A LEXI SEARCH APPROACH TO CONSTRAINT ASSIGNMENT PROBLEM

Dr. U.BALAKRISHNA
Professor of Mathematics, Department of Science & Humanities,
Sreenivasa Institute of Technology and Management Studies, Chittoor

It is a two dimensional problem where the time matrix M(i, j) is the time of the jth job is assigned to ith person. The time matrix M(i. j)[i=1,2,3,…,m; j=1,2,3,…,n] is known. There are n jobs and out of the given n jobs only n^1(<n) are to be assigned to m(<n^1) persons, and each of the person is constrained to do the specified number of jobs. All the persons start working on the jobs simultaneously but a person cannot work on more than one job at a time. The problem is to assign the n^1 jobs to m persons, with minimum total time with the restriction that the each person to do given specified number of jobs. A Lexi search approach is proposed using pattern recognition technique to find an optimal feasible assignment. For this problem a computer program is developed for the algorithm and is tested. It is observed that it takes less time for solving higher dimension problems also.

Keywords: Constraint assignment problem, Lexi search algorithm, Pattern recognition technique, Pattern, Alphabet

Introduction:

Assignment problems deals with the question how to assign n objects to m other objects in an injective fashion in the best possible way. An assignment problem is completely specified by its two components the assignments, which represent the underlying combinatorial structure, and the objective function to be optimized, which models "the best possible way". The assignment problem refers to another special class of linear programming problem where the objective is to assign a number of resources to an equal number of activities on a one to one basis so as to minimize total costs of performing the tasks at hand or maximize total profit of allocation. In other words, the problems is, how should the assignment be made so as to optimize the given objective. Different methods have been presented for assignment problem and various articles have been published on the see [1], [2] , [3] and [4] for the history of these methods. But in all the above attempts the simple combinatorial structure of the assignment problem is not at all taken into consideration.

Few Application of the Assignment Method: 1. Assign sales people to sales territories. 2. Assign Vehicles to routes. 3. Assign accountants to client accounts. 4. Assign contracts to bidders through systematic evaluation of bids from competing suppliers. 5. Assign naval vessels to petrol sectors. 6. Assign development engineers to several construction sites. 7. Schedule teachers to classes etc. 8.Men are matched to machines according to pieces produced per hour by each individual on each machine. 9. Teams are matched to project by the expected cost of each team to accomplish each project.

Here we study the following “CONSTRAINT ASSIGNMENT PROBLEM” :

There are I={1,2,3,…..,m} set of m persons, J={1,2,3,…..,n} set of n jobs and T(i,j) is the time that ith person take to complete jth job. Out of the given n jobs only n^1(<n) are to be assigned to m persons. It is given that 1)n^1 > m 2) ith person has to do m_i jobs such that $\sum m_i = n^1$ 3) A job can not be assigned to more than one person 4) All the persons start working on the jobs simultaneously 5) The cumulative times of the m_i jobs assigned to ith person is the complete time of the ith person 6) The completion time of the n jobs is the maximum among m persons complete times. The problem is to assign n^1 jobs to m persons with minimum total time with the above restrictions. In the sequel we will develop 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.

Numerical illustration:

The concepts and the algorithm developed will be illustrated by a numerical example for which n=6, n^1 = 4,m=2,m_1=2, m_2=2 . The Time array M(i, j) is given table-1:

TABLE-1

$$M(i,j)=\begin{bmatrix}3 & 3 & 4 & 3 & 4 & 3\\ 5 & 2 & 3 & 4 & 5 & 4\end{bmatrix}$$

A Feasible Assignment of the jobs can be represented by an appropriate mxn indicator array X=[X(i,j), X(i,j)=0 or 1 in which X(i,j)=1 indicates that the jth job is assigned to ith person and if there is no such assignment it is indicated by X(i,j)=0].'X' is called a 'solution'. The indicator X given in Table-2 where 2x6 of X is represented a solution to the numerical example and is represented as follows:

TABLE-2

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

The representation of the solution X to the problem is that 1st person does the jobs {1,5} , 2nd person does the jobs {2,4}. This solution is a feasible solution; the completion time is 3+4+2+4=13.
The assignment corresponding to Table-2 can be represented as [(1,1),(1,5),(2,2),(2,4)].

Concepts and Definitions:

Definition of a pattern:

An indicator two-dimensional array which is associated with an assignment is called a 'pattern'. A Pattern is said to be feasible if X is a solution. The pattern represented in the table-2 is a feasible pattern. Now T(X) the value of the pattern X is defined as T(X) = $\sum\sum T(i,j)X(i,j)$. The value T(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 doubles [(i,j)] for which X(i,j)=1, with understanding that the other X(i,j)'s are zeros.

