International Journal of Mathematical Archive-4(8), 2013, 6-19
Available online through www.ijma.info ISSN 2229 – 5046

THREE DIMENTIONAL P - TRUCKS ROUTE MINIMUM COST SUPPLY TO THE CITIES FROM THE HEAD QUARTER CITY

Madhu Mohan Reddy P[1*], Suresh Babu C[2], Purusotham S[3] and Sundara Murthy M[4]

[1, 2, 4]Department of Mathematics, Sri Venkateswara University, Tirupati, Andhra Pradesh, India

[3]Statistics and Operations Research Division, Department of mathematics,
Vellore Institute of Technology University, Vellore, Tamil Nadu, India

(Received on: 30-06-13; Revised & Accepted on: 25-07-13)

ABSTRACT

Many Combinatorial programming problems are NP-hard (Non Linear Polynomial), and we consider one of them called P-path minimum cost connectivity from head quarter to the cities. Let there be n cities and the cost matrix D(i, j, k) is given from ith city to jth city using kth 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 head quarter city has the capacity to supply the load (capacity) different cities according to their requirements. The problem is to find minimum cost to connect all the cities from head quarter (say 1) through 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, load.*

---

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 modeled by introducing constraints that bound the node degrees. Garey at 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/cost by using k facilities.

Corresponding author: Madhu Mohan Reddy P[1*]
[1, 2, 4]Department of Mathematics, Sri Venkateswara University, Tirupati, Andhra Pradesh, India
E-mail: mmrphdsv@gmail.com

2. PROBLEM DESCRIPTION

In this chapter we study the problem called “Three Dimensional P- Trucks route minimum Cost Supply to the Cities from the Head Quarter City “. The objective is to find the minimum total cost of the path and required capacities are supplied to each city from head quarter city “1” through the P-trucks (say=3). There is a restriction that all n-1 cities connected through the P –paths from head quarter city and also each path requirement is not greater than 100 units of capacity. The total capacity in head quarter city is 300 units. Let there be N= {1, 2, 3, 4……n} cities whose costs NXN are given. In this chapter we consider Pattern Recognition Based Lexi-Search Approach for **minimum spanning network connectivity problem (msncp).** There is a restriction that N_1 cities must have same facility. An exact algorithm is proposed for this msncp. The algorithm solves the problem on identify the key patterns which optimize the objective of the cost/distance.

Let N be the set of n stations defined as N={1,2,3,4,……..n} and the set of k facilities in K = {1,2, . . ., q}. Let D (i, j, k) be the cost from i^th^ city to j^th^ city using k^th^ facility where i, j Є N and k Є 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 connect 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 P- trucks has starts from head quarter city {1}, from P-paths. The total units in P- trucks are greater than or equal to the requirement of the cities. 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

$$Minimize\ Z(X) = \sum_{I=1}^{n}\sum_{J=1}^{n}\sum_{K=1}^{q} D(i,j,k), X(i,j,k) \tag{1}$$

Subject to the constraints

$$\sum_{s=1}^{p} x(1, n_{s,1}) \tag{2}$$

$$\sum_{s=1}^{p} n_s = n-1 \tag{3}$$

$$\sum_{s=1}^{n_s-1} x(\alpha_{s,j}\ ,\ \alpha_{s,j+1}) = n_s - 1 \tag{4}$$

$$\sum_{j=1}^{n_s} QR(\alpha_{s,j}) \le VQ \tag{5}$$

$$\sum_{j=1}^{P} QR(j) \le P.VQ \tag{6}$$

If X $(\alpha_1, \beta_1, \gamma_1)$ = X $(\alpha_2, \beta_2, \gamma_2)$ =1 and β_1, β_2 ЄM Then $\gamma_1 = \gamma_2$ (7)

X (i, j, k) = 0 or 1 (8)

Equation (1) represents that the objective function of the problem. i.e., to find total minimum cost connecting head quarter to all the cities. The equation (2) represent that the total number of paths from cities to head quarter is p.. Equation (3) represent that the total number cities in all p paths are n-1. Equation (4) represent in s^{th} path there are connected n_s-1 arcs connecting n_s cities. Equation (5) represent the requirement in each path is less than or equal to the vehicle capacity VQ. Equation (6) represent that the total requirement of the cities is less than or equal to the total capacity of the P paths. Equation (7) represent that the total cities in M the same facilities. Equation (8) describes that if a city *‘i’* is connected to city j using facility k then *(i, j, k) = 1*. Otherwise it will be equal to 0.

