ISSN No. 0976-5697
Volume 1, No. 3, Sept-Oct 2010
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

A Lexi-Search Approach for Variant Mutiple Travelling Salesmen Problem

K.Sobhan Babu*
Department of Mathematics
University College of Engineering, Jntuk, Kakinada
Andhra Pradesh, India
sobhanjntu@gmail.com

Chandra Kala.K
Assistant System Engineer
Tata Consultancy Services, Hyderabad
Andhra Pradesh, India
chandrakala.kuruba@tcs.com

B. Naganna
Associate Professor, Department of Mathematics
Vardhaman College of Engineering, Shamshabad,
Hyderabad, Andhra Pradesh, India
naganna.bandi@gmail.com

Sundara Murthy. M
Senior Professor, Department of Mathematics
Sri Venkateswara University, Tirupati, Chittor (Dt)
Andhra Pradesh, India
profmurthy@gmail.com

Abstract: The multiple travelling salesmen problem (mTSP) is a generalization of the well-known travelling salesman problem (TSP), where more than one salesman is allowed to be used in the solution. More over, the characteristics of the mTSP seem more appropriate for real-life applications, and it is also possible to extend the problem to a wide variety of vehicle routing problems (VRPs) by incorporating some additional side constraints. Although there exists a wide body of the literature for the TSP and the VRP, the mTSP has not received the same amount of attention. In this paper we develop an efficient Lexi-Search method for solving the multiple travelling salesmen problem. Although Lexi-Search methods are among the most widely used techniques for solving hard problems, it is still a challenge to make these methods smarter. The motivation of the calculation of the lower bounds is based on ideas frequently used in solving problems. Computationally, the algorithm extended the size of the problem and find better solution

Keywords: Multiple Travelling Salesmen Problem, Lexi-Search, Pattern Recognition, Tour, Alphabet Table, Search Table.

I. INTRODUCTION

A generalization of the well-known traveling salesman problem (TSP) is the multiple traveling salesmen problem (mTSP), which consists of determining a set of routes for m salesmen who all start from and turn back to a home city (depot). Although the TSP has received a great deal of attention, the research on the mTSP is limited. In this paper we develop an efficient Lexi-Search method for solving the multiple travelling salesmen problem. Although Lexi-Search methods are among the most widely used techniques for solving hard problems, it is still a challenge to make these methods smarter. The motivation of the calculation of the lower bounds is based on ideas frequently used in solving problems. Computationally, the algorithm extended the size of the problem and find better solution.

II. MATHEMATICAL FORMULATION

$$\underset{x}{Min\, Z(x)} = \underset{i=1,2,\dots m}{Max\, Z}\left[\sum_{i\in\overline{N_i}}\sum_{j\in\overline{N_i}}\sum_{k\in K} t(i,j,k)\,X(i,j,k)\right] \quad \text{---------- (1)}$$

Where , the i^{th} salesman visits the n_i cities and =
,

$$\sum_{i\in\overline{N_i}}\sum_{j\in\overline{N_i}}\sum_{k\in K} X(i,j,k \quad \text{-------- (2)}$$

$i = 1, 2. . . m$

--------- (3)

Constraint (2) represents that the i^{th} salesman visits in his tour n_i cities starting from headquarter {1}. Another constraint of the salesman is that each salesman in their N_i tours with n_i cities make the tour starting from {1} headquarter city visits the n_i cities and returns to the headquarter city.

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. PROPOSED ALGORITHM

The name ***Lexicographic-search*** or ***Lexi-search*** method implies that the search is made for an optimal solution in a systematic way, just as one search for meaning of a word in a dictionary. When the process of feasibility checking of a partial word becomes difficult, though lower bound computation is easy, ***Pattern Recognition Technique*** (Sundara Murthy, 1979) can be used. Lexi-Search algorithms, in general, require less memory, due to the existence of Lexicographic order of partial words. If Pattern Recognition Technique is used, the dimension requirement of the problem can be reduced, since it reduces to the two-dimensional cost array into a linear and the problem can be reduced to a linear form of finding an optimal word of length n (Sundara Murthy, 1979) and hence reduces computational work in getting an optimal solution.