There are M=mxn ordered doubles in the two-dimensional array X. For convenience these are arranged in ascending order of their corresponding times and are indexed from 1 to M (Sundara Murthy-1979). Let SNo= [1, 2, 3,….., M] be the set of M indicies. Let DT be the corresponding array of times. If a, b$\in$SNo and a<b then DT(a)$\leq$DT(b). Also let the arrays R, C be the array of row, column indices of the ordere
d doubles represented by SNo and TC be the array of cumulative sum of the elements of DT. The arrays SNo, DT, TC, R, C for the numerical example are given in the table-3. If p$\in$SNo then (R(p),C(p)) is the ordered triple and DT(a)=T(R(a),C(a)) is the value of the ordered triple and TC(a)= $\sum_{i=1}^{a} TD(i)$

TABLE- 3
ALPHABET TABLE

SN	DT	TC	R	C
1	2	2	2	2
2	3	5	1	1
3	3	8	1	2
4	3	11	1	4

5	3	14	1	6
6	3	17	2	3
7	4	21	1	3
8	4	25	1	5
9	4	29	2	4
10	4	33	2	6
11	5	38	2	1
12	5	43	2	5

Let us consider $12 \in SN$. It represents the ordered double (R(12),C(12))=(2,5).Then DT(12)=T(2,5)=5 and TC(12)=43

Definition of an Alphabet-Table and a word:

Let $L_K = \{a_1,a_2,\ldots\ldots,a_K\}$, $a_i \in SN$ be an ordered sequence of k indicies from SN. The pattern represented by the ordered doubles whose indicies are given by L_k is independent of the order of a_i in the sequence. Hence for uniqueness the indicies are arranged in the increasing order such that $a_i<a_{i+1}$, i=1, 2… n-1. The set SN is defined as the "Alphabet-Table" with alphabetic order as (1, 2, 3…M) 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, 2,3…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. Therefore a partial feasible word is said to be feasible if $k=n^1$.

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. Consider $L_{K-1} = (a_1,a_2,\ldots\ldots\ldots,a_{k-1})$. The alphabet table for the kth position is $SNa_{k-1}=(a_{k-1}+1,a_{k-1}+2\ldots M)$, where SN_P is defined as $SN_P=(p+1,p+2,\ldots\ldots,M)$. Thus for example consider a word with two letters as $(a_1,a_2)=(1,3)$. Then $SNa_2=SN_3=(4, 5, 6,\ldots,12)$ is the alphabet for the third position. We concentrate on the set of words of length n(for the numerical example it is 4). A leader L_k $(k<n^1)$ is said to be feasible, if the block of words defined by it contains at least one feasible word or equivalently there should not be inconsistency in the partial pattern defined by the partial word.

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.

IC	be an array where IC (i) =1, $i \in I$ represents that the ith job assigned to a person, otherwise zero.
L	be an array where L (i) is the letter in the ith position of a word
XL	be an array where XL(i) indicates that the person assigned to ith job
ALV	be an array where ALV(i) = p indicates that p is the sum of the times corresponding to 1st person
BLV	be an array where BLV(i) = q indicates that q is the sum of the times corresponding to 2nd person
NL	be an array where NL (i) =s indicates that s jobs are covered by the persons

Then for a given partial word $L_K = (a_1,a_2,\ldots..,a_K)$ the values of the arrays IC, L,XL,NL,ALV,BLV are as follows.

IC(C (a_i)) =1, i=1, 2, 3…….K
L (i) =a_i, i=1, 2, 3…….K
XL(C (a_i)) =R (a_i) i=1, 2, 3…….K
NL(R (a_i)) =NL(R (a_i)) +1 i=1, 2, 3…….K
ALV (i) =ALV(i-1) +DT (a_i) i=1, 2, 3…….K
BLV (i) =BLV (i-1) +DT (a_i) i=1, 2, 3…….K

For example consider a sensible partial word L_5 = (1, 2, 4) which is feasible. The array IC, L, XL, NL, ALV, BLV takes the values represented in the table-4 given below.

