International Journal of Mathematics Research.
ISSN 0976-5840 Volume 7, Number 2 (2015), pp. 159-166

http://www.irphouse.com

Time Dependent Bulk Transportation Problem

Dr. B. Naganna and Dr. M. Sundara Murthy

Lecturer in Mathematics, Govt. Polytechnic, Adoni, Kurnool(Dist)
Andhra Pradesh-518301
Rtd. Professor in Mathematics, S. V. University, TIRUPATI-517 502 (A. P)

ABSTRACT

There is a set I = (1, 2,---, m) of m sources which produces a particular product, a set J = (1, 2,---, n) of n destinations which require this product and a set k = (1, 2,---, p) which is a time / facility. The requirement of the destination j ε J is DR (j) and the capacity of the source i ε I is SC (i). C(i, j, k) be the Time Dependent Bulk Transportation Cost from source 'i' to the destination 'j' at time / facilities k. There is a restriction that each destination in 'J' should get its requirement from any one source at any point of time / facility k, but the source can supply to any number of destinations subject to its capacity. The objective is to find the least transportation cost satisfying the requirements of all the destinations in 'J' at some point of time / facility 'k'.
An effective "Lexi Search" algorithm is developed using" Pattern Recognition Technique" and it is effective in solving for higher values of m, n and k. For this problem a computer programmed in 'C' language is developed for the algorithm and is tested.

Keywords: Time Dependent Bulk Transportation Problem, Lexi search Algorithm, Pattern Recognition Technique, Trip-Schedule, Alphabet Table, Word, and Search Table.

1. Introduction:

In this chapter we will study 'Time Dependent Bulk Transportation Problem" (TD Bulk TP) with the following characteristics.

(i) The sources are fixed and the capacity of each source is given
(ii) The requirements of the destinations are given
(iii) In C(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 'C'. 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.). Under this consideration Miller *etal.* (1960), perskalla (1966), Picard and Queyranne (1978), Bhavani and Sundara Murthy (2005), S. Das and Monisha Borthakur (2006) have studied a variety of problems.

(iv) A destination should get its requirement from any one source at some point of time / facility and a source can supply to any number of destinations subject to its capacity.

(v) The bulk-transportation cost from a source to a destination at some point of time / facility is given.

The objective is to supply the requirements of the destinations with a minimum cost satisfying the above conditions.

The above Bulk Transportation without time/facility can be thought of as a two dimension bulk transportation problem (with cost c (i, j)). This also can be formulated as a 0-1 programming problem and can be solved (Balas-1965 and Glover-1965). But none of these methods will take the advantage of the combinatorial structure of the problem which is very close to that of the 'assignment problem'. Demaio and Roveda-1971 first formulated this problem and they have developed a branch and bound algorithm to solve it. They also gave a practical situation where this problem arises. Later, Srinivasan and Thompson-1973, Vanita verma and M. C. Puri-1996 were developed a branch and bound algorithm for the same problem. They have formulated a modified transportation problem for which the optimal solution of the above problem will be a basic feasible solution. They believe that this algorithm is better than the one presented by Demaio and Roveda-1971 because their algorithm utilises the structure of the transportation problem. The main drawback of Demaio and Reveda's algorithm is a lot of calculations are required for checking the feasibility of a solution, and several solutions have to be recorded till the optimal solution is identified. The letter's algorithm ignores the simple structure of the problem, which is very close to that of assignment problem in finding an optimal solution. In this algorithm a series of transportation problems are formulated and solved. Integer solutions are expected at each stage.

In the present study, takes out the drawbacks of Demaio and Roveda algorithm and we developed an algorithm called the Lexi-Search algorithm which takes the advantage of the simple structure of the problem to get an optimal feasible solution ((a) Pandit and Sundara Murthy-1975 and (b) Sundara Murthy-1976) was presented.

