Volume 3, No. 1, Jan-Feb 2012

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Variant Minimum Spanning Network Connectivity Problem

C.Suresh Babu
Research Scholar
Department of Mathematics
Sri Venkateswara University
Tirupati, A.P., India
suresh8044@gmail.com

Dr.K.Sobhan Babu*
Assistant Professor
Department of Mathematics
U C E K, J N T U K A K I N A D A
Kakinada, A.P., India
sobhanjntu@gmail.com

Dr.M.Sundara Murthy
Senior Professor (Retd)
Department of Mathematics
Sri Venkateswara University
Tirupati, A.P., India
profmurthy@gmail.com

Abstract: Many Combinatorial programming problems are *NP-hard* (Non Linear Polynomial), and the theory of NP-completeness has reduced hopes that NP-hard problems can be solved within polynomials bounded computation times. Nevertheless, sub-optimal solutions are sometimes easy to find. Consequently, there is much interest in approximation and heuristic algorithms that can find near optimal solutions within reasonable running time. We consider a minimum spanning tree problem and apply the Lexi-Search algorithm base d on the Pattern Recognition which takes care of simple combinatorial structure of the problem and computational results are reported.

Keywords: Minimum Spanning Tree Problem, Lexi-Search, word, Pattern Recognition

I. INTRODUCTION

Many combinatorial optimization problems are *NP-hard* (Non Linear Polynomial), and the theory of *NP*-completeness has reduced hopes that *NP-hard* problems can be solved within polynomials bounded computation times. Nevertheless, sub-optimal solutions are sometimes easy to find. Consequently, there is much interest in approximation and heuristic algorithms that can find near optimal solutions within reasonable running time [1].

In mathematical programming, a *heuristic method* or *heuristic* for short is a procedure that determines good or near-optimal solutions to an optimization problem. As opposed to exact methods, heuristics carry no guarantee that an optimal solution will be found. Practically, for many realistic optimization problems good solutions can be found efficiently and heuristics are typically among the best strategies in terms of efficiency and solution quality for problems of realistic size and complexity. Heuristics can be classified as either *constructive* (greedy) or as *local search* heuristics. The former are typically one-pass algorithms whereas the latter are strategies of *iterative improvement.* Useful references on heuristic methods can be found in [1], [2] and [3]. Given an undirected graph whose nodes are partitioned into a number of subsets (clusters), the Generalized Minimum Spanning Tree (GMST) problem is then to find a minimum-cost tree which includes *exactly* one node from each cluster.

In this paper we study the problem called variant minimum spanning network connectivity problem.

In D(i, j, k), 'k' stands for the third dimension which is generally called time/facility, but it is not the usual continuous time. It stands for another independent factor which influences the cost 'D'. The cost generally depends on 'i' and 'j'. For example in the case of cost or distance it depends not only on i, j, the third factor may be the nature of vehicle used (i.e. Petrol vehicle or diesel vehicle or luxury vehicle etc.). Let N = {1, 2 . . n} cities and N × N distances matrix. The problem is to find the minimum spanning connectivity cost/distance of all the n-1 cities to the headquarters city {1}.

According to Kruskal, J.B. if a (finite) connected graph has a positive real number attached to each edge (the length of the edge), and if these lengths are all distinct, then among the spanning trees of the graph there is only one, the sum of whose edges is a minimum; that is, the shortest spanning tree of the graph is unique [4].

Let α^1 be the city either headquarters {1} or a city connected to the headquarters and $d(\alpha_i, \alpha^1)$ is the cost/distance then the problem is

II. MATHEMATICAL FORMULATION

$$Minimize\ Z(x) = \sum_{\alpha_i} D(\alpha_i, \alpha_i') \text{ ----- (1)}$$

$$\alpha_i \in N - \{1\}$$

$$\sum_i \sum_j X(i,j) = |N| - 1 \quad \text{---------- (2)}$$

$$i, j \in N$$

X(i, j) = 0 or 1 ---------- (3)

