An Exact Algorithm for Multi – Product Bulk Transportation Problem

Purusotham, S [#1] and Sundara Murthy, M [#2]

#1 Purusotham, S., Lecturer, Dept. of Mathematics, S. V. J. College, TTD, Tirupati, Andhra Pradesh, India.
#2 Sundara Murthy, M., Professor, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India.
Email: drpurusotham.or@gmail.com; profmurthy@gmail.com

Abstract:— The paper investigates an NP-Hard nature Problem, where several commodities are produced in several plant sites with capacity constraints, and distributed to several destination sites according to demands and transportation constraints. We deal with the special case where the cost of the transportation of the goods from plants to warehouse is a bulk cost. The problem becomes Multi-Product Bulk Transportation Problem (MPBTP) where one desires to get the requirement of different products depending on the availability from any plants. The model intends to minimize the total cost of the bulk transportation for meeting the demands of all products specified over the planning horizon of various warehouses while satisfying the capacity availability of the production plants without according priorities to them at a given time/facility. The practical restriction is that the entire requirement of each warehouse is to meet from one or more plants and a plant can supply to any number of destinations subject to the capacity available of the product at it.

For this problem we developed a Pattern Recognition Technique based Lexi Search Algorithm, which comes under the exact methods. The concepts and the algorithm involving in this problem are discussed with a suitable numerical example. We programmed the proposed Lexi Search algorithm using C. This algorithm takes less CPU run time and hence it suggested for solving the higher dimensional problems.

Keywords: — Bulk Transportation problem, Lexi-Search Algorithm, Pattern Recognition Technique.

I. INTRODUCTION

The Classical Transportation Problem is to minimize the total cost for shipping the various capacities of the goods on the requirement of destinations from the available sources. The model often can be built as a linear programming model, an NP-hard problem. Generally the transportation cost of one unit of a commodity is depending on the source and the destination. In some situations the cost may not depend on the actual amount is transported, in that situation the cost is treated as 'Bulk-Transportation cost'. This leads to one more generalization of the transportation problem is that the Bulk Transportation Problem. The objective function in this case is $\Sigma\Sigma\ C(i, j) * H(x(i, j))$, where H is a step function with $H(\alpha) = 1$ if $\alpha > 0$ and $H(\alpha) = 0$ if $\alpha = 0$, and C(i, j) is the Bulk-Transportation cost. This was first investigated by Maio and Roveda [8], developed a branch and bound algorithm to solve it. Later, Srinivasan and Thompson [21] developed a branch and bound algorithm for the same problem and formulated a modified transportation problem for which the optimal solution of the above problem will be a basic feasible solution. The main drawback of Demaio and Reveda's [8] 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.

Sundara Murthy[22] studied this problem with the additional restriction that a destination should get its supply from one source only, solved with the efficient Lexi-Search algorithm using the Pattern Recognition Technique and mentioned that the efficiency of his algorithm over branch and bound algorithm.

A few additional remarks are in order concerning how the present model fits into the existing literature. Its chief ancestors are, of course, the well known and much simpler "Plant Location" models (see Balinski and Spielberg [2]; Ellwein and Gray [4] for surveys). These are basically single commodity transportation problems with fixed charges for the use of sources. Often the sources are assumed to have unlimited capacity. A natural extension of the capacitated plant location problem to the optimal location intermediate facilities in multi-echelon systems has been studied by Marks et.al. [9], they report reasonably good computational experience with a conventional branch and bound algorithm in which the linear programs, which specialize to capacitated trans-shipment problems.

Multi-index transportation problems are the extension of conventional transportation problems, and are appropriate for solving transportation problems with multiple supply points, multiple demand points as well as problems using diverse modes of transportation demands or delivering different kinds of products. Thus the forwarded problem would be more complicated than conventional transportation problems. Junginer [6] proposed a set of logic problems to solve multi-index transportation problems has also conducted a detailed

investigation regarding the characteristics of multi-index transportation problem model. Rautman *et al.* [17], used multi-index transportation problem model to solve the shipping scheduling suggested that the employment of such transportation efficiency but also optimize the integral system. These references are only a single objective model and its constraints are not fuzzy member. In the case of cost-time trade- off bulk transportation problem prakash *et al.* [14] studied multi objective models and presented efficient heuristic for this multi objective bulk transportation problem. Naganna, B -[10] studied a problem titled 'Time Dependent Bulk Transportation problem' and solved with Pattern Recognition based Lexi-Search Approach.

