,=VT
71			7						04	17	2	6	2	R=VT
72		3							02	14	1	2	2	R
73		4							03	16	5	9	1	A
74			5						05	16	7	4	1	A
75				6					07	16	8	5	1	A
76					7				09	16	2	6	2	R
77					8				09	17	2	9	2	R=VT
78				7					07	17	2	6	2	R=VT
79			6						05	17	8	5	1	R=VT
80		5							03	17	7	4	1	R=VT
81	2								01	14	1	8	1	A
82		3							02	14	1	2	2	A
83			4						04	14	5	9	1	A
84				5					06	14	7	4	1	A
85					6				08	14	8	5	1	R
86					7				08	15	2	6	2	A
87						8			10	15	2	9	2	R
88						9			10	16	6	7	2	A
89							10		13	16	1	6	1	R
90							11		13	16	6	7	1	R
91							12		13	16	1	3	2	A
92								13	16	16	1	6	2	R
93								14	16	16	5	9	2	R
94								15	16	16VT	8	5	2	A
95							13		13	16	1	6	2	R=VT
96						10			11	17	1	6	1	R>VT
97					8				08	16	2	9	2	R=VT

98				6					06	15	8	5	1	R
99				7					06	16	2	6	2	R=VT
100			5						04	15	7	4	1	A
101				6					06	15	8	5	1	R
102				7					06	16	2	6	2	R=VT
103			6						04	16	8	5	1	R=VT
104		4							03	16	5	9	1	R=VT
105	3								01	16	1	2	2	R=VT

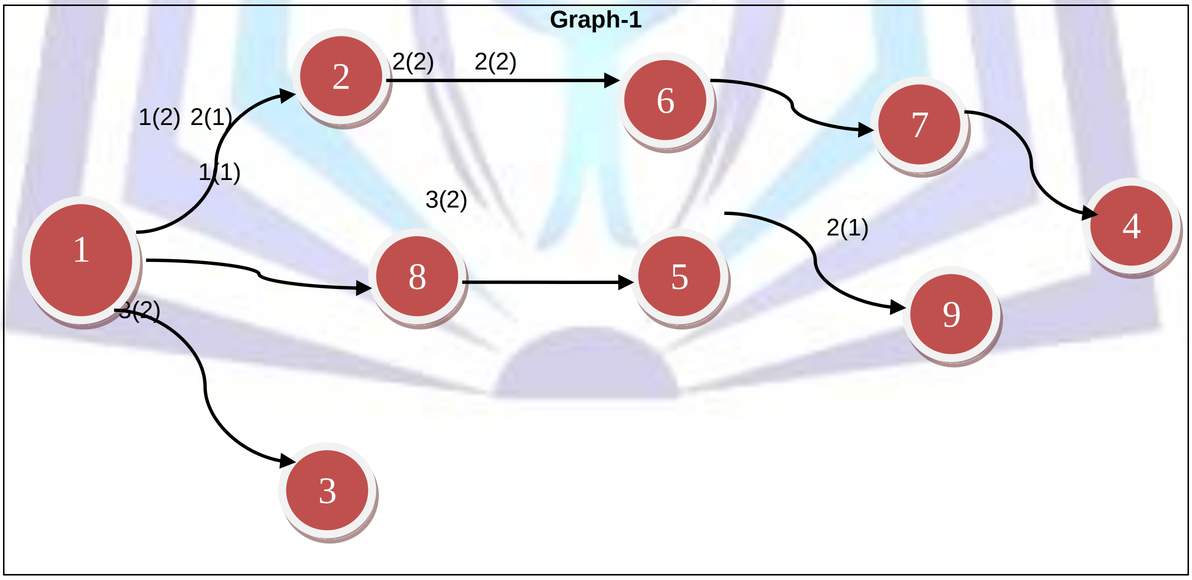
The above table – 6, gives optimal solution of the taken numerical example and the partial word is L_8 = (2, 3, 4, 5, 7, 9, 12 and 15) is a feasible partial word. The end of search table optimal solution VT is 16. It is in the 94th row of the search table. For this partial word the array IR, IC, SW, L are given an the following Table – 7

Table – 7

	1	2	3	4	5	6	7	8	9
L	2	3	4	5	7	9	12	15	-
IC	1	1	1	1	-	1	1	1	1
SW	2,3,8	6	-	-	9	7	4	5	-

At the end of the search the current value of VT is **16** and is the value of optimal feasible word L_8 = (2, 3, 4, 5, 7, 9, 12 and 15). Then the following graph – 1 represents the optimal solution to the Network.