4. NUMERICAL ILLUSTRATION

The concepts and algorithm developed will be illustrated by a numerical example for which we have to take total number of cities N={1,2,3,4,5,6,7,8,9} among them 1 as a head quarter city and take cities 2,4,9 as one cluster i.e, M={2,4,9}. In the following numerical example, D(i, j, k)'s are taken non- negative integers it can be easily seen that this is not a necessary condition .In table 1, D(6,3,1)=6 means that the cost of the connecting the city 6 to 3 by using facility 1 is 6.'-'indicates the there is no connection between the corresponding cities. The following table represents the requirement of the cities. All the cities are connected the head quarter city either directly or indirectly through P-paths. Then the cost array is given below.

TABLE-1A

D(i,j,1)=

-	1	∞	1	7	∞	∞	24	25
∞	-	29	∞	30	∞	38	∞	37
∞	42	-	2	33	∞	37	49	50
∞	3	40	-	∞	23	∞	54	1
∞	∞	∞	6	-	52	36	∞	38
∞	40	6	4	34	-	∞	26	∞
∞	∞	9	∞	30	26	-	32	27
∞	∞	25	4	38	∞	33	-	32
∞	18	32	∞	30	34	19	42	-

TABLE-1B

D(i,j,2) =

-	∞	2	42	∞	39	∞	∞	∞
∞	-	30	10	1	32	16	4	27
∞	21	-	38	15	6	4	4	19
∞	∞	49	-	∞	18	∞	25	∞
∞	28	∞	∞	-	22	1	24	∞
∞	28	26	26	29	-	∞	34	22
∞	22	55	∞	40	36	-	24	∞
∞	23	∞	20	43	4	28	-	29
∞	44	8	36	∞	46	∞	3	-

For our convenience we consider all the cities are taken in two dimensional arrays B. Then the array B is given below

TABLE-2

B(i) =

1	2	3	4	5	6	7	8	9
0	1	0	1	0	0	0	0	1

In the above numerical example given in Table – 2, B (i) = 1 represent that the city i is belongs to cluster M. Otherwise B(i) = 0. Suppose B (2)=1 means city 2 is in cluster M. consider the requirement of the cities taken in the following matrix Q as table-3

TABLE-3

QR(j) =

1	2	3	4	5	6	7	8	9
-	30	30	20	30	40	40	30	40

From the above table-3, QR (j) = α means that the requirement of city j is α. suppose Q(2) = 30 means that the requirement of city 2 is 30. Here QR(1)= '-' means that the city 1 act as head quarter so no need to requirement at 1 but it has the availability 300.

4. CONCEPTS AND DEFINITIONS

4.1. Definition of a pattern:

An indicator three-dimensional array which is associated with an assignment is called a 'pattern'. A Pattern is said to be feasible if X is a solution. $V(x) = \Sigma\Sigma\ \Sigma\ D(i, j, k) . X(i, j, k)$, here $i \varepsilon N$, $j \varepsilon N$, $k \varepsilon K$. The pattern represented in the table-4 is a feasible pattern. The value V(X) gives the total cost of the connectivity of the cities 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 triples [(i, j, k)] for which X(i, j, k)=1, with understanding that the other X(i, j, k)'s are zeros.

4.2 Feasible solution:

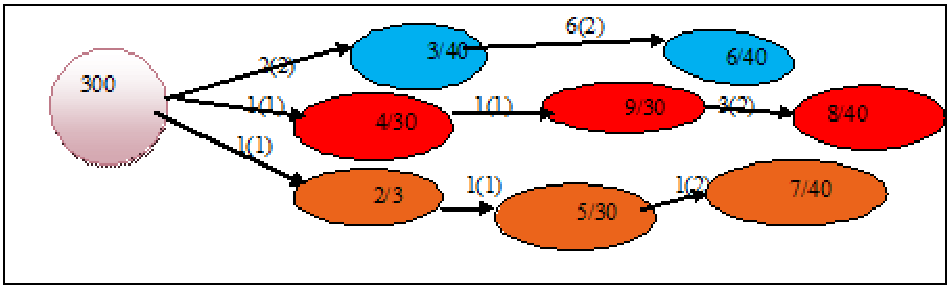
Consider an ordered triple set{(1,2,1),(1,4,1),(4,9,1),(1,3,2),(2,5,2),(3,6,2),(5,7,2),(8,8,2),(9,8,2)}Represents the pattern given in the table-4, which is a feasible solution.