IV. CONCEPTS AND DEFINITIONS OF OUR ALGORITHM

A. *Pattern*

An indicator matrix X, associated with an appropriate assignment of tasks to the agents is defined as a *Pattern*. A Pattern is said to be **feasible,** if X is feasible.

Each pattern X can also be represented by the set of all ordered triples {(i, j, k)}, for which X (i, j, k) =1. In general, there will be m*n*k ordered pairs in a matrix X (m, n, k).

B. Alphabet – Table & Word

Let SN = (1,2, . . . ,n^3) be the set of indices, BD be an array of corresponding costs of the ordered pairs and DD be the array of cumulative sums of elements in BD. Let arrays R, C and T be respectively row, column and time/facility indices of the ordered triples. Let $L_k = (a_1,a_2. . .a_k)$. a_iε SN be a ordered sequence of k indices from S. The pattern represented by the ordered triples indices are given by L_k is independent of the order of a_i in the sequence. For uniqueness, the indices in L_k are arranged in increasing order, such that $a_i < a_{i+1}$, i = 1,2, . . .,k-1. The set S is defined as ***Alphabet-Table*** with alphabetic order as (1,2, . . .,n^3) and the ordered sequence L_k is defined as a word of length k. A word L_k is said to be *sensible* word if $a_i < a_{i+1}$, i = 1, 2. . . k-1; *non sensible* otherwise. It is said to be feasible, if it represents a ***feasible*** pattern. Any of the letters in S can occupy the first place in a word L_k. Our interest is only in set of words of length atmost equal to n, 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 words defined by it has at least one feasible word.

C. Value of the Word

The value of the (partial) word L_k, $V(L_k)$ is recursively defined as $V(L_k) = V(L_{k-1}) + BD(a_k)$ with $V(L_0)$ =0, where BD is the cost array arranged such that, $BD(a_k) < BD(a_{k+1})$. V (L_k) and the value of the pattern X, will be the same, since X is the (partial) pattern represented by L_k.

D. Feasibility Criterion of a Partial Word

A recursive algorithm is developed for checking the feasibility of a partial word $L_{k+1} = (a_1, a_2… a_k, ak_{+1})$ given that L_k is a feasible partial word. We will introduce some more notations which will be useful in the sequel. (The feasible checking algorithm is written for m=2 and it can be easily be generalized for 'm').

IR be an array where IR (i) =1, i N represents that the salesman is visiting some city from city i, otherwise zero.

IC be an array where IC (i) =1, i N represents that the salesman is coming to city from some city i, otherwise zero.

IT be an array where IT (i), i P represents that the salesman visits a pair of cities at some time.

LW be an array where LW (i) is the letter in the i^{th} position of a word.

SWI be an array where SWI (i) is the city the salesman is visiting from city i at some time, otherwise SWI (i) = 0.

S be an array where S (i) indicates that the i^{th} salesman

Then for a given partial word $L_k = (a_1, a_2. . . a_k)$ the values of the arrays IR, IC, IT, S, LW, SWI are as follows:

$IR(R(a_i))=1$, i= 1,2,3, …k and IR(j)=0 for other elements of j.

IC(C (a_i) =1, i= 1, 2, 3 ….k otherwise IC (j) =0

SW(R (a_i)) =C (a_i), i=1, 2, 3….k otherwise SW (j) =0

LW (i) =a_i, i=1,2,….k otherwise LW(i)=0

SWT(R (a_i)) =T (a_i), i=1,2,…k otherwise SWT (j)=0

SWI (C (a_i)) =R (a_i), i=1,2,…k otherwise SWI (j)=0

ALGORITHM 1

```
STEP 0: IS IX=0                      IF YES GOTO 1
                                     IF NO GOTO 11
STEP 1: IS (IR (RA) =1)              IF YES GOTO 11
                                     IF NO GOTO 2
STEP 2: IS (IR (CA) =1)              IF YES GOTO 11
                                     IF NO GOTO 3
STEP 3: I = I + 1                    IF YES GOTO 11
              IS (I > M)             IF NO GOTO 4
STEP 4: MS (I) = S (I)
              S (I) = N (I)          GOTO 5
STEP 5: CAX = IC (CA) + 1            IF YES GOTO 6
RAX = IR (RA) +1                     IF NO GOTO 3
IS [ (RAX ≤ N(I)) & (CAX ≤ N(I))]
STEP 6: WI = CA                      GOTO 7
STEP 7: IF (SWI (WI)) = 0            GOTO 9
                            ELSE     GOTO 8
STEP 8: IF (WI = RA)                 GOTO 10
                            ELSE     WI = SW (WI)
                                     GOTO 7
STEP 9: IX =1                        GOTO 10
STEP 10:IF K = NI                    GOTO 9
                            ELSE     GOTO 11
STEP 11:      STOP & END
```