Another variation of this problem is that if a plant is producing different commodities instead of single commodity and the objective is to shipping all the commodities to warehouses with the minimum bulk cost, then the problem becomes the multi-commodity bulk transportation problem, studied by Sobhan Babu, K and Sundara Murthy, M [19], [20]. In his problem the cost is depending on the source, the destination and type of commodity. The problem often involves three dimensions; the third dimension refers to the commodity. The problem is then solved by using Pattern Recognition based Lexi-Search Approach and mentioned the efficiency of his algorithm over the Branch and Bound Approach. In the next section we discuss the brief description of the proposed problem nature.

II. PROBLEM DISCRIPTION

Here we consider an NP-Hard nature Problem, where several commodities are produced in several plant sites with capacity constraints, and distributed to several destination sites according to demands and transportation constraints. We deal with the special case where the cost of the transportation of the goods from plants to warehouse is a bulk cost. The problem becomes Multi-Product Bulk Transportation Problem (MPBTP) where one desires to get the requirement of different products depending on the availability from any plants.

The model intends to minimize the total cost of the bulk transportation for meeting the demands of all products specified over the planning horizon of various warehouses while satisfying the capacity availability of the production plants without according priorities to them at a given time/facility. The practical restriction is that the entire requirement of each warehouse is to meet from one or more plants and a plant can supply to any number of destinations subject to the capacity available of the product at it. The bulk transportation cost generally depends on 'i' and 'j'. But some times the cost can be influenced by some other independent factors. 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.). The problem often involves three dimensions. Under this consideration Picard *et al.* [13], Bhavani and Sundara Murthy [3], S. Das [18], Naganna [10], Soban Babu *et al.* [20] have studied a variety of problems.

Let there are be ***'m'*** plants, each plant ***(i)*** producing ***'p'*** products and there are be ***'n'*** warehouses, each warehouse ***(j)*** requires the same ***'p'*** products at a given time/ facility. The bulk transportation cost from a plant point ***'i'*** to the warehouse point ***'j'*** at a given time/facility ***'k'*** is $C(i, j, k)$. 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' as explained in the above. The capacity of each product at some plant ***'i'*** is denoted by $S(i, p)$ and the requirement of the same product at warehouse ***'j'*** is indicated by $D(j, p)$. In case of the usual constraints, the task can be expressed as a zero-one integer linear programming problem, where the binary variables are applied for deciding whether the goods are transported from plants to warehouses or not. The following diagram represents that the brief outline of the present problem.

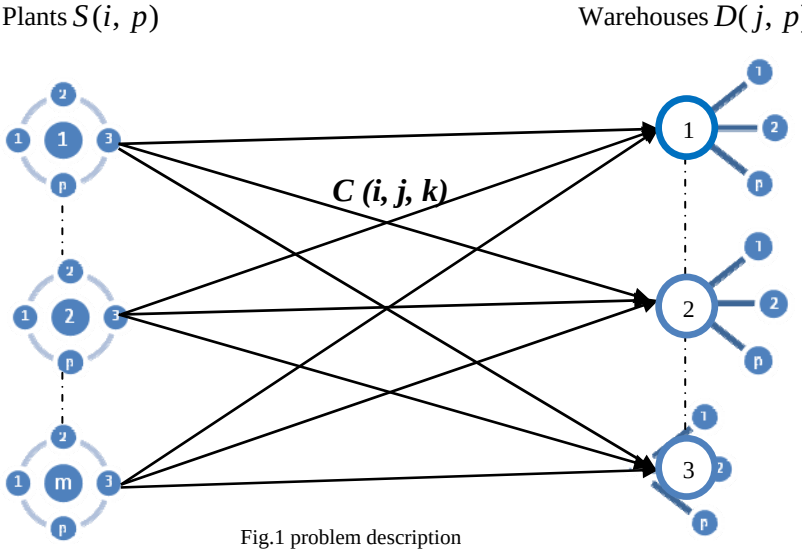

Fig.1 problem description

III. MATHEMATICAL FORMULATION OF MPBTP

$$Min \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} C(i, j, k)\, X(i, j, k) \qquad \text{..... (4.3.1)}$$