TABLE- 4

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

The above solution represented in the following Figure-1, in the following Figures ellipses are represented as cities and Circle is taken as head quarter city. The values in Ellipses indicated as fractions. In a fraction numerator represents the particular city and denominator represents the requirement of respective city. The values on the arrow mark are the cost between the two cities and the value in the parenthesis represent facility of the respective cities which are connected. The following **figure** having the 3 paths. First path having the cities 1, $\alpha_{11}, \alpha_{12} =$ 1, 3, 6. Same way Second path having the cities 1, $\alpha_{21}, \alpha_{22}, \alpha_{23} =$1, 4, 9, 8 and third path having the cities 1, $\alpha_{31}, \alpha_{32}, \alpha_{33} =$ 1, 2, 5, 7

Figure-1



From the above figure-1, three trucks are started from head quarter city 1 with 100 units of capacity. In first path the city 1 is connected to city 3 at facility 2 with 40 units, city 3 is connected to city 6 at facility 2, with 40 units. In second path city 1is connected to city 4 at facility 1 with 30 units, city 4 is connected to city 9 with facility1 with 30 units; city 9 is connected to city 8 at facility 2 with 40 units. In third path city 1is connected to city 2 at facility 1 with 30 units, city 2 is connected to city 5 at facility2 with 30 units, city 5 connected to city 7 at facility 2 with 40 units. So that all the cities in cluster M (2, 4,9) using the same facility 1 to connect from other cities. Hence the solution is

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

$=1+1+1+1+1+3+2+6= 16$

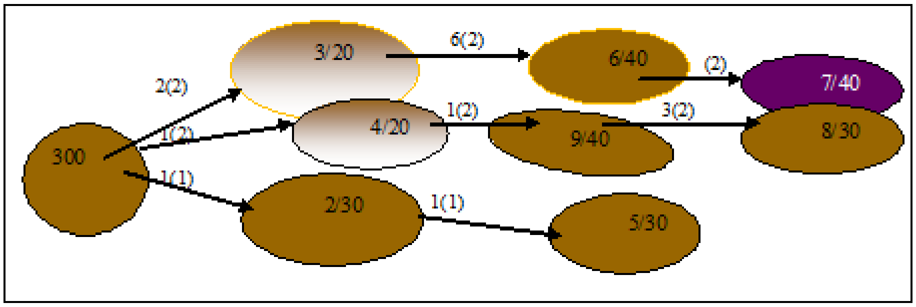
4.3 Infeasible solution:

Consider a ordered triple set {(1,2,1),(2,5,1),(1,3,2),(1,4,2),(3,6,2),(4,9,2), (6,7,2),(9,8,2)} represents the pattern given in the following table-5, which is an infeasible solution

$$D(I,j,1)\begin{bmatrix} 0&1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \qquad D(i,j,2)=\begin{bmatrix} 0&1&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0 \end{bmatrix}$$

The pattern of above infeasible solution represented in the following Figure-2

Figure-2



In the above figure2, the three trucks are started from head quarter city 1 with 100 units of capacity and city 1 connected to 2 at facility 1and City 1 also connected to 4 at facility 2. The cities 2 and 4 are in same cluster but they do not use the same facility and also city 6 to city 7 with facility 2 have no distance. So it gives the infeasible solution.