The above one is a two dimensional problem. 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 where i, j ∈ N; N = {1,2, . . ., n} and K = {1,2, . . ., k}. Then the three dimensional problem is:

$$Minimize\ Z(x) = \sum_{\alpha_i} \sum_{k \in K} D(\alpha_i, \alpha_i^1, k) X(i,j,k) \text{ ----(4)}$$

$$\alpha_i \in N - \{1\}$$

Subject to

$$\sum\sum\sum X(i,j,k) = |N| - 1 - - - - - - - (5)$$

$X(i_1, j_1, k_1) = X(i_2, j_2, k_2) = 1$

and $i_1 \in N_i$, $i_2 \in N_j$

if i=j then $k_1 = k_2$ and if $i \neq j$ then $k_1 \neq k_2$ ----- (6)

$$UN_i = N;\ \forall 1,2,3 \ldots p - - - - - - - (7)$$

i.e, if i_1 contains N_1 and i_2 contains N_2 then $k_1$and k_2 should be used different facilities

In the sequel we developed a Lexi-search 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.

III. NUMERICAL ILLUSTRATION

The concepts and the algorithm developed will be illustrated by a numerical example for which total number of cities N = {1, 2, 3, 4, 5, 6, 7, 8} here we divide N into two clusters as N_1 and N_2 cluster N_1 = {3, 5, 6, 8} and cluster N_2 = {2, 4, 7} and 1 as a head quarter.

Facility k = {1, 2} the following table- 1 represent that the cities which are belongs to the respective clusters

Table-1

Cities	1	2	3	4	5	6	7	8
Respective cluster	-	2	1	2	1	1	2	1

The cost matrices are given as follows.

Table-2

$$D(i,j,1) = \begin{bmatrix} - & - & - & - & - & - & - & - \\ - & \infty & - & 8 & 13 & 17 & - & 20 \\ 23 & 9 & \infty & - & 3 & 22 & - & - \\ - & 12 & - & \infty & 19 & 8 & 21 & - \\ 16 & - & 10 & 11 & \infty & 15 & 5 & 7 \\ 15 & - & 13 & 9 & - & \infty & 3 & 1 \\ - & 5 & 15 & - & 20 & 7 & \infty & 16 \\ 14 & - & - & 7 & 4 & - & 19 & \infty \end{bmatrix}$$

Table-3

$$D(i,j,2) = \begin{bmatrix} - & - & - & - & - & - & - & - \\ 2 & \infty & 9 & 21 & - & 4 & - & 15 \\ 16 & 5 & \infty & - & 12 & 9 & 17 & - \\ - & 8 & 10 & \infty & 19 & - & 21 & 11 \\ 12 & 9 & - & - & \infty & 8 & 4 & - \\ 19 & - & 17 & 14 & 18 & \infty & - & - \\ - & 6 & 16 & - & 19 & 11 & \infty & 18 \\ 13 & 9 & - & 18 & 12 & 9 & 14 & \infty \end{bmatrix}$$

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 cost can't as well as negative quantities. Suppose D(2, 4, 2) = 21 means the cost of the connecting the city 2 to 4 by using facility 2 is 21.

IV. CONCEPTS AND DEFINITIONS

A. *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.

$$V(x) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} D(i,j,k)X(i,j,k)$$

The value V(X) gives the total time of the assignment for the solution represented by X. Thus the value of the feasible pattern gives the total time 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.

There are M= $m \times n \times p$ 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 [3][4]. 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, F 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, F for the numerical example are given in the table-4. If p$\in$ SN then (R(p),C(p),F(p)) is the ordered triple and D(a)=T(R(a),C(a),F(a)) is the value of the ordered triple and

$$DC(a) = \sum_{i=1}^{a} D(i)$$

Table-4 (Alphabet Table)

SN	D	DC	R	C	F
1	1	1	6	8	1
2	2	3	2	1	2
3	3	6	3	5	1
4	3	9	6	7	1
5	4	13	8	5	1
6	4	17	2	6	2
7	5	22	5	7	1
8	5	27	7	2	1
9	6	33	7	2	2
10	7	40	5	8	1
11	7	47	7	6	1
12	8	55	4	2	2
13	8	63	2	4	1
14	8	71	4	6	1
15	8	79	3	2	2
16	8	87	5	6	2
17	9	96	3	2	1
18	9	105	4	5	1
19	9	114	6	4	1
20	9	123	8	6	2
21	9	132	2	3	2
22	9	141	3	6	2
23	9	150	5	2	2
24	9	159	8	2	2
25	10	169	5	3	1
26	10	179	3	7	2
27	10	189	4	3	2
28	11	200	5	4	1
29	11	211	4	8	2
30	11	222	7	6	2