We will introduce the following notations:

Let

$I = \{i = 1, 2, ----, m\}$ be the set of m sources

$J = \{j = 1, 2, ----, n\}$ be the set of n destinations

$T = \{k = 1, 2, ----, p\}$ be the set of p time / facilities

$SC = SC(i)$, $i \varepsilon S$ be an array, where SC (i) is the availability at source i

$DR = DR(j)$, $j \varepsilon J$ be an array where DR (j) is the requirement of the destination j

$C = C(i, j, k)$, $i \varepsilon I$, $j \varepsilon J$, $k \varepsilon T$, where C (i, j, k) be the bulk-transportation Cost of supplying from source i to destination j at any point of time / facility k.

The problem is to find a solution i.e. Trip-schedule with a least total transportation cost in which the destinations get their requirements with the given condition.
The time/facility constraint introduces the complexity. Hence the algorithm for solving the Bulk-Transportation problem of two dimensions is not directly useful. In the sequel a Lexi search algorithm is developed to solve this problem.

2. Mathematical Formulation:

$$\text{Minimize} \quad Z = \sum_{i\varepsilon I} \sum_{j\varepsilon J} \sum_{k\varepsilon K} C(i, j, k)\ X(i, j, k)$$

$$\text{Subject to} \quad \sum_{j\,\varepsilon\, J} \sum_{k\varepsilon K} DR(j)\ X(i, j, k) \leq SC(i), \quad \text{for } i \varepsilon I$$

$$\sum_{i\,\varepsilon\, I} \sum_{k\varepsilon K} X(i, j, k) = 1, \text{ for } j \varepsilon J$$

$$X(i, j, k) = 0 \text{ or } 1 \text{ for } i \varepsilon I, j \varepsilon J, k \varepsilon K$$

X (i, j, k) = 1, indicates that the requirement at destination j is supplied from the source i at time/facility k. Each destination should get its requirement from one source only and the source can supply to any number of destinations subject to its capacity. The problem is to find the Trip-schedule X i. e. a solution with a least total transportation cost.
In the sequel we developed a Lexi-search algorithm based on the "Pattern Recognition Technique" to get the optimal solution for the problem and it takes care of the simple combinatorial structure of the problem.

4. The Concepts and Definitions:

Definition of pattern:
An indicator three dimensional array which is associated with a solution is called "pattern". A pattern is said to be feasible if the matrix X is feasible solution. The pattern given by Table-2 is a feasible solution, whereas the pattern given by table-3 is in feasible.
The words pattern, solution, and word are used synonymously.
Now V (x) be the value of the pattern X is defined as

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

The value V (x) gives the total cost of transportation for the solution represented by X. Thus the value of the feasible pattern V(X) gives the cost represented by it. In the algorithm which is developed in the sequel a search is made for the pattern X which is represented by the set of ordered triple (i, j, k) for which x (i, j, k) = 1 with the understanding that the other X (i, j, k)'s are zeros.
There are n^3 ordered triples in the three dimension array X. For convenience these are arranged in ascending order of their corresponding costs and are indexed from 1 to n^3 (Sundara Murthy, 1973). Let SN = [1, 2,----, n^3] be the set of n^3 indices. Let D (i. e.

C(i, j, k)) be the corresponding array of costs and arranged them in ascending order (i. e. if a, b ε SN and a < b then D (a) ≤ D (b)). Also let the arrays R, C and T be the array of row, column and time / facility indices of the ordered triples represented by SN and DC be the cumulative sums of the elements of D. The arrays SN, D, DC, R, C, T for the numerical example are given in table-4. If t ε SN the (R (t), C (t), T (t)) is the ordered triple and D (a) = C (R(a), C (a), T (a)) is the value of the ordered triple and $DC(a) = \Sigma D(i)$ where i=1 to a.