4.4 Alphabet Table:-

There are $|m \times n \times p|$ ordered triples in the three-dimensional array X. For our convenience these are arranged in ascending order of their corresponding distance 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 DC be the array of cumulative sum of the elements of D. The arrays SN, D, DC, R, C, and K for the numerical example are given in the table-6. If p $\in$ SN then (R(p),C(p),K(p)) is the ordered triple and D(a)=D(R(a),C(a),K(a)) is the value of the ordered triple and DC (a) = $\sum_{i=1}^{a} D(i)$

Table-6

SN	D	DC	R	C	K
1	1	1	1	2	1
2	1	2	1	4	1
3	1	3	4	9	1
4	1	4	2	5	2
5	1	5	5	7	2
6	2	7	3	4	1
7	2	9	1	3	2
8	3	12	4	2	1
9	3	15	9	8	2
10	4	19	6	4	1
11	4	23	8	4	1
12	4	27	3	8	2
13	4	31	8	6	2
14	6	37	5	4	1
15	6	43	6	3	1
16	6	49	3	6	2
17	7	56	1	5	1
18	8	64	9	3	2
19	9	73	7	3	1
20	15	88	3	5	2
21	16	102	3	7	1
22	18	120	9	2	1
23	18	138	4	6	2
24	19	157	9	7	1
25	19	176	3	9	2
26	20	196	8	4	2

27	21	217	3	2	2
28	22	239	5	6	2
29	22	261	6	9	2
30	22	283	7	2	2
31	22	305	4	6	1
31	23	328	8	2	2
33	24	352	1	8	1
34	24	376	5	8	2
35	24	400	7	8	2
36	25	425	1	9	1
37	25	450	8	3	1
38	25	475	4	8	1
39	26	501	6	8	1
40	26	527	7	6	1
41	26	553	6	3	2
42	26	579	6	4	2
43	27	606	7	9	1
44	27	633	2	9	2
45	28	661	5	2	2
46	28	689	6	2	2
47	28	717	8	7	2
48	29	746	2	3	1
49	29	775	6	5	2
50	29	804	8	9	2
51	30	834	2	5	1
52	30	864	7	5	1
53	30	894	9	5	1
54	30	924	2	3	2
55	32	956	7	8	1
56	32	988	8	9	1
57	32	1020	9	3	1
58	32	1052	2	6	2
59	33	1085	3	5	1
60	33	1118	8	7	1
61	34	1152	6	5	1
62	34	1186	9	6	1
63	34	1220	6	8	2
64	36	1256	5	7	1
65	36	1292	7	6	1
66	36	1328	9	4	1
67	37	1365	2	9	1
68	37	1402	3	7	1
69	37	1439	2	7	2
70	38	1477	2	7	1
71	38	1515	8	5	1
72	38	1553	3	4	2
73	38	1591	5	9	1
74	39	1630	1	6	2
75	40	1670	4	3	1
76	40	1710	2	4	1
77	40	1750	7	5	2
78	42	1792	3	2	1
79	42	1834	9	8	1
80	42	1876	1	4	2
81	43	1919	8	5	2
82	44	1963	9	2	1
83	46	2009	9	6	2

84	49	2058	3	8	1
85	49	2107	4	3	2
86	50	2157	3	9	1
87	52	2209	5	6	1
88	54	2263	4	8	1
89	55	2318	7	3	2

Let us consider $19 \in SN$. It represents the ordered triple (R(19),C(19),F(19)) =(7,3,1).Then D(19) = 9 and DC(19) =73.

4.3. Definition of an Alphabet - Table and a word:

Let SN = (1,2,…) be the set of indices, D be an array of corresponding costs of the ordered triples and DC be the 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 $L_k = \{a_1, a_2, …., a_k\}$, $a_i \in$ 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 $a_i \leq a_{i+1}$, i = 1, 2,…., k-1. The set SN is defined as the "Alphabet-Table" with alphabetic order as (1, 2, …. n^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 called “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 n-1, since the words of length greater than n-1 are necessarily infeasible, as any feasible pattern can have only n-1 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 leader is said to be feasible, if the block of word, defined by it has at least one feasible word.

4.5. Value of the word:

The value of the (partial) word L_k, $V(L_k)$ is defined recursively as $V(L_k) = V(L_{k-1}) + TD(a_k)$ with $V(L_o) = 0$ where $TD(a_k)$ is the cost array arranged such that $TD(a_k) < TD(a_{k+1})$. $V(L_k)$ 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.

4.6. 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 $L_k = (a_1, a_2,…, a_k)$ 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 $L_4 = (2, 4, 8, 16) = V(L_4) = 1+1+3+6= 11$

$$LB(L_4) = V(L_4) + DC(a_4 + n - k) - DC(a_4)$$

$$= 11+ DC(16+ 8 - 4) - DC(16)= 11+ DC(20) - DC(16)= 11 + 88 – 49= 50$$

Where $DC(a_k) = \sum_{i=1}^{k} DC(a_i)$. It can be seen that LB (L_k) is the value of the complete word, which is obtained by concatenating the first (n-k) letters of SN (a_k) to the partial word L_k.