Graph-1



from the above graph-1, the city 1 connected to 2 at fecility 1, city 8 at facility 1 and connected to 3 by facility 2. In similar way all the cities are connected from headquarter 1 directly or indirectly. More over all the cities in M are connected by same (2) facility. Hence the solution of the above graph as follow

Z=D(1,8,1)+D(1,2,2)+D(5,9,1)+D(7,4,1)+D(2,6,2)+D(6,7,2)+D(1,3,2)+D(8,5,2)
=1+1+2+2+2+2+3+3=16

7. EXPERIMENTAL RESULTS:

The following table shows that the computational results for proposed algorithm called pattern recognition technique based lexi-search algorithm. We write Computer program for this algorithm in C language and it is verified by the system COMPAQ dx2280 MT. We ensure this algorithm by trying a set of problems for different sizes. We take the different random numbers as values in distance matrix, business matrix along with route business. The distance Matrix D (i, j, k) takes the values uniformly random in [0,1000]. We tried a set of problems by giving different values to M, N and CL. The results are tabulated in the Table -8 are given below. For each instance, five to eight data sets are tested. It is seen that the time required for the search of the optimal solution is fairly less.

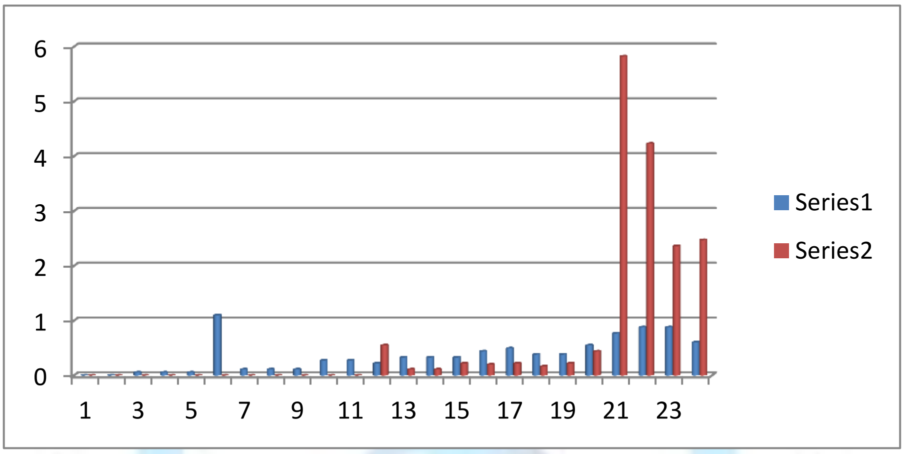
Table-8

SN	Problem dimension			NPT	CPU runtime in seconds	
	N	M	CL		Avg AT	Avg ST
1	4	2	1	6	0	0
2	4	3	1	6	0	0
3	8	2	2	6	0.054945	0
4	8	3	1	6	0.054945	0
5	8	3	2	6	0.054945	0
6	8	4	2	6	1.09890	0
5	12	2	1	8	0.109890	0
6	12	2	2	8	0.109890	0
7	12	3	2	8	0.109890	0
8	12	4	2	8	0.274725	0
9	12	4	3	8	0.274725	0
10	16	2	1	8	0.219780	0.54945
11	16	3	1	8	0.329670	0.109890
12	16	3	2	8	0.329670	0.109890
13	16	3	3	8	0.329670	0.219780
14	16	4	2	8	0.439560	0.19890
15	16	4	3	8	0.494505	0.219780
16	20	2	1	8	0.384615	0.164835
17	20	2	2	8	0.384615	0.219780
18	20	3	1	8	0.549451	0.439560
19	20	4	1	8	0.769231	5.824176
20	24	2	2	8	0.879121	4.230769
21	24	2	3	8	0.879121	2.362637
22	28	1	1	8	0.604396	2.472527

In the above table-8, SN = serial number, N = number of cities, M = number of facilities, CL = number of clusters to be taken, NPT = number of problems tried. In the next columns Avg. AT = average CPU run time to form an alphabet table.

Avg. ST = average CPU run time to form search table for obtaining the optimal solution. It is seen that time required for the search of the optimal solution is moderately less.

Graph – 2



The graphical representation of the above instances is given below. In the following Graph – 2, X axes taken the SN and Y axes taken the values of CPU run time for forming alphabet table and getting optimal solution. The bars of different colours indicate CPU run time for formation of alphabet table and search time for getting optimal solution respectively. i.e., series1 represent that CPU run time for formation of alphabet table and series 2 represent that CPU run time for searching the optimal solution.

8. COMPARISON DETAILS:

We implement the Pattern Recognition Technique based Lexi Search Algorithm (LSA) with C language for this model. We tested the proposed algorithm by different set of problems and compared the computational results with the published minimum spanning tree model by C. Suresh Babu Volume 3, No. 1, Jan-Feb 2012 International Journal of Advanced Research in Computer Science. Then Table-9 shows that the comparative results of different sizes.