Definition of an Alphabet-Table and a word:

Let $L_k = \{a1, a_2,-----, a_k\}$, a_i ε 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 indicies are arranged in the increasing order such that $a_i < a_{i+1}$, i = 1, 2,----, k-1. The set SN is defined as the "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 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 an "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 atleast 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 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 word, defined by it has atleast one feasible word.

Value of the word:

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

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.

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

$$= V(L_k) + DC(a_k + n-k) - DC(a_k)$$

Feasibility criterion of a partial word:

A feasibility criterion is developed, in order to check the feasibility of a partial word

$L_{k+1} = (a_1, a_2, \text{-----} a_k, a_{k+1})$ given that L_k is a feasible word. We will introduce some more notations which will be useful in the sequel.

IC be an array where IC (j) = 1, j ε J = (1, 2,----, n) represents that destination j is getting it supply from some source i, otherwise IC (j) = 0

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

Then for a given partial word $L_k = (a_1, a_2, \text{----}, a_k)$

RX be an array, where RX (i), is the total quantity supplied from the source i to the destinations.

The values of the arrays IC, IT, LW are as follows

IC (C (a_i)) = 1, i = 1, 2,-----, k and IC (j) = 0 for other elements of j

LW (i) = α_i, i = 1, 2,-----, k, and LW (j) = 0, for other elements of j

RX (R(a_i)) = RX (R(a_i)) + DR (C(a_i)), i=1, 2, ……, k and RX (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 RA = R (a_{p+1}), CA = C (a_{p+1}) and TA = T (a_{p+1}).

Algorithm-1:

STEP 1: IX = 0

IS (IC (CA) = 0) IF YES GO TO 2

IF NO GO TO 4

STEP 2: IS ((RX (RA) + DR (CA)) ≤ SC (RA))

IF YES GO TO 3

IF NO GO TO 4

STEP 3: I X = 1 GO TO 4

STEP 4: STOP

We start the algorithm with a very large value say 9999 as a trial value 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 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 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 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 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) ≥ 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.

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

feasible word.

5. Algorithm-2: (Lexi-Search algorithm)

STEP 1: Initialization

The arrays SN, D, DC, R, C, T, SC, DR, N, are made available. IC, RX, LW, V, LB 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-I
V (I) = V (I-1) + D (J)
LB (I) = V (I) + DC (JA)-DC (J)
GOTO 4

STEP 4: IS (LB (I) $\geq$ VT) IF YES GO TO 10
IF NO GO TO 5

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

STEP 6: Check the feasibility of L (I) (using algorithm 1)
IS (IX = 1) IF YES GO TO 7
IF NO GO TO 2

STEP 7: IS (I = N) IF YES GOTO 9
IF NO GO TO 8

STEP 8: L (I) = J
IC (CA) = 1
RX (RA) = RX (RA) + DR (CA)
I = I + 1
(I $\leq$ N) GO TO 2

STEP 9: L (I) = J, L (I) is a full length word and is feasible
VT = V (I), Record L (I) and VT
GOTO 11