TABLE-4

	1	2	3	4	5	6
IC	1	1	0	1	0	0
L	1	2	4	-	-	-
XL	1	2	-	1	-	-
NL	2	1	-	-	-	-
ALV	0	3	6	-	-	-
BLV	2	2	2	-	-	-

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 RT=R (a_{p+1}), CT=C (a_{P+1})

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 can be defined as follows:

IF TR=1

$$LB(L_K) = MAX \begin{Bmatrix} lVA = LVA(i) + CT[a_k + M1 - (LN(TR)] - CT(a_k), \\ lVB = LVB(i) + CT[a_k + M2 - (LN(TR+1)] - CT(a_k) \end{Bmatrix}$$

IF TR=2

$$\mathbf{LB(L_K) = MAX} \begin{Bmatrix} lVA = LVA(i) + CT[a_k + M1 - (LN(TR-1)] - CT(a_k), \\ lVB = LVB(i) + CT[a_k + M2 - LN(TR)] - CT(a_k) \end{Bmatrix}$$

ALGORITHM: 1

STEP1 : IX=0
STEP2 : IS (IC (CT) = 1) IF YES GOTO 10
IF NO GOTO 3
STEP3 : IS (RC (RT, CT) =0) IF YES GOTO 4
IF NO GOTO 10
STEP4 : IS RT=1 IF YES GOTO 5
IF NO GOTO 6
STEP5 : NL (RT) =NL (RT) +1
GOTO 5A
STEP5A : IS NL (RT) ≤ M1 IF YES GOTO 5E

IF NO GOTO 5A1

STEP5A1 : NL (RT) =NL (RT)-1
GOTO 10

STEP5E : lVA=ALV (I-1) +TC [J+M1-NL (TR)]-TC (J)
lVB=BLV (I-1) +TC [J+M2-(NL (TR+1)]-TC (J)
GOTO 5F

STEP5F : IS lVB<lVA IF YES LB=lVA GOTO 5G
IF NO LB=lVB GOTO 5G

STEP5G : IS LB<VT IF YES IX=1 GOTO 10
IF NO GOTO 10

STEP6 : NL (RT) =NL (RT) +1
GOTO 6A

STEP6A : IS NL (RT) ≤ M2 IF YES GOTO 6E
IF NO GOTO 6A1

STEP6A1 : NL (RT) =NL (RT)-1
GOTO 10

STEP6E : lVB=LVB(I-1)+CT[J+M2-LN(TR)]-CT(J)
lVA=LVA(I-1)+CT[J+M1-(LN(TR-1)]-CT(J)
GOTO 6F

STEP6F : IS lVB<lVA IF YES LB=lVA GOTO 6G
IF NO LB=lVB GOTO 6G

STEP6G : IS LB<VT IF YES IX=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 value of LB (L_p). Then two cases 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 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).

2. LB (L_P) ≥ 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.

4.5 ALGORITHM-2: (LEXI-SEARCH ALGORITHM)

The following algorithm gives an optimal feasible word

STEP 1 : (Initialization)
The arrays SNo, DT, TC , R, C and values of N, N^1, M, M1, M2 are made available IR, IC,L, XL, NL,ALV,BLV are initialized to zero. The values I=1, J=0, VT=9999, NZ=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
GOTO 4
STEP 4 : RT=R (J)
CT=C (J)
DTV=TD (J)
GOTO 5
STEP 5 : CHECK THE FEASIBILITY OF L (USING ALGORITHM-1)
IS (IX=0) IF YES GOTO 2
IF NO GOTO 6
STEP 6 : IS ($I=N^1$) IF YES GOTO 10
IF NO GOTO 7
STEP 7 : L (I) = J
CI (CT) = 1
GOTO 7A
STEP 7A : IS (RT=1) ALV (I) =ALV (I) +DTV
ELSE BLV (I) =BLV (I) +DTV
GOTO 8
STEP 8 : I=I+1
MAX=MAX+1
GOTO 2
STEP10 : L (I) =J
L (I) IS FULL LENGTH WORD AND IS FEASIBLE.
VT=LB, record L (I), VT
GOTO 13
STEP11 : IS (I=1) IF YES GOTO 15
IF NO GOTO 12
STEP12 : I=I-1
MAX=MAX+1
GO TO 13
STEP13 : J=L (I)
RT = R (J)
CT = C (J)
DTV=DT (J)
IC (CT) = 0
NL (RT) =NL (RT)-1
GOTO 14
STEP 14 : IS (RT=1) ALV (I) =ALV (I)-DTV
ELSE BLV (I) =BLV (I)-DTV
GOTO 2
STEP15 : STOP
END

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

4.6 Search Table:

The working details of getting an optimal word, using the above algorithm for the illustrative numerical example is given in the Table-5. The columns (1),(2),(3),(4) gives the letters in the first, second, third, fourth

places respectively. The rows R, C gives the row, column indices of the letter. The corresponding ALV (i), BLV(i) and LB (i) are indicated in the last columns. The last column gives the remarks regarding the acceptability of the partial words. In the following table A indicates ACCEPT and R indicates REJECT.