We start the algorithm with a very large value say M=9999 as a trial value (VT). If the value of a feasible word is known, we can start with the value as VT. During the search value of VT is improved. At the end of search the current value of VT gives the optimal feasible word. We start with the partial word $L_1=(a_1)=(1)$. Then two situations arises one for branching and the other for continuing the search.

1. $LB(L_p) < VT$. Then we check whether L_p is feasible or not. If it is feasible we proceed to consider a partial word of order (p+1) which represents a sub-block of the block of words represented by L_p. If L_p is not feasible then consider the next partial word of order (p-1).

2. $LB(L_p) \geq VT$. In this case 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.

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

ALGORITHM 2: (LEXI-SEARCH ALGORITHM)

The following algorithm gives an exact solution for the proposed problem.

STEP 1 : (INITILIZATION)

The arrays SN, D, DC, R, C, T, N, P and M are made available. IR, IC, IT, SW, SWT, SWI, LW, V, LB arrays are initialized to zero. The values I = 1, j=0, VT = 9999.

```
STEP 2    :J = J + 1              GOTO 3
STEP 3    :L (I) = J
           JA = J + N – 1
           V (I) = V (I – 1) + D (J)
           LB (I) = V (I) + DC (JA) – DC (J)
STEP 4    :IS (LB (I) ≥ VT) IF YES GOTO 12
```

IF NO GOTO 5

STEP 5 : RA = R (J)
CA = C (J)
TA = T (J)
GOTO 6

STEP 6 : CHECK THE FEASIBILITY OF L (1) (USING THE ALGORITHM 1)

IS (IX = 0)
IF YES GOTO 2
IF NO GOTO IS (IXA =1)
IF YES GOTO 8
IF NO GOTO 7

STEP 7 : IS (I = N)
IF YES GOTO 11

IF NO GOTO 9

STEP 8 : SW (RA) = CA
PRINT (I, SW (I), I =1 TO N)
GOTO 9

STEP 9 : L (I) = J
IR (RA) = 1
IC (CA) = 1
IT (TA) = IT (TA) +1
SW (RA) = CA
SWT (RA) = TA
SWI (CA) = RA
GOTO 10

SETP 10 : I = I + 1 GOTO 2

STEP 11 : L (I) = J
L (I) is a full length of word and is feasible
VT = V (I), RECORD L (I) and VT GOTO 13

STEP 12 : IS (I =1)
IF YES GOTO 14

STEP 13 : I = I -1
J = L (I)
RA = R (J)
CA = C (J)
TA = T (J)
IR (RA) = 0
IC (CA) = 0
IT (TA) = IT (TA) -1
SW (RA) = 0
SWT (RA) = 0
SWI (CA) = 0
GOTO 2

STEP 14 : STOP

The current value of VT at the end of the search is the value of the optimal solution for a feasible word. At the end if VT=9999 it indicates that there is no feasible solution.

V. COMPUTATIONAL EXPERIENCE

The cost matrix was generated randomly in the interval [0,100]. For each type of instance we considered six trials. Our algorithms have been implemented in C. The computational experiments were performed on a personal computer with AMD Sempron™ Processor LE-1200, 2.10 GHz, 896 RAM and OS Windows XP Professional. In table-1 we have presented the computational results for solving the problem using the Lexi-Search algorithm based on the Pattern Recognition Technique.