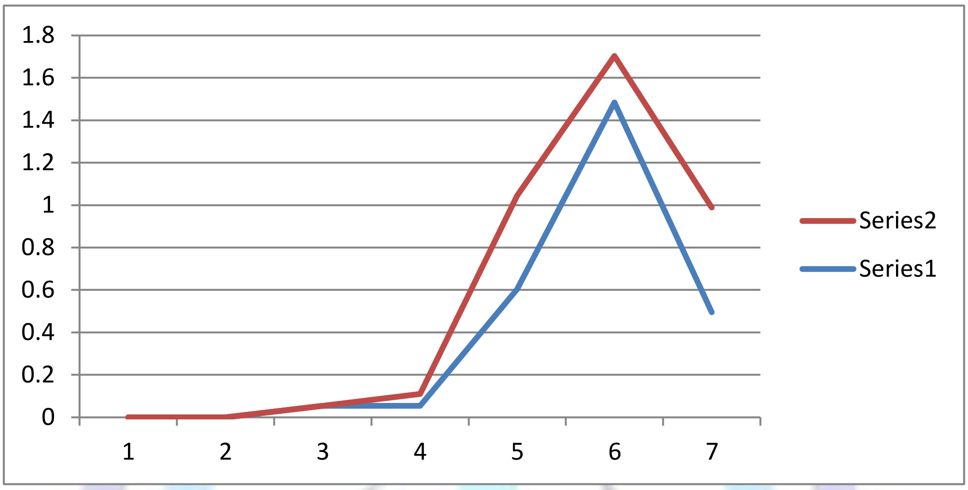
Table-9

SN	N	M	CL	NPT	VT	CPU Run Time in seconds	
						Published model	Proposed model
1	4	2	1	5	399	0.000	0.000
2	8	3	1	5	399	0.000	0.000
3	12	3	3	5	399	0.05490	0.000
4	16	2	1	5	399	0.054945	0.054945
5	18	3	2	5	399	0.604396	0.43956
6	20	2	2	5	399	1.483516	0.219780
7	22	2	2	5	399	0.49451	0.49450

In the above table 9, The last two columns show the CPU run time of published model and proposed model. As compared the two models of size N=20. The runtime of this instance with the existing model is 1.483516 sec., and the proposed model took 0.219780 Sec., it is reasonably less time. The present model takes very less computational time for finding the optimal solution. Hence, suggested the present model for solving the higher dimensional problems also.

The graphical representation of the CPU run time for the two models presented in the above 7 instances is given below. In the Graph-3, X axes taken the SN and Y axes taken the values of CPU run time for the published and proposed models.

Graph-3



From the above Graph-3, series2 represent that CPU run time for getting optimal solution by published model and series 1 represent that CPU run time for searching the optimal solution by the proposed model. Also the proposed model takes less time than published model for giving the solution.

9. CONCLUSION

In this paper, we have presented an exact algorithm called lexi-search algorithm based on pattern recognition technique to solve the Minimum Spanning Tree Problem. First the model is formulated into a zero-one programming problem. A Lexi-Search Algorithm based on Pattern Recognition Technique is developed for getting an optimal solution. The problem is discussed with suitable numerical illustration. We have programmed the proposed algorithm using C-language. The computational details are reported. As an observation the CPU run time is fairly less for higher values to the parameters of the problem to obtain an optimal solution.

10. REFERENCES

[1]. A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity, B. Chazelle, Journal ACM 47 (2000), 1028-1047. Prelim. version in FOCS 1997.

[2]. Garey, Michael R.; Johnson, David S. (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, ISBN 0-7167-1045-5.

[3]. Karger, David R.; Klein, Philip N.; Tarjan, Robert E. (1995), "A randomized linear-time algorithm to find minimum spanning trees", *Journal of the Association for Computing Machinery* **42** (2): 321–328, doi:10.1145/201019.201022, MR 1409738

[4]. Pettie, Seth; Ramachandran, Vijaya (2002), "A randomized time-work optimal parallel algorithm for finding a minimum spanning forest", *SIAM Journal on Computing* **31** (6): 1879–1895, doi:10.1137/S0097539700371065, MR 1954882.

[5]. Pettie, Seth; Ramachandran, Vijaya (2002), "Minimizing randomness in minimum spanning tree, parallel connectivity, and set maxima algorithms", *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, San Francisco, California, pp. 713–722.

[6]. Pop, P.C.,"New models of the Generalized Minimum Spanning Tree Problem", Journal of Mathematical Modelling and Algorithms, Volume 3, issue 2, 2004, 153-166.

[7]. Reeves, C.R., “Moderen Metaheuristics Techniques for Combinatorial Problems”, Blackwell, Oxford, 1993.

[8]. Sobhan Babu, K., Chandra Kala, K., Purusotham, S. and Sundara Murthy, M. “A New Approach for Variant Multi Assignment Problem”, International Journal on Computer Science and Engineering, Vol.02,No.5, 2010, 1633-1640.

[9]. Sundara Murthy, M. “Combinatorial Programming: A Pattern Recognition Approach,” A Ph.D. Thesis, REC, Warangal. 1979.

[10]. Suresh Babu C, Sobhan Babu K and Sundara Murthy M, 2011, published a paper entitled “Variant Minimum Spanning Network Connectivity Problem” by the International Journal of Engineering Science and Technology for publication. Volume 3, No. 1, Jan-Feb 2012.