TABLE-5
SEARCH TABLE

SN0	1	2	3	4	R	C	ALV	BLV	LB	REM
1	1				2	2	0	2	6	A
2		2			1	1	3	2	6	A
3			3		1	2				R
4			4		1	4	6	2	6	A
5				5	1	6				R
6				6	2	3	6	5	6	A=VT=6
7			5		1	6	6	2	6	R=VT
8		3			1	2			6	R=VT
9	2				1	1	3	0	6	R=VT

At the end of the search the current value of VT is 6 and it is the value of the optimal feasible word L_4=(1,2,4,6).The array IC,L,XL,NL, ALV,BLV takes the values represented in the table-6 given below. It is given in the 6th row of the search table. The pattern represented by the above optimal feasible word is represented in the following table-7.

TABLE-6

	1	2	3	4	5	6
IC	1	1	1	1	0	0
L	1	2	4	6	-	-
NL	2	2	-	-	-	-
ALV	0	3	6	6	-	-
BLV	2	2	2	5	-	-

TABLE-7

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

The assignment represented by the above pattern is [(2,2), (1,1), (1,4), (2,3)] where 2nd person does job 2 , 1st person does job 1 , 1st person does job 4, 2nd person does job 3 i.e., 1st person does jobs {1,4} and 2nd person does jobs {2,3}. This solution is a feasible solution with completion time 11.

The Assignment is:

Person	1	1	2	2
Job assignment	1	4	2	3

4.7 Computational Experience:

A Computer program for the above algorithm is written in C language. Random numbers are used to construct the Time matrix. The following table-8 gives the list of the problems tried along with the average CPU time in seconds required for solving them.

In the table TA represents the CPU time to construct the alphabet-table and TE represents the CPU time taken for the search of a feasible word. The time is represented in seconds. In the table-8 ‘n’ is the number of jobs, m is the number of persons, n^1 is the number of truncated jobs, m_1 is the number of jobs

done by one person and m_2 is the number of jobs done by other person and p_1 represents not to utilize the facilities more than a given number of p_1 times.

Experiments are carried out and by generating the three different classes of random data sets, where the three types of data sets are defined as follows:

Type 1: C (i, j) are uniformly random in [1,100]
Type 2: a) C (i, j) are uniformly random in [1,100]
b) VT=0.85VT
Type 3: a) C (i, j) are uniformly random in [1,100]
b) Max=(nxm)/3

And the results are tabulated in Table. For each type, three data sets are tested. It is seen that time required for the search (ET) of the optimal solution is fairly less.

TABLE-8

Problem dimensions					No. of prob's	TA	Total time taken(TE)								
							TYPE 1			TYPE 2			TYPE 3		
n	m	n^1	m_1	m_2			min	max	avg	min	max	Avg	min	max	avg
10	2	8	5	3	3	0.27	5.23	5.93	5.58	5.08	5.58	5.25	4.15	4.75	4.41
42	2	10	16	14	3	0.46	7.05	7.40	7.19	7.02	7.25	7.15	6.19	6.37	6.29
115	2	12	47	35	3	0.49	11.3	11.8	11.5	11.2	11.7	11.3	9.12	9.40	9.25

In the above table it can be notice that the average CPU times for Type 1, Type 2 and Type 3 are in decreasing order since in Type 2 the search is made around 0.85VT and in Type 3 the search is using 1/3 of the alphabet table. But in all the cases we are getting the same optimal solution as a coincidence.

REFERENCES:

[1] M.S. Bazarra, John J. Jarvis, Hanif D. Sherali(2005) Linear programming and network flows.
[2] B.S. Goel, S.K. Mittal(1982) Operations Research, Fifty Ed., 2405-2416.
[3] Hamdy A.Taha(2007) Operations Research, an introduction, 8th Ed
[4] H.J. Zimmermann, Rudrajit Tapador(1996) Solving the assignment, third Ed. kluwer Academic, Boston
[5] Balakrishna, U and Sundara Murthy, M. (2009). Three dimensional TSP and ASP models, PhDThesis,SVUniversity,Tirupati.
[6] Sundara Murthy, M. (1979). Combinational Programming. A Pattern Recognition Approach – Ph.D. Thesis, REC, Warangal.
[7] Balakrishna, U(2017). A New Approach To Variant Assignment Problem, Journal on Mathematics,Vol 7(1),32-40