Subject to the constraints:

$$\sum_{j=1}^{n} \sum_{k=1}^{l} D^1(j, p)\, X(i, j, k) \le S(i, p), \quad \forall \quad i, p \qquad \text{..... (4.3.2)}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{l} D^1(i, p)\, X(i, j, k) = D(j, p), \quad \forall \quad j, p \qquad \text{..... (4.3.3)}$$

$$X(i, j, k) = 0 \quad or \quad 1 \qquad \text{..... (4.3.4)}$$

$$\text{If } X(i_1, j_1, k_1) = X(i_2, j_2, k_2) = 1 \text{ and } i_1 = i_2, j_1 \neq j_2 \text{ then } k_1 = k_2 \qquad \text{..... (4.3.5)}$$

Here $D^1(i, p)$ denotes the quantity of p^{th} product is supplied from i^{th} plant to some warehouse. $D^1(j, p)$ is the quantity of p^{th} Product supplied to j^{th} warehouse from a plant.

Constraint (**4.3.1**) represents that the total cost of the bulk supply of the goods from plants *(i)* to the warehouses *(j)* and utilizing the facility *(k)*. Constraint (**4.3.2**) & (**4.3.3**) represents that a plant can supply its capacity of the goods to more than one warehouse, a warehouse get its requirement from more than one plant respectively. Constraint (**4.3.5**) indicates that if a plant is supplying its products more than one warehouse then the supply must use the same facility. Finally the binary values of (**4.3.4**) indicate that whether the goods are supplied from a plant to a warehouse or not.

The above Bulk Transportation without time/facility and with single product can be thought of as a two dimension bulk transportation problem with cost *C (i, j)*. The problem can be formulated and solved it as Integer programming problem. 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'. For availing the simplicity in the combinatorial structure of this problem, we developed a Lexi-Search algorithm by using Pattern Recognition Technique. In the present paper, we illustrated the problem and explained the concepts with a suitable numerical example.

IV. LEXICOGRAPHIC SEARCH APPROACH

Lexicographic Search Approach is a systematized Branch and Bound approach, developed by Pandit in the context of solving of loading problem in 1962. In principle, it is essentially similar to the Branch and Bound

method as adopted by Little et.al.-1963. This approach has been found to be productive in many of the Combinatorial Programming Problems. It is significance mentioning that Branch and Bound can be viewed as a particular case of Lexicographic Search approach [Pandit [11], [12]]. The name Lexicographic Search itself suggests that, the search for an optimal solution is done in a systematic manner, just as one searches for the meaning of a word in a dictionary and it is derived from Lexicography the science of effective storage and retrieval of information. This approach is based on the following grounds [Pandit [11]].

(i) It is possible to list all the solutions or related configurations in a structural hierarchy which also reflects a hierarchical ordering of the corresponding values of these configurations.

(ii) Effective bounds can be set to the values of the objective function, when structural combinatorial restraints are placed on the Allowable configurations.

4.1 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 we get two situations in the choice of the alphabet-table [Sundara Murthy, M [22]].

(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** (Sundara Murthy [22]) 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. The pattern-recognition technique can be described as follows.

"A unique pattern is associated with each solution of a problem. Partial pattern defines a partial solution. An alphabet-table is defined with the help of which the words, representing the pattern are listed in a Lexicographic order. During the search for an optimal word, when a partial word is considered, first bounds are calculated and then the partial words for which the value is less than the trail value are checked for the feasibility"

Lexi Search algorithms, in general, require less memory, due to the existence of the Lexicographic order of the partial words. Using Pattern Recognition technique reduces the dimensions requirement of the problem. If Pattern Recognition is used, the problem can be reduced to a linear form of finding an optimal word of length n. This reduction in the dimension for some problems reduces the computational work in getting an optimal solution [Sundara Murthy M. [22], Ramana [23], Purusotham et.al. [15], [16]]. The present paper uses the Lexicographic Search in general and makes use of the Pattern Recognition approach at the appropriate situations.