31	12	234	4	2	1
32	12	246	3	5	2
33	12	258	5	1	2
34	12	270	8	5	2
35	13	283	2	5	1
36	13	296	6	3	1
37	13	309	8	1	2
38	14	323	8	1	1
39	15	338	2	8	2
40	16	354	5	1	1
41	16	370	7	8	1
42	16	386	3	1	2
43	16	402	7	3	2
44	17	419	2	6	1
45	17	436	6	3	2
46	18	454	4	2	2
47	18	472	6	5	2
48	18	490	7	8	2
49	18	508	8	4	2
50	19	527	4	5	2
51	19	546	6	1	2
52	19	565	6	1	2
53	19	584	7	5	2
54	20	604	2	8	1
55	20	624	7	5	1
56	21	645	4	7	1
57	21	666	2	4	2
58	21	687	4	7	2
59	22	709	3	6	1
60	23	732	3	1	1
61	-	-	-	-	-
62	-	-	-	-	--
63	-	-	-	-	-
-	-	-	-	-	-
120-	-	-	-	-	-

Let us consider 21∈ SN. It represents the ordered triple (R(21),C(21),F(21))=(2,3,2).Then D(21)=T(2,3,2)= 69 and DC(21)=132.

B. Definition of Alphabet-Table and word:

Let SN = (1,2,…) be the set of indices, D be an array of corresponding costs of the ordered triples and DCT be the array of cumulative sums of elements in D. Let arrays R, C and F 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 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 atmost 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 = 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 [5].

V. 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, [5],and [6].

VI. 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.

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

VII. 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∈ N indicates that the i^{th} city is connected to some city j.
Otherwise IC (i) = 0
IK be an array where IK (k) = 1, k ∈ K indicates that the k^{th} facility is utilized by a city i to connect the city j.
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
LW be an array where L[i] = α_i, i∈ N is the letter in the i^{th} position of a word.
The values of the arrays IR, IK, SW,LW are as follows
IR (R (a_i)) = 1, i = 1, 2, - - - - - , k and IR (j) = 0 for other elements of j
IK (T (a_i)) = 1, i = 1, 2, - - - - - , k and IT (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
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 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_{p+1}), TC = C (a_{p+1}) and TK = F (a_{p+1}).

Algorithm-1:

STEP1: IX=0
STEP 2: IS (IR(TR) = 1) IF YES GOTO 13
IF NO GOTO 3
STEP3: W = TC GOTO 4
STEP 4: IS W = TR IF YES GOTO 13
IF NO GOTO 5
STEP5: IS SW(W) = 0 IF YES GOTO 7
IF NO GOTO 6

STEP 6; W = SW (W) GOTO 4
STEP7:IS TK = CL (TR) IF YES GOTO 8
IF NO GOTO 9
STEP 8 : IX=1
STEP 9 : STOP

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.

a. 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).
b. 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.

Algorithm- 2: (Lexi - Search algorithm)

STEP 1 : (Initialization)
The arrays SN, D, DC, R, C, T and LN the values of N, M are made available IR, IK, L, V, LB are initialized to zero. The values I=1, J=0, LN (TR) =0, VT= ∞, NZ=M*N*K-1 MAX=NZ-1
STEP 2 J=J+1
IS (J>MAX) IF YES GOTO 14
IF NO GOTO 3
STEP 3: L (I) = J
IS (I = 1) IF YES V (I) =D (J) GOTO 3B
IF NO GOTO 3A
STEP 3A V (I) =V (I-1) +D (J) GOTO 3B

STEP 3B:LB (I) =V (I) +DC (J+N-I)-DC (J) GOTO 4
STEP 4: IS (LB (I)$\geq$VT) IF YES GOTO 10
IF NO GOTO 5
STEP 5 : TR=R (J)
TC=C (J)
TK=T (J) GOTO 6
STEP 6: CHECK THE FEASIBILITY OF L (USING ALGORITHM-1)
IS (IX=0) IF YES GOTO 2
IF NO GOTO 7
STEP 7 : IS (I=N) IF YES GOTO 8
IF NO GOTO 9
STEP 8: L (I) = J
L (I) IS FULL LENGTH WORD AND IS FEASIBLE.
VT=V (I), RECORD L, VT GOTO 12
STEP9 : IR (TR) =1
SW(TR) = TC

I = I + 1 GOTO 2
STEP10 :IS (I=1) IF YES GOTO 13
IF NO GOTO 11
STEP11 : I=I-1 GO TO 12
STEP12 : J=L (I)
TR = R (J)
TK = T (J)
IR (TR) =0 GOTO 2
STEP13 : STOP & END.