4.7. Feasibility criterion of a partial word

An algorithm was developed, in order to check 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 word. We will introduce some more notations which will be useful in the sequel.

- ➢ IR be an array where IR (i) = 1, $i \in N$ indicates that the i^th city is connected to some city j. Otherwise IR (i) = 0
- ➢ IC be an array where IC (j) = 1, $j \in N$ indicates that the i^th city is connected from some city i. Otherwise IC (j) = 0
- ➢ CL be an array, where CL (i) = N_i, indicates that the i^th city is belongs to N_i cluster, otherwise CL (i) = 0.
- ➢ SW be an array where SW (i) = j indicates that the i^th city is connected to some city j, Otherwise SW (i) = 0
- ➢ SWI be an array where SWI (j) = i indicates that the j^th city is connected from some city i, Otherwise SWI (j) = 0
- ➢ LW be an array where $L[i] = \alpha_i$, $i \in N$ □is the letter in the i^th position of a word.

The values of the arrays IR, IK, SW and LW 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.

SWI(C (a_i)) = R ((a_i)), i = 1, 2,…., k and SW (j) = 0 for other elements of j.

LW (i) = N_i, i = 1, 2,…., k, and LW (j) = 0 for other elements of j.

The recursive algorithm for checking the feasibility of a partial word L_p is given as follows. For this algorithm we have TR = R (a_{p+1}), TC = C (a_{p+1}) and TK = F (a_{p+1}). We start with the partial word $L_1 = (a_1) = (1)$. A partial word L_p is constructed as $L_p = L_{p-1} * (\alpha_p)$. Where * indicates chain formulation. We will calculate the values of V(L_p) and LB (L_p) 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 proceed to consider a partial word of under (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 p by taking another letter which succeeds a_p in the 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) $\geq$ VT. In this case we reject the partial word L_p. 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.

4.7 Algorithm- : (Lexi - Search algorithm)

STEP 0: (Initialization)

The arrays SN, D, DC, R, C, T and LN the values of N, M are made available IR,IC,SW,SWI,F, IK, L, V, LB are initialized to zero. The values I=1, J=0, LN (TR) =0, VT=∞, MAX= (n*n)/2

```
STEP 1:   J=J+1
          MAX= (n*n)/2
          LDP=0
          IS (J>MAX)                          IF YES GOTO 30
                                              IF NO GOTO 2
STEP2:    L (I) =J
          TR= R (J)
          TC= C (J)
          TK= T (J)                           GOTO 3

STEP3:    V (I) =V (I-1) +D (J)
          LB (I) =V (I) +CD (J+N-1-I)-CD (J))   GOTO 4

STEP4:    IS (LB (I)>=VT)                     IF YES GOTO 34
                                              IF NO GOTO 5

STEP5:    IS (TR= =HC)                        IF YES GOTO 6
                                              IF NO GOTO 7
STEP6:    IS (IR [TR] <P)                     IF YES GOTO 8
                                              IF NO GOTO 2
STEP7:    IS (IR [TR] = =1)                   IF YES GOTO 2
                                              IF NO GOTO 8
 STEP8:   IS(IC [TC] = =1)                    IF YES GO TO 2
                                              IF NO GOTO 9
STEP9:    W=TC                                GOTO 10
```

```
STEP10:     LDP=LDP + q[w]
            IS (LD<LDP)                        IF YES   GOTO2
                                               IF NO GOTO 12

STEP12:     IS (SW[w] = =0)                    IF YES   GOTO15
                                               IF NO GOTO 13

STEP13:     IS (W= =TR))                       IF YES   GOTO2
                                               IF NO GOTO 14

STEP14:     W=SW [W]                           GOTO10

STEP15:     W=TR                               GOTO16

STEP16:     LDP=LDP + q[w]                     IF YES   GOTO2
            IS (LD<LDP)                        IF NO GOTO 17
STEP17:     IS (SWI[w] = =0)                   IF YES   GOTO20
                                               IF NO GOTO 18

STEP18:     IS (W= =TC))                       IF YES   GOTO2
                                               IF NO GOTO 19

STEP19:     W=SWI [W]                          GOTO16

STEP20:     IS (b [TC] = =1)                   IF YES   GOTO21
                                               IF NO GOTO 25

STEP21:     IS (NB= =0)                        IF YES   GOTO22
                                               IF NO GOTO 23
STEP22:     B=b [TC]
            F [B] = TK                         GOTO24

STEP23:     IS (F [B] = = TK)                  IF YES   GOTO24
                                               IF NO GOTO 2

STEP24:     NB= NB+1                           GOTO 25

STEP25:     IS (i= =n-1)                       IF YES   GOTO28
                                               IF NO GOTO 26
STEP26:     L [I] =J
            IC [TC] =1
            SW [TR] =TC
            SWI [TC] =TR
            IS (TR= =HC)                       {IR [TR] =IR [TR] +1 GOTO 27}

                                               IR [TR] =1 GOTO 27}
STEP27:     I=I+1
            Z2=P-IR [HC]
            IS (Z2<=n-I)                       IF YES   GOTO2
                                               IF NO GOTO 29
STEP28:     T=V [I]
            L [I] =J                           GOTO 29

STEP29:     I=I-1                              GOTO 30

STEP30:     J=L [I]
            TR=R [J]
            TC=C [J]
```

```
                IR [TR] =0
                IC [TC] =0
                SW [TR] =0
                SWI [TC] =0
                L [I+1] =0
                IS (TR= =HC)                IR [HC] =    IR [HC]-1 GOTO 31}
                                            IR [TR] =0 GOTO 31}
STEP31:     IS (b [TC] = =1)                IF YES   GOTO32
                                            IF NO GOTO 2
STEP 32:    NB=NB-1                         GOTO 33

STEP33:     IS (I= =1)                      IF YES   GOTO34
                                            IF NO GOTO 29
STEP34:     STOP
```