4.2 Definition of Pattern and Word:

An indicator three dimensional array X which gives the supply schedule of the goods and is transported from plants to warehouses is called a "**pattern**". A pattern is said to be feasible if X has a feasible solution. Now the value of the pattern X is defined as follows.

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

The value $V(X)$ gives the total cost of the MPBTP of the solution represented by X. Thus the value of the feasible pattern gives the total cost. 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. There are **Max= *m* * *n* * *l*** ordered triples in the three dimensional array X. These are arranged in ascending order of their corresponding costs and are indexed from 1to M (Sundara Murthy, [22]). The set SN is defined as

the **'Alphabet Table'** with alphabetic order as (1, 2… Max). In the next section a suitable numerical example is discussed.

4.3 Numerical Illustration:

The concepts and the algorithm developed will be illustrated by a numerical example for which M = 4, N = 6, L = 2, P = 2. The cost matrix $C(i, j, k)$ of BTP is given as follows.

Table –1

$C(i, j, 1)$ =						
	14	25	11	07	40	30
	16	22	06	33	27	15
	18	20	31	03	17	09
	07	19	35	12	18	21

$C(i, j, 2)$ =						
	30	19	25	14	02	24
	05	18	16	19	08	42
	40	10	04	24	15	37
	16	13	23	33	32	07

The corresponding capacity of the i^{th} plant is *S(i, p)* and the requirement of j^{th} warehouse is *D(j, p)* for various products is given below.

$S(i, p)$ =		
	150	50
	80	200
	70	150
	100	50

$D(j, p)$ =						
	30	60	50	70	50	40
	80	100	30	20	120	50

In this numerical example we are given four plants and six warehouses. Assume that each plant is producing two different products. Now the problem is to minimizing the total transportation cost of the goods from the production plants to warehouses by utilizing the given facilities, subject to the resource constraints of the bulk transportation.

The entire $C(i, j, k)$'s are taken as non–negative integers but it can be easily seen that this is not a necessary condition and the cost can as well as real quantities. For example $C(3, 6, 1)$ = 09, represents the bulk cost of transporting the required goods of 6th warehouse subject to the availability from 3rd plant by utilizing the first facility, S(3, 1) = 70 and S(3, 2) = 150 indicates that the corresponding production capacity of two types of products at the 3rd plant. Similarly D(6, 1) = 50 and D(6,2) = 120 shows that the required capacity of the same products at 6th warehouse.

An indicator matrix *X = [X (i, j, k) / X (i, j, k)= 0 or 1]* in which *X (i, j, k)=1* indicates that the i^{th} plant is supplied its capacity to jth warehouse subjected to the availability and requirement of goods by utilizing a given facility, otherwise *X (i, j, k)=0*. *X* is called a solution. The indicator matrix *X* with 0 or 1 is given in the following table.

Table – 2

$$X(i, j, 1) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad X(i, j, 2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The indicator matrix *X* is represented in Table – 2 is feasible. The corresponding ordered triples of *X* are indicated as ***(1, 4, 1), (2, 1, 2), (2, 5, 2),(3, 2, 2), (3, 3, 2), (4, 3, 1) (4, 6, 1)***. The representation of the solution *X (1, 4, 1) =1* to the problem is that the first plant supplied its products to 4th warehouse utilizing first facility. Similarly, the second plant supplied its products to first and fifth warehouses, the third plant is supplied its products to 2nd and 3rd warehouses by utilizing the second facility and the 6th and 3rd warehouses received the required capacity of goods from 4th plant by utilizing the first facility. For the above example of the feasible allocation set is represented below.