STEP 10: IS (I = 1) IF YES GO TO 12
IF NO GO TO 11

STEP 11: I = I-1
J = L (I)
CA = C (J)
RA = IR (J)
IC (CA) = 0
RX (RA) = R X (RA)-DR (CA)
GOTO 2

STEP 12: STOP

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

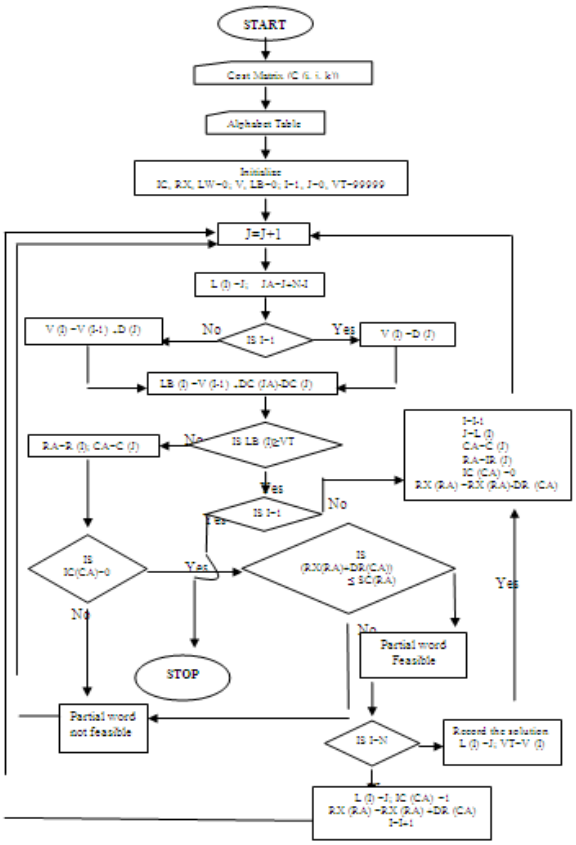
7. Computational Experience:

A computer program for the above algorithm is written in 'C' language and is tested on the system Pentium IV. Random numbers are used to construct the cost matrix. The following Table-10 gives the list of the problems tried along with the average CPU time, in seconds required for solving them.

In the table AT represents the CPU time to construct the Alphabet Table, ET represents the CPU time taken for the search of a feasible word. The time is represented in seconds. In table-10, n is the number of destinations, m is the number of sources and p is the number of time/facilities. It is seen that the time required for the search (ET) of the optimal solution is fairly less.

Table-10

n	m	p	AT	ET
6	4	3	0. 11	0. 05
8	6	5	0. 32	0. 16
10	8	6	0. 65	1. 7
12	10	8	1. 31	2. 85
15	13	10	2. 63	3. 9



Flow chart for TDBTP

REFERENCES

1. Balas, E (1965): An additive algorithm for solving linear programming problems with 0-1 variables. Opns. Res., 13, pp. 517-546.
2. Bhavani, V and Sundara Murthy M (2005): Time Dependent Travelling salesman Problem, Opsearch, 42, pp 199-227.
3. Demaio, A and Roveda, C (1971): An all 0-1 algorithm for a certain class of transportation Problems, Opns. Res. 19, pp. 1406-1418.
4. Glover. F (1965): A Multiphase-Dual algorithm for the 0-1 Integer programming problem, Opns. Res. 13, pp. 879-919.
5 Miller, C. E. (1960): Integer Programming formulation of Travelling salesman problem through 'n' sets of nodes, paper presented at the 9th annual convention of ORSI, Culcutta.
6. Pandit, S. N. N. and Sundara murthy, M (1975): Allocation of sources to destinations-paper presented in ORSI, held at Bombay.
7. Perskalla, W. P. (1966): The Tri-substitution method for obtaining near optimal solution to the three dimensional Assignment Problem, Tech. Memo, No. 71, Institute of Technology, Cleveland, Ohio.
8 Picard, J. L and Queyaranne, M (1978): The Time Dependent Travelling Salesman Problem and its applications to the Tradiness Problem in one machine scheduling, Opns. Res., 26, pp 86-110.
9 Shila Das and Monisha Borthakur (2006): A mixed Constrained (Identical) Vechical Routing Problem for Time minimization, Opsearch, 43, pp 31-48.
10 Srinivasan V and Thamspson G. L (1973): An algorithm for assigning uses to sources in a Special class of Transportation Problems. Opns. Res. 21, No. 1 pp. 285-295
11 Sundara Murthy, M (1976): Bulk Transportation Problem, Opsearch, 13, Pp 143-155
12 Sundara Murthy. M (1979): Combinatorial programming-A pattern Recognition approach, Ph. D. Thesis REC, Warangal, India.
13 Vanita Verma and M. C. Puri (1996): A branch and bound procedure for cost minimizing Bulk Transportation Problem, Opsearch, 33, pp:145-161