4.8. 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 K 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 -7, A indicates ACCEPT, R indicates REJECT and CS indicates capacity of the particular city.

TABLE-7

SN	1	2	3	4	5	6	7	8	V	LB	R	C	K	REMARK
1	1								1	12	1	2	1	A,(K=1)
2		2							2	12	1	4	1	A,(K=1)
3			3						3	12	4	9	1	A,(K=1)
4				4					4	12	2	5	2	A
5					5				5	12	5	7	2	A
6						6			7	12	3	4*	1	R
7						7			7	13	1	3	2	A
8							8		10	13	4	2*	1	R
9							9		10	14	9	8	2	A
10								10	14	14	6	4*	1	R
11								11	14	14	8	4*	1	R
12								12	14	14	3	8*	2	R
13								13	14	14	8	6	2	R> CS
14								14	16	16	5	4*	1	R
15								15	16	16	6	3*	1	R
16								16	16	16	3	6	2	A,VT=16
17							10		11	15	6	4*	1	R
18							11		11	15	8	4*	1	R
19							12		11	15	3	8	2	A
20								13	15	15	8	6	2	A,VT=15
21							13		11	17*	8	6	2	R,>VT
22						8			8	15*	4	2	1	R,=VT
23					6				6	14	3	4*	1	R
24					7				6	16*	1	3	2	R,>VT
25				5					4	14	5	7	2	A
26					6				6	14	3	4*	1	R
27					7				6	16*	1	3	2	R,>VT
28				6					5	17	3	4	1	R,>VT
29			4						3	14	2	5	2	A
30				5					4	14	5	7	2	A

31					6				6	14	3	4*	1	R
32					7				6	16*	1	3	2	R,>VT
33				6					5	17*	3	4	1	R,>VT
34			5						3	17*	5	7	2	R,>VT
35		3							2	14	4	9	1	A
36			4						3	14	2	5	2	A
37				5					4	14	5	7	2	A
38					6				6	14	3	4	1	A
39						7			8	14	1	3	2	A
40							8		11	14	4	2*	1	R
41							9		11	15*	9	8	2	R,=VT
42						8			9	16*	4	2	1	R,>VT
43					7				6	16*	1	3	2	R,>VT
44				6					5	15*	3	4	1	R,=VT
45			5						3	17*	5	7	2	R,>VT
46		4							2	17*	2	5	2	R,>VT
47	2								1	12	1	4	1	A
48		3							2	12	4	9	1	A
49			4						3	12	2	5	2	A
50				5					4	12	5	7	2	A
51					6				6	12	3	4*	1	R
52					7				6	16*	1	3	2	R,>VT
53				6					4	16*	3	4	1	R,>VT
54			5						3	17*	5	7	2	R,>VT
55		4							2	17*	2	5	2	R,>VT
56	3								1	17*	4	9	1	R,>VT