Table-5 (Search - Table)

SN	1	2	3	4	5	6	7	TD	LB	R	C	T	REM
1	1							1	22	6	8	1	A
2		2						3	22	2	1	2	A
3			3					6	22	3	5	1	A
4				4				9	22	6	7	1	R
5				5				10	24	8	5	1	A
6					6			14	24	2	6	2	R
7					7			15	26	5	7	1	A
8						8		20	26	7	2	1	R
9						9		21	28	7	2	2	A
10							10	28	28	5	8	1	R
11							11	28	28	7	6	1	R
12							12	29	29	4	2	2	A= 29
13						10		22	29	5	8	1	R
14					8			15	28	7	2	1	R
15				6				10	26	2	6	2	R
16				7				11	29	5	7	1	R
17			4					6	22	6	7	1	R
18			5					7	27	8	5	1	A
19				6				11	27	2	6	2	R
20				7				12	30	5	7	1	R
21			6					7	30	2	6	2	R
22		3						4	30	6	7	1	R
23	2							2	26	2	1	2	A
24		3						5	26	3	5	1	A
25			4					8	26	6	7	1	A
26				5				12	26	8	5	1	A
27					6			16	26	2	6	2	R
28					7			17	28	5	7	1	A
29						8		22	28	7	2	1	R
30						9		23	30	7	2	2	R
31					8			17	30	7	2	1	R
32				6				12	28	2	6	2	R
33				7				13	31	5	7	1	R
34			5					9	29	8	5	1	R
35		4						5	29	6	7	1	R
36	3							3	30	3	5	1	R

The partial word is L_8 = (1, 2, 3, 4, 7, 8, 9, 12) is a feasible partial word. For this partial word the array IR, IC, IT, LW are given in the following Table –6.

Table-6

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
LW	1	2	3	5	7	9	12	-
IR	-	1	1	1	1	1	1	1
IK	-	2	1	2	1	1	2	1
SW	-	1	5	2	7	8	2	5

At the end of the search the current Value of VT is 29 and it is the value of optimal feasible word. L_7 = (1, 2, 3, 5, 7, 9, 12). It is given in the 12^{th} row of the search table-5.

VIII.COMPUTATIONAL RESULTS

A Computer program for the proposed algorithm is written in C language and is tested on the COMPAQ system. We tried a set of problems for different sizes. Random numbers are used to construct the Time matrix. The following table-7 gives the list of the problems tried along with the average CPU time in seconds required for solving them.

Table – 7 (Computational Results)

S.No	**Problem Dimensions**		**AT**	**Type-I**			**Type-II**			**Type-III**		
	M	n		Min	Max	Avg	Min	Max	Avg	Min	Max	avg
1	10	10	0.009	0.060	0.095	0.0775	0.075	0..0960	0.0855	0.0967	0.0845	0.0906
2	15	15	0.017	0.09	0.075	0.0825	0.660	0.945	0.8016	0.89	0.941	0.917
3	25	25	0.965	1.26	1.34	1.3	1.241	1.675	1.458	1.654	1.6	1.627

IX. CONCLUSIONS

The problems are solved by using Lexi-search algorithm based on the pattern recognition technique. The cost matrix was generated randomly in the interval [0,100]. Our algorithm has been implemented in C program. The computational experiments were performed on a personal computer with AMD Sempron™ Processor LE-1200, 2.10GHz, 896RAM and OS Windows XP Profesional. In the table-7 we have presented the computational results for solving the problem using the Lexi-Search algorithm based on the Pattern Recognition Technique.

X. REFERENCES

[1]. Osman, I.H. and Laporte, G., "Metaheuristics: A Bibliograpgy", Annals of Operations Research, 63, 513 – 623.

[2]. Reeves, C.R., "Moderen Metaheuristics Techniques for Combinatorial Problems", Blackwell, Oxford, 1993.

[3]. 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.

[4]. Kruskal, J.B., "On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings to the Americal Mathematical Society 7, 1956, 48 – 50.

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

[6]. 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.