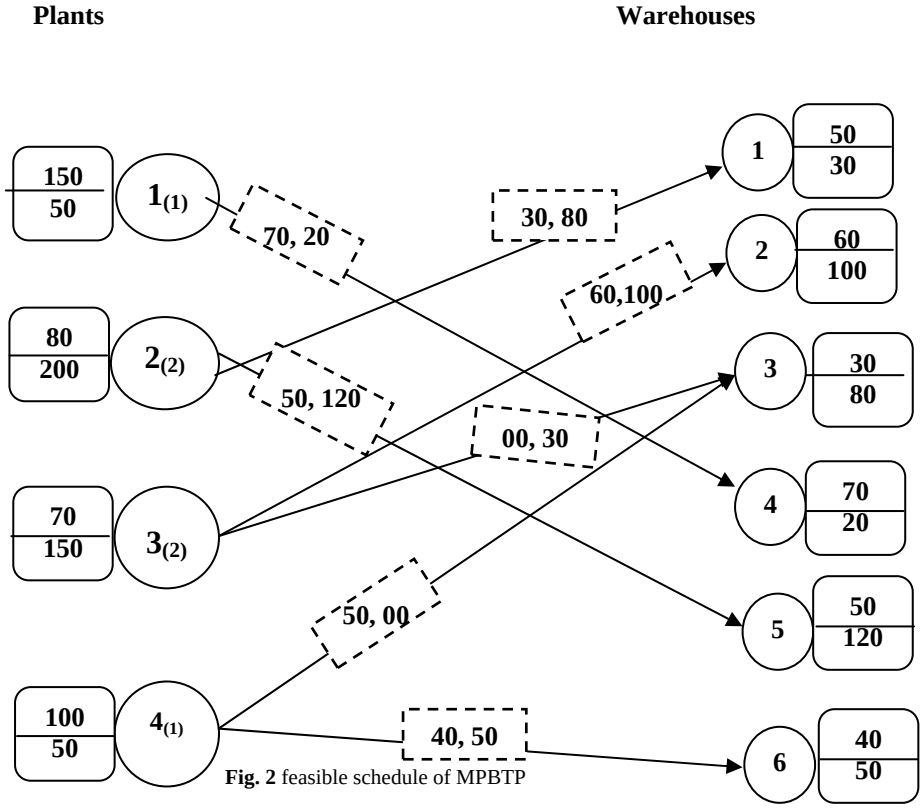

Fig. 2 feasible schedule of MPBTP

In *Fig.2* the values in the rectangular boxes of the left most and the right most indicate the available capacity and requirement of the two types of goods respectively. The numbers in the circles represented plants and warehouses. The suffixed numbers at plants measures that the goods are transported by utilizing the respective facilities for this example. The values in the doted rectangular boxes show the amount of the goods shipped from a plant to warehouse. Now this pattern satisfies all the constraints of the problem nature, hence it is called a feasible supply schedule.

4.4 Steps in LSA: The proposed LSA approach consist the following steps while processing the search for an optimal solution of MPBTP.

a. Sorting the cost matrix (Alphabet Table)
b. Checking the feasibility
c. Effective bound settings
d. Searching mechanism towards the optimal solution

a) Sorting the cost matrix (Alphabet Table):

Let $\mathbf{L_k} = \{\alpha_1, \alpha_2 \ldots\ldots, \alpha_k\}$, $\alpha_i \in \mathbf{SN}$ be an ordered sequence of **k** indices from **SN**. The pattern represented by the ordered triples whose indices are given by $\mathbf{L_k}$ is independent of the order of α_i in the sequence, the indices are arranged in the increasing order such that $\alpha_I < \alpha_{i+1}$, *i =1, 2… n-1*. The set **SN** is defined as the **'Alphabet Table'** with alphabetic order as (*1, 2… Max*). The arrays **SN, D, DC, R, C and T** are represent the serial number, bulk cost, cumulative cost, row, column and time/facility indices respectively, see the following table.