At the end of the search the current value of VT is 15 and it is the value of the optimal feasible word $L_{7\,=}$(1, 2, 3, 4, 5, 7, 12, 13). It is given in the 20th row of the search table. The arrays IR, IC, SW, SWI and LW take the values represented in the table-8 given below. Hence the pattern gives the optimal feasible word.

TABLE-8

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
IR	1,1,1	1	1	1	1	-	-	1	-
IC	-	1	1	1	5	1	1	-	1
SW	2,3,4	5	8	9	7	-	-	6	-
SWI	-	1	1	1	2	8	5	3	4
LW	1	2	3	4	5	7	12	13	

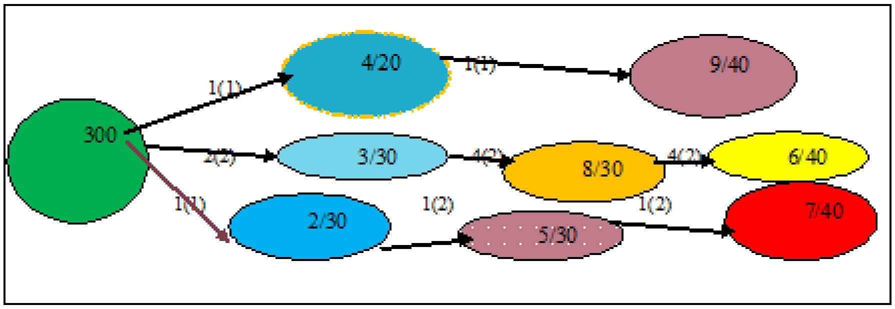
From the above arrays optimal feasible word $L_{7\,=}$ (1, 2, 3, 4, 5, 7, 12, 13) and the respective ordered triple set{(5,1,1),(8,5,1),(7,8,2),(2,1,2),(3,2,1),(4,2,2),(6,4,1)}represents the pattern given in the table-9

TABLE-9

$$X(i, j, 1) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad X(i, j, 2) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The paths represented by the above pattern is [(1,2,1),(1,3,2),(1,4,1),(2,5,2),(5,7,2),(3,8,2),(8,6,2), (4,9,1)]. The cities {2, 4, 9} using the same facility 1. The diagrammatic representation of this solution can also see in the following figure-3.

Figure-3



The above figure-3 represents a feasible solution. In first path the city 1 is connected to the city 4 at facility 1 with 20units, city 4 is connected to city 9 at facility 1, with 40 units. In second path, the city 1 is connected to city 3 at facility 2 with 30 units, city 3 is connected to city 8 with facility 2 with 30 units, and city 8 is connected to city 6 at facility 2 with 40 units. In third path city 1is connected to city 2 at facility 1 with 30 units, city 2 is connected to city 5 at facility2 with 30 units; city 5 is connected to city 7 at facility 2 with 40 units. Hence the solution is

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

$$=1+1+1+1+1+2+4+4= 15$$

5. EXPERIMENTAL RESULTS:

The following table shows that the computational results for proposed algorithm called pattern recognition technique based on 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. 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 N, K and CL. The results are tabulated in the Table -10 are given below. For each instance, seven to nine data sets are tested. It is seen that the time required for the search of the optimal solution is fairly less.