Table I. Time taken by the proposed algorithm

Sr. No.	*n*	*m*	*Alphabet Table*			*Total TimeTaken*		
		m_1, m_2, m_3	Min	Max	Avg	Min	Max	Avg
1	10	*5, 4, 3*	0.002	0.002	0.002	0.004	0.005	0.0045
2	10	*4, 5, 3*	0.001	0.002	0.0015	0.001	0.001	0.001
3	10	*3, 5, 4*	0.0023	0.0024	0.00235	0.0022	0.0024	0.0023
4	20	*6, 9, 7*	0.04	0.04	0.04	0.03	0.04	0.035
5	20	*6, 7, 9*	0.012	0.014	0.013	0.012	0.016	0.014
6	20	*9, 6, 7*	0.040	0.048	0.044	0.048	0.060	0.045
7	30	*7, 10, 15*	0.007	0.008	0.0075	0.0065	0.0089	0.0075
8	30	*10, 7, 15*	0.03	0.03	0.03	0.039	0.031	0.035
9	30	*10, 10, 12*	0.09	0.091	0.09	0.12	0.14	0.13

VI. CONCLUSION

The problems are solved by using the Lexi-Search algorithm based on the Pattern Recognition Technique. In table-1, the number of salesmen (m = 3) and m_1 means the first salesman visits the number of cities, m_2 means the second salesman visits the number of cities and m_3 means that the third salesman visits the number of cities. The cost matrix was generated randomly in the interval [0,100]. For each type of instance we considered six trials. Our algorithms have been implemented in C. The computational experiments were performed on a personal computer with AMD Sempron™ Processor LE-1200, 2.10 GHz, 896 RAM and OS Windows XP Professional. In table-1 we have presented the computational results for solving the problem using the Lexi-Search algorithm based on the Pattern Recognition Technique.

VII. ACKNOWLEDGMENT

The authors are very much thankful to the referees for their suggestions & useful comments.

VIII. REFERENCES

[1] A.E.carter and C.T.Ragsdale, (2006): A new approach to solving the multiple traveling salesmen problem using genetic algorithms, European Journal of Operations Research, Vol.175 (1), PP.246-257.

[2] Laporte, G and Y. Nobert (1983) : A cutting planes algorithms for the M-Salesmen Problem, J.Opns. Res. Vol. 31, No. 11, P. 1107.

[3] Pandit, S.N.N., and Sundara Murthy, M., "*Restriced TSP through n sets of nodes"*,paper presented at the 9th Annual Convention of ORSI, Calcutta, 1975.

[4] Pandit, S.N.N., "*The Loading Problem",* Opns. Res., 10, 1962, pp.639-646.

[5] Pandit, S.N.N., "*Some observations on the Longest Path Problems",* Opns. Res., Vol. 11., 1964, 361.

[6] Svestka, J.A (1976) : Response to - A Note on the formulation of the M-TSP. Mgt. Sci., Vol. 22, P.704.

[7] Sundara Murthy, M (1979) : Combinatorial Programming - A Pattern Recognition Approach, Ph.D. Thesis, REC, Warangal, India.

[8] Sundara Murthy, M and Bhavani, V (2006): Truncated M-TSP, Opsearch, Vol.43 No.2, pp. 152 – 177.

[9] Svestka, J.A and Huckfeldt, V.E. (1973): Computational Experience with an M-TSP algorithm Mgt. Sci., 19, No. 7, pp. 790 – 799.

[10] .Shiela Das and Borah P.C (1993): A Lexi Co graphic Search Approach for the Constrained M-TSP, A Paper Presented in the XXVI - Annual Convention of ORSI, Bhubaneswar, Orissa, India.

[11] Rao, M.R. (1980): "A note on the M-TSP" Opens, Res. Vol. 28, No.3.

AUTHORS

Mr.K.SOBHAN BABU is presently working as a Assistant Professor in the Department of Mathematics, University College of Engineering, JNTUK, KAKINADA, Andhra Pradesh, INDIA.

Mrs.K.CHANDRA KALA is presently working as a Assistant System Engineer in Tata Consultancy Services, Hyderabad, Andhra Pradesh, INDIA.

Dr.B.Naganna is currently working as a Associate Professor in the Department of Mathematics, Sri Vardhaman College of Engineering, Hyderabad. Andhra Pradesh, INDIA.

Dr.M.SUNDARA MURTHY is a Senior Professor in the Department of Mathematics, Sri Venkateswara University, Tirupati, Chittor (Dt) Andhra Pradesh, India. Research Area is Combinatorial Optimization & Algorithm Design, etc.