ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Travelling Salesman Problem with a grouping Constraint – A Lexi-Search Approach

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

Dr. Keshava Reddi.E
Associate Professor in Mathematics
College of Engineering, J N T U Anantapur
Anantapur, A.P., India
keshava_e@gmail.com

Prof.Sundara Murthy.M
Senior Professor in Mathematics
Sri Venkateswara University, Tirupati
Chittor, A.P., India
profmurthy@gmail.com

Abstract: There are many algorithms for usual one man TSP developed by researchers from time to time. But the problem has not received much attention in its required context. In this paper we study a problem called TSP with Variant Constraint. Lexi-Search is by far the mostly used tool for solving large scale NP-hard Combinatorial Optimization problems. Lexi-Search is, however, an algorithm paradigm, which has to be filled out for each specific problem type, and numerous choices for each of the components exist. Even then, principles for the design of efficient Lexi-Search algorithms have emerged over the years. 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 problem and find better solution.

Keywords: Travelling Salesman Problem, Tour, Lexi-Search, Word, Pattern

I. INTRODUCTION

The Travelling Salesman Problem (TSP) is a classical problem of combinatorial optimization of operations research area. The purpose is to find a shortest tour through a given no. of locations such that every location is visited exactly once. The cost of travelling from location i to location j is denoted by C_{ij}. These costs are symmetric if $C_{ij} = C_{ji}$ for each of pair of cities i and j, and asymmetric otherwise. There are several practical uses for this problem, such as vehicle routing with the additional constraints of vehicle's route, such as capacity of vehicles [1], drilling problems [2], minimize waste [3], clustering data arrays [4], X-ray crystallography [5], shot sequence generation for scan lithography [6] and many others.

This problem has also been used during the last years as comparison basis for improving several optimizations techniques, such as genetic algorithms [7], simulated annealing [8], tabu search [9], local search [10], ant colony [11] and Branch and Bound (B&B). The principal types of B&B used to solve the TSP are: The best known exact algorithms are based on either the B&B method for the Asymmetric TSP (ATSP) [12] or the Branch and Cut (B&C) method for the Symmetric TSP (STSP) using the double index formulation of the problem [13]. Currently, most algorithms for the TSP ignore high cost arcs or edges and save the low cost ones. A drawback of this strategy is that costs of arcs and edges are not accurate indicators whether those arcs or edges are saved in an optimal TSP solution.

"There are n cities and N= {1, 2, 3,... ,n}. The cost array C (i, j, k) is the cost of a salesman visiting from city i to city j at time (availing facility) k is known (i, j=1, 2, 3,…, n; k=1, 2, 3,…..m)".Here the third dimension need not be the usual time which is continuous, but a factor which influences the cost C and can be a facility.

A variant of well – known Traveling Salesman Problem where a tour does not necessarily visit all cities is called the Generalized Traveling Salesman Problem. More specifically, the set of 'n' cities are divided into r sets such that the N = $\{N_1, N_2, \cdots N_r\}$ and $N_i \cap N_j = \phi$. A subset with m < n cities has to be traveled by the salesman. The number of cities travelled by as salesman is m. The Travelling Salesman has to visit n_p cities in the N_p sets. The problem is to find a minimum cost tour by visiting 'm' cities with given number of n_p cities, where

$$\sum_{p=1}^{r} n_p \leq m$$

In the sequel we developed a lexi-search algorithm based on the "Pattern Recognition Technique" to solve this problem which takes care of the simple combinatorial structure of the problem. In Section 2, a Lexi-Search method was developed for the TSP with a variant constraint and mathematical formulation is shown in Section 3. The algorithm is presented in Section 4. The computational results are provided in Section 5 and the concluding remarks are given in Section 6.

II. AN 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*** [14] 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 [14] and hence reduces computational work in getting an optimal solution.

III. MATHEMATICAL FORMULATION

$$Minimize\ Z = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} C(i,j,k)\, X(i,j,k) \qquad \text{------------------(1)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X(i,j,k) = 1, \qquad k = 1, 2 \ldots m \qquad \text{---------------- (2)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} X(i,j,k) = m, \qquad \text{--------------- (3)}$$

$$X(i,j,k) = 0 \text{ or } 1 \qquad \text{------------- (4)}$$

Constraints (2) & (3) and the restrictions are

$$\sum_{j=1}^{n}\sum_{k=1}^{m} X(i,j,k) = \xi_i$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} X(i,j,k) = \xi_i, \xi_i = 1, i \in N_p \ \&\ \xi_i = 0, otherwise$$

and

$$\sum_{i \in N_p} \xi_i = n_p,$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} X(i,j,k) = \xi_j, \xi_j = 1, j \in N_p \ \&\ \xi_j = 0, otherwise$$

and

$\sum_{j \in N_p} \xi_j = n_p$, define the constraint set of the generalized TSP, whose objective function (1) is minimum. Equation (4) represents that the salesman visits the city j from city i at time or facility k is 1 otherwise 0. X is a feasible tour is it satisfies all the constraints and restrictions

IV. THE 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*** [14] 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 [14]&[15] and hence reduces computational work in getting an optimal solution. The concepts and notations involved in the Lexi-Search are briefly described below.