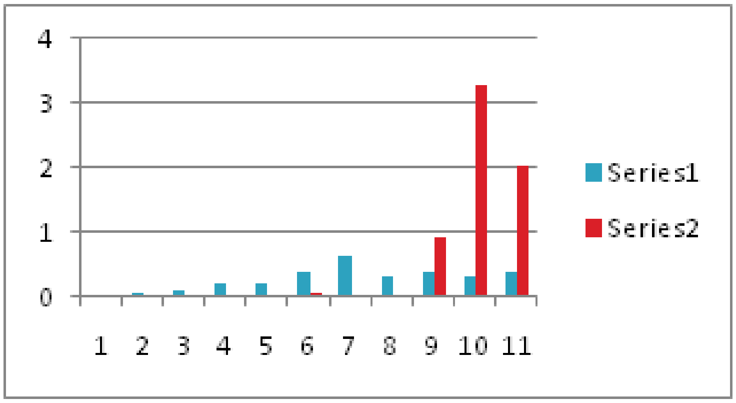
Table-10

SN	**Problem dimension**			**NPT**	CPU runtime in seconds	
	N	**K**	**CL**		**Avg AT**	**Avg ST**
1	4	2	1	7	0	0
2	4	2	2	7	0	0
3	5	2	1	7	0	0
4	5	3	1	7	0.054945	0
5	10	2	1	7	0.109890	0
6	10	3	1	7	0.21978	0
7	10	3	2	7	0.219780	0
8	15	2	1	9	0.384615	0.054945
9	15	3	1	9	0.659341	0
10	15	3	2	9	0.329670	0
11	20	2	1	9	0.384615	0.934066
12	20	2	2	9	0.329670	3.296703
13	20	3	2	9	0.384615	2.032552

In the above table the notations are represents that N= number of cities, K = 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.

The graphical representation of the above instances is given below. In the following Graph –1, 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 colors 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

GRAPH-1



6. COMPARISION RESULTS

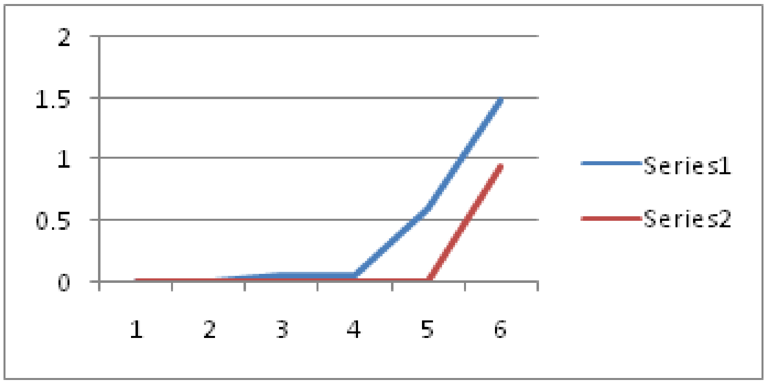
We implement the Pattern Recognition Technique based Lexi Search Algorithm (LSA) with C language for this model. We tested the proposed algorithm by different set of problems and compared the computational results with the published minimum spanning tree model by C. Suresh Babu Volume 3, No. 1, Jan-Feb 2012 International Journal of Advanced Research in Computer Science. Then Table-11 shows that the comparative results of different sizes.

TABLE-11

SN	**N**	**M**	**CL**	**NPT**	**T**	**CPU Run Time in seconds**	
						Published model	Proposed model
1	4	2	1	5	99	0	0
2	5	3	1	5	99	0	0
3	10	2	3	5	99	0.05490	0
4	10	3	1	5	99	0.054945	0
5	15	3	1	5	99	0.604396	0
6	20	2	1	5	99	1.483516	0.934066

In the above table 11, 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 0.934066 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 6 instances is given below. In the Graph-2, X-axes taken the SN and Y-axes taken the values of CPU run time for the published and proposed models.

GRAPH-2



From the above Graph-2, 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.

7. CONCLUSION

In this chapter we studied a model, “Three dimensional P- trucks route minimum cost supply to the cities from the head quarter city” and developed a Lexi- search algorithm based on pattern recognition technique to solve the model. The model is then formulated as a zero one programming problem. A suitable numerical example is quoted for better understood the concepts and the steps involved in the algorithm. We programmed the proposed algorithm using C-language. The computational details are reported.AS an observation CPU run time is fairly less for higher dimensional problem, it gives an optimal solution. Moreover, Lexi search algorithms are proved to be more efficient in many combinatorial problems. Many researchers have used different types of alphabet tables and have shown that their Lexi -search algorithms are efficient and are faster. Based on this experience we strongly feel that this algorithm can perform larger size problems and more over it is very efficient.

8. 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..

Source of support: Nil, Conflict of interest: None Declared