Table – 3: ALPBHABET TABLE

SN	D	DC	R	C	T
1	1	1	1	4	1
2	2	3	1	5	2
3	3	6	3	4	1
4	4	10	3	3	2
5	5	15	2	1	2
6	6	21	2	3	1
7	7	28	4	1	1
8	7	35	4	6	2
9	8	43	2	5	2
10	9	52	3	6	1
11	10	62	3	2	2
12	11	73	1	3	1
13	12	85	4	4	1
14	13	98	4	2	2
15	14	112	1	1	1
16	14	126	1	4	2
17	15	141	2	6	1
18	15	156	3	5	2
19	16	172	2	1	1
20	16	188	2	3	2
21	16	204	4	1	2
22	17	221	3	5	1
23	18	239	3	1	1
24	18	257	4	5	1

Table – 3: ALPBHABET TABLE

SN	D	DC	R	C	T
25	18	275	2	2	2
26	19	294	4	2	1
27	19	313	1	2	2
28	19	332	2	4	2
29	20	352	3	2	1
30	21	373	4	6	1
31	22	395	2	2	1
32	23	418	4	3	2
33	24	442	1	6	2
34	24	466	3	4	2
35	25	491	1	2	1
36	25	515	1	3	2
37	27	542	2	5	1
38	30	572	1	6	1
39	30	602	1	1	2
40	31	633	3	3	1
41	32	665	4	5	2
42	33	698	2	4	1
43	33	731	4	4	2
44	35	766	4	3	1
45	37	803	3	6	2
46	40	843	1	5	1
47	40	883	3	1	2
48	42	925	2	6	2

Feasibility criterion of partial word:

A recursive algorithm is developed for checking the feasibility of a partial word. A leader $\mathbf{L_k}$ is said to be feasible if the block of words defined by it contains atleast one feasible word. Let $L_{k+1} = (\alpha_1, \alpha_2 \ldots \alpha_k, \alpha_{k+1})$ given that L_k is a feasible partial word. We will introduce some more notations which are useful in the sequel.

IR be an array where IR (i) =1, i ε m represents that the goods of *ith* plant is completely transported to warehouses, otherwise zero.

IC be an array where IC (j) =1, $j \varepsilon n$ indicates that the required goods at *ith* warehouse is completely received from plants, otherwise zero.

SX be an array where $SX(i) = k, i \varepsilon m, k \varepsilon l$ indicates that the i^{th} plant is utilized the k^{th} facility on transporting the goods to some warehouse.

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

S be an array where S (i, q) = α, $i \varepsilon m$ and $q \varepsilon p$ indicates that the quantity (α) of q^{th} product is available at i^{th} plant.

D be an array where D (j, q) = γ, j ε n and $q \varepsilon p$ indicates that the quantity (γ) of q^{th} product is required at j^{th} warehouse.

The recursive algorithm for checking the feasibility of a partial word L_k 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 are consider **RA=R (α_k), CA=C (α_k), and TK = T (α_k).**

b) Algorithm 1: (Checking the Feasibility of a partial word)

Step 1: IX = 1 ***Goto2***

Step 2: IS (IR (RA)) = 1) ***If Yes Goto10; Else Goto3***

Step 3: IS (IC (CA)) = 1) ***If Yes Goto10; Else Goto4***

Step 4: SX (RA) = K

IS (K = TK) ***If Yes Goto4a; Else Goto10***

Step 4a: ST = ST + 1 ***Goto5***

Step 5: S (RA, q) =α; D (CA, q) = γ

IS ($\alpha < \gamma$) ***If Yes Goto6; Else Goto7***

Step 6: S (RA, q) = S (RA, q) – α, D (CA, q) = D (CA, q) – α Goto8

Step 7: S (RA, q) = S (RA, q) – γ, D (CA, q) = D (CA, q) – γ Goto8

Step 8: IX = 1(the required goods are partially supplied from the plant) Goto10

Step 9: IX = 1 (the required goods are fully supplied from the plant) Goto10

Step 10: STOP.

c) Upper and Lower Bounds:

This recursive algorithm is used in Lexi Search algorithm to check the feasibility of a partial word. We start the algorithm with a high value say 'HV = ∞' 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 the search the current value of VT gives the optimal solution of a feasible word. We start the partial word $L_1=(a_1)=(1)$. A partial word L_k is constructed as $L_k = L_{k-1} * (a_k)$ where * indicates concatenation i.e. chain formation. We will calculate the values of V (L_k) and LB (L_k) simultaneously using the following formulas.