A. Pattern Recognition Technique:

The search efficiency of a Lexi-Search algorithm is based on the choice of an appropriate Alphabet-Table. In this case two conflicting characteristics of the search list have to be taken into account: one is the difficulty in setting bounds to the values of the partial words (that defines partial solutions representing subsets of solutions). The other difficulty is in checking the feasibility of a partial word. Thus two cases arises in the choice of Alphabet Table [14]

(i). The process of checking the feasibility of a partial word is easy, while the calculations of a lower bound is bulky and (ii). Computation of lower bound is easy, while the feasibility checking is difficult.

When the process of feasibility checking of a partial word becomes difficult and the lower bound computation is easy, a modified Lexi-Search i.e. Lexi-Search with recognizing the Pattern of the solution known as Pattern Recognition Technique [14] can be adopted. In this method, in order to improve the efficiency of the algorithm, first the bounds are calculated and then the partial word, for which the value is less than the initial (trial) value are checked for the feasibility.

B. 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).

C. Alphabet Table & Word:

Let L_k = $(a_1,a_2. . .a_k)$. $a_i ε$ 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.The value of the (partial) word L_k, $V(L_k)$ is recursively defined as $V(L_k) = V(L_{k-1}) + D(a_k)$ with $V(L_0) = 0$, where D is the cost array arranged such that, $D(a_k) < D(a_{k+1})$. $V(L_k)$ and the value of the pattern X, will be the same, since X is the (partial) pattern represented by L_k.

D. Lowerbound of a Partial Word:

A lower bound LB (L_k), for the values of the blocks of words represented by $L_k = (a_1, a_2. . . a_k)$ can be defined as follows:

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

E. Feasibility criterion of a Partial Word

A recursive algorithm is developed for checking the feasibility of a partial word $L_{K+1} = (a_1, a_2, \ldots, a_k, a_{k+1})$ given that L_K is a feasible partial word. We will introduce some more notations which will be useful in the sequel.

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

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

IT be an array where IT (i) =1, $i \in N$ represents that the salesman at time (facility) 'i' travels one pair of cities.

SW be an array where SW (i) is the city that the salesman is visiting from city i,Sw(i) = 0 indicates that the salesman is not visiting any city from city i.

L be an array where L (i) is the letter in the ith position of a word.

NL be an array where NL (i) =q indicates that q cities are covered up to i^{th} Position of a Word.

GS be an array where GS (i) represents i^{th} group number of cities.

Then for a given partial word $L_K = (a_1, a_2, \ldots, a_K)$ the values of the arrays RI, CI, TI, SW, L ,NL and GS as follows.

IR(R (a_i)) =1, i=1, 2, 3……..K
IC(C (a_i)) =1, i=1, 2, 3……..K
IT (T (a_i)) =1, i=1, 2, 3……. K
SW(R (a_i)) =C (a_i), i=1, 2, 3……..K
L (i) =a_i, i=1, 2, 3……..K

NL(i)=NL(i-1)+1, if IC(R(a_i))=0 and NL(i)=NL(i-1)+1, if IR(C(a_i))=0 i=1,2,3….K

GS (GN(R (a_i)) =GS (GN(R (a_i)) +1 if IC(R (a_i)) =0 and
GS (GN(C (a_i)) =GS (GN(C (a_i)) +1 if IR(C (a_i)) =0 i=1,2,3….K

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 TT=T (a_{P+1}).

Algorithm – 1:

STEP1: IX=0
NXA=NL [I-1]
TCX=TC
GOTO 2

STEP 2: IS (IR (TR) = 1) IF YES GOTO 10
IF NO GOTO 2A

STEP2A:IS (IC (TC) = 1) IF YES GOTO 10
IF NO GOTO 2B

STEP2B:IS (IT (TT) = 1) IF YES GOTO 10
IF NO GOTO 3

STEP3: IS IC (TR) =0 IF YES NXA=NXA+1 GOTO 3A
IF NO GOTO 3A

STEP3A:IS IR (TC) =0 IF YES NXA=NXA+1 GOTO 3B
IF NO GOTO 3B

STEP3B:IS (NXA>M) IF YES GOTO 10
IF NO GOTO 4

STEP4: IS IC (TR) =0 IF YES GS (GN (TR)) = GS (GN (TR)) +1 GOTO 4A
IF NO GOTO 4A

STEP4A:IS IR (TC) =0 IF YES GS (GN (TC)) = GS (GN (TC)) +1 GOTO 4B
IF NO GOTO 4B

STEP4B:IS GS (GN (TR)) <= GP (TR)
IF YES GNZ1=0 GOTO 4C
IF NO GNZ1=1 GOTO 4C

STEP4C:IS GS (GN (TC)) <= GP (TC)
IF YES GNZ2=0 GOTO 4D
IF NO GNZ2=1 GOTO 4D

STEP4D:NPA=NP-(GNZ1+GNZ2) GOTO 5

STEP5: IS M – NXA <= NPA IF YES GOTO 6
IF NO GOTO 7

STEP6: IS (SW (TCX) = 0) IF YES IX=1 GOTO 10
IF NO IK=SW (TCX) GOTO 6A

STEP6A:IS (IK=TR) IF YES GOTO 6B
IF NO TCX=IK GOTO 6

STEP6B:IS (I=M) IF YES IX=1 GOTO 10
IF NO GOTO 10

STEP7: IS IC (TR) =0 IF YES GS (GN (TR)) = GS (GN (TR)) – 1 GOTO 7A
IF NO GOTO 7A

STEP7A:IF IR (TC) =0 IF YES GS (GN (TC)) = GS (GN (TC)) – 1 GOTO 10
IF NO GOTO 10

STEP10: STOP.

This recursive algorithm will be used as a subroutine in the lexi-search algorithm. We start the algorithm with a very large value, say, 9999 as a trial value of VT. If the value of a feasible word is known, we can as well start with that value as VT. During the search the value of VT is improved. At the end of the search the current value of VT gives the optimal feasible word. We start with the partial word L_1= (a_1) = (1). A partial word $L_p=L_{p-1}*(a_p)$ where $*$ indicates chain form or concatenation. We will calculate the values of V (L_p) and LB (L_p) simultaneously. Then two cases arises (one for branching and other for continuing the search).

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 Lp is not feasible then consider the next partial word of order p by taking another letter which succeeds a_p in the p^{th} position. If all the words of order p are exhausted then we consider the next partial word of order (p-1).

LB (L_P) $\geq$ VT. In this case we reject the partial word meaning that the block of words with L_p as leader is rejected for not having an optimal word and we also reject all partial words of order p that succeeds L_p.

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

ALGORITHM 2: (LEXI-SEARCH ALGORITHM)

The following algorithm gives an optimal feasible word.

STEP 1 : (Initialization)
The arrays SN, D, DC, R, C, T and values of N, M, GN, GP are made available IR, IC, IT, SW, L, NL, GS, V, LB are initialized to zero. The values I=1, J=0, VT=9999, NZ=N$*$N$*$M –N, MAX=NZ-1

STEP 2: J=J+1
IS (J>MAX) IF YES GOTO 11
IF NO GOTO 3

STEP 3: L (I) = J
JA = J + M - I
IS (I = 1)
IF YES NXA = 0, V (I) = D (J) GOTO 3B
IF NO NXA=NL (I-1) GOTO 3A

STEP 3A: V (I) = V (I -1) + D (J)
GOTO 3B

STEP 3B: LB (I) = V (I) + DC (JA) – DC (J)
GOTO 4

STEP 4:IS (LB (I) $\geq$ VT) IF YES GOTO 11
IF NO GOTO 5

STEP 5: TR=R (J)
TC=C (J)
TT=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=M) IF YES GOTO 10
IF NO GOTO 8

STEP 8 : L (I) = J
IR (TR) = 1
IC (TC) = 1
IT (TT) = 1
SW (TR) = TC
NL (I) = NXA
GOTO 9

STEP 9 : I=I+1
MAX=MAX+1
GOTO 2

STEP10 : L (I) =J L (I) IS FULL LENGTH WORD AND IS FEASIBLE.
VT=V (I), record L (I), VT,

GOTO 13

STEP11 : IS (I=1) IF YES GOTO 14
IF NO GOTO 12

STEP12 : I=I-1
MAX=MAX+1
GO TO 13

STEP13 : J=L (I)
NL (I) = 0
TR = R (J)
TC = C (J)
TT = T (J)
IR (TR) = 0
IC (TC) = 0
IT (TT) = 0
SW (TR) = 0
GS (GN (TR)) = GS (GN (TR)) -1
GS (GN (TC)) = GS (GN (TC)) -1
GOTO 2

STEP14 : STOP END

V. 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 gives the list of the problems tried along with the average CPU time in seconds required for solving them.