Now the value of the word $\boldsymbol{L_k}$ is defined as follows.

$$\boldsymbol{V(L_k) = V(L_{k-1}) + D(\alpha_k) \text{ with } V(L_0) = 0}$$

A lower bound LB (L_k) for the values of the block of words represented by $\mathbf{L_k}$ can be defined as follows.

$$\boldsymbol{LB(L_k) = V(L_k) + DC(\alpha_k + n_o) - DC(\alpha_k)}$$

Consider the partial word $\mathbf{L_4 = (1, 5, 8, 12)}$

$\mathbf{V(L_4)} = 01 + 05 + 07 + 11 = 24$

$\mathbf{LB(L_4)} = V(L_4) + DC(\alpha_4 + 6 - 3) - DC(\alpha_4)$

$= V(L_4) + DC(12+6-2) - DC(12)$

$= 24 + 98 - 73 = 49$

Two cases arise when calculating the bounds, one for branching and the other for continuing the search.

1. **$LB(L_k) < VT$**. Then we check whether L_k is feasible or not. If it is feasible we proceed to consider a partial word of order (k+1), which represents a sub block of the block of words represented by L_k. If L_k is not feasible then consider the next partial word of order by taking another letter which succeeds a_k in the k^{th} position. If all the words of order 'k' are exhausted then we consider the next partial word of order (k–1).
2. **$LB(L_k) > VT$.** In this case we reject the partial word L_k. We reject the block of word with L_k as leader as not having optimum feasible solution and also reject all partial words of order 'k' that succeeds L_k.

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

d) Algorithm -2 (Lexi - Search Algorithm (LSA) –search mechanism for getting an optimal solution)

Step 1: (Initialization)
The arrays SN, D, DC, R, C, T, G (i, q), H (j, q) and the values of m, n, l are made available IR, IC, IT, L, V, LB are initialized to zero. The values I =1, J = 0, VT =HV (High value), MAX = $m * n * l - n$, $n_o = n$.

Step 2: ***J = J + 1***
IS (J > MAX) ***If Yes Goto14; Else Goto3***

Step 3: ***L (I) = J***
IS (I=1) ***If Yes V (I) =D (J), goto3B; Else Goto3A***

Step 3A: V (I) =V (I – 1) + D (J) ***Goto3B***

Step 3B: LB (I) =V (I) +DC (J + n_o – I) – DC (J) ***Goto4***

Step 4: ***IS (LB (I) ≥VT)*** ***If Yes Goto11; Else Goto5***

Step 5: ***RA = R (J)***
CA = C (J)
TK = T (J) ***Goto6***

Step 6: ***Check the feasibility of L (using algorithm 1)***
IS (IX=0) ***If Yes Goto2; Else Goto7***

Step 7: ***IS (IX=1) (with partial supply)*** ***If Yes Goto7b; Else Goto7a***

Step7a: ***IS (IX=1) (with complete supply)*** ***If Yes Goto8; Else Goto2***

Step7b: ***IS (I= m * n * l)*** ***If Yes Goto12; Else Goto7c***

Step7c: ***J = J + 1***
$n_o = n_o + 1$ ***Goto3***

Step 8: ***IS (I > = n)*** ***If Yes Goto9; Else Goto10***

Step 9: ***L (I) =J***
L (I) is full length word and is feasible
VT=V (I), record L (I), VT, G (i, q), H (i, q) ***Goto13***

Step 10: ***IS (S(RA, q) = 0)*** ***If Yes IR (RA) = 1***
Else IR (RA)=0 Goto10a

Step 10a: ***IS (D(CA, q) = 0)*** ***If Yes IC (CA) = 1***
Else IC (CA)=0 Goto10b