Table I. (CPU run time in seconds for Lexi-Search Algorithm)

Sr. No.	***Problem Dimensions***		***AT***	***Type-I***			***Type-II***			***Type-III***		
	n	m		Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
1	10	5	0.010	0.021	0.040	0.035	0.026	0.004	0.003	0.002	0.004	0.003
2	20	12	0.0027	0.02	0.005	0.0035	0.004	0.006	0.050	0.002	0.005	0.0035
3	25	16	0.0037	0.002	0.004	0.003	0.002	0.004	0.003	0.002	0.004	0.003
4	35	24	0.0036	0.002	0.004	0.003	0.002	0.004	0.003	0.002	0.004	0.003
5	40	28	0.0038	0.002	0.005	0.0035	0.004	0.006	0.005	0.002	0.005	0.0035

VI. CONCLUSIONS

The problems are solved by using Lexi-search algorithm based on the pattern recognition technique. In the above table m represents that the salesman has visited the number of cities. The cost matrix was generated randomly in the interval [0,100]. For each type of instance we considered six trails. Our algorithm has been implemented in C program. The computational experiments were performed on a personal computer with AMD Sempron^{TM} Processor LE-1200, 2.10GHz, 896RAM and OS Windows XP Profesional. In the table-1 we have presented the computational results for solving the problem using the Lexi-Search algorithm based on the Pattern Recognition Technique.

VII. REFERENCES

[1]. Laporte, G.," The vehicle rotting problem: an overview of exact and approximate algorithms", Eur. J. Oper. Res. 59 (2), (1992) 345–358.

[2]. Onwubolu, G.C. and Clerc, M., " Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization", Int.J. Prod. Res. 42 (3), (2004) 473–491.

[3]. Grafinkel, R. S "Minimizing wallpaper waste, part I: a class of Traveling Salesman Problems", Oper Res 25, (1977) 741- 751.

[4]. McCormick, W. T., Schweitaer, P. J. and White, T. W., "Problem decomposition and data reorganization by a Clustering technique", Oper Res 20, (1972) 993-1009.

[5]. Bland, R. G.and Shallcross, D.F., "Large Traveling Salesman Problem arising experiment in x-ray Crystallography: A Preliminary reported on computation", Oper Res 8, (1989) 125-128.

[6]. Shinano, Y.,Inui, N., Fukagawa, Y. and Takakura, N., "An Application of Traveling- Salesman Models to Shot Sequence Generation for Scan Lithography", 5th European Congress on Computational Methods in Applied Sciences and Engineering, (2008) 30 –Julys 5, Venice, Italy.

[7]. Affenzeller, M., Wanger, S., "A self-adaptive model for selective pressure handling within the theory of genetic algorithms", EUROCAST 2003, Las Palmas de Gran Canaria, Spain, Lect. Notes Comp. Sci. 2809 (1),(2003) 384–393.

[8]. Budinich, M., 1996," A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing", Neural Comput. 8, pp. 416–424.

[9]. Liu, G., He, Y., Fang, Y., Oiu, Y.," A novel adaptive search strategy of intensification and diversification in tabu search", in: Proceedings of Neural Networks and Signal Processing, Nanjing, China.(2003)

[10]. Bianchi, L., Knowles, J., Bowler, J., " Local search for the probabilistic traveling salesman problem: Correction to the 2- p-opt and 1-shift algorithms", Eur. J. of Oper. Res. 162(1), (2005) 206–219.

[11]. Chu, S.C., Roddick, J.F., Pan, J.S.," Ant colony system with communication strategies", Inform. Sci. 167 (1–4), (2004), 63–76.

[12]. Fischetti, M., Lodi, A., Toth, P., Exact Methods for the Asymmetric Traveling Salesman Problem. In: Gutin, G., Punnen, A.P. (Eds.), The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, (2002). 169–194 (Chapter 9).

[13]. Naddef, D., 2002, Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP.In: Gutin, G., Punnen, A.P. (Eds.), The Traveling Salesman Problem and its Variations. Kluwer Dordrecht, (1991) 29–116 (Chapter 2).

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

[15]. Sobhan Babu,K & Sundara Murthy.M, "An efficient algorithm for Variant Bulk Transportation Problem", International Journal of Engineering Science and Technology, 2010, Vol.2(7), pp.2595-2600.

AUTHORS PROFILE

Dr.K.SOBHAN BABU is presently working a Assistant Professor in the department of Mathematics, UCEK, JNTUK and Additional Controller of Examinations in JNTUKAKINADA.

Dr.E.KESHAVA REDDI is presently working a Associate Professor in the department of Mathematics, College of Engineering, Anantapur and Controller of Examinations in JNTU Anantapur. His Research area includes Optimization, Data Mining etc.

Dr.M.SUNDARA MURTHY is a senior Professor in the Department of Mathematics, Sri Venkateswara University, Tirupati. He has so many publications in National and International reputed Journals.