Step10b: ***S(RA, q) = δ, D(CA, q) = μ***
I = I + 1 ***Goto 2***

Step 11: ***IS (I = 1)*** ***If Yes Goto14; Goto12***

Step 12: ***I = I – 1*** ***Goto13***

Step 13: ***J = L (I); RA = R (J)***
CA = C(J); TK = T(J)
IR(RA)=0; IC (CA) = 0
ST = ST – 1
S (RA, q) = S (RA, q) + α, D (CA, q) = D (CA, q) + α (or)
S (RA, q) = S (RA, q) + γ, D (CA, q) = D (CA, q) + γ ***Goto2***

Step 14: ***STOP & END***

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

4.5 Search Table:

The working details of getting an optimal word using the above algorithm for the illustrative numerical example is given in the following table. The columns named (1), (2), (3), (4), (5), (6) and (7) gives the letters in the first, second, third, fourth, fifth, sixth and seventh places of a word respectively, the corresponding V(I) and LB(I) are indicated in the next two columns. The column R, C and K gives the row, column and time/facility indices of the letter. The last column gives the remarks regarding the acceptability of the partial words. In the following table A, A+ indicates the acceptance of partial supply, complete supply and R for Rejectance.

Table – 4: [Search Table (ST)]

SN	1	2	3	4	5	6	7	V	LB	R	C	K	REM
1	1							1	21	1	4	1	A+
2		2						03	21	1	5	2*	R
3		3						04	26	3	4*	1	R
4		4						05	30	3	3	2	A+
5			5					10	30	2	1	2	A+
6				6				16	30	2	3	1*	R
7				7				17	32	4	1*	1	R
8				8				17	34	4	6	2	A+
9					9			25	34	2	5	2	A+
10						10		34	44	3	6*	1	R
11						11		35	46	3	2	2	A
12							12	46	46	1	3*	1	R
13							13	47	47	4	4*	1	R
14							14	48	48=VT	4	2	2	A+
15						12			36	1	3*	1	R
16						13			37	4	4*	1	R
17						14			38	4	2	2	A
18							15		52>VT				R
19						15			49>VT				R
20					10			26	36	3	6*	1	R
21					11			27	38	3	2	2	A
22						12		38	50>VT				R
23					12			37	37	1	3*	1	R
24					13			38	38	4	4*	1	R
25					14			30	44	4	2	2	A
26						15		44	58>VT				R
27					15			31	45	1	1*	1	R
28					16			31	46	1	4*	2	R
29					17			32	47	2	6*	1	R
30					18			32	48=VT				R
31				9				18	37	2	5	2	A+
32					10			27	37	3	6	1*	R
33					11			28	39	3	2	2	A
34						12		39	51>VT				R
35					12			29	41	1	3*	1	R
36					13			30	43	4	4*	1	R
37					14			31	45	4	2	2	A
38						15		45	59>VT				R
39					15			32	46	1	1*	1	R
40					16			32	46	1	4*	2	R
41					17			33	48=VT				R
42				10				19	40	3	6	1*	R
43				11				20	43	3	2	2	A
44					12			31	56>VT				R
45				12				21	46	1	3*	1	R
46				13				22	49>VT				R
47			6					11	33	2	3*	1	R
48			7					12	37	4	1	1	A
49				8				19	46	4	6	2*	R
50				9				20	50>VT				R
51			8					12	39	4	6	2	A+
52				9				20	39	2	5	2	A+
53					10			29	39	3	6*	1	R
54					11			30	41	3	2	2	A
55						12		41	53>VT				R
56					12			31	43	1	3*	1	R

57					13			32	45	4	4*	1	R
58					14			33	47	4	2	2	A
59						15		47	61>VT				R
60					15			34	48=VT	1	1*	1	R
61				10				21	42	3	6*	1	R
62				11				22	45	3	2	2	A
63					12			33	58>VT				R
64				12				23	48=VT				R
65			9					13	43	2	5	2	A+
66				10				22	43	3	6	1*	R
67				11				23	46	3	2	2	A
68					12			34	59>VT				R
69				12				24	49>VT				R
70			10					14	47	3	6*	1	R
71			11					15	51>VT				R
72		5						06	34	2	1	2	A+
73			6					12	34	2	3	1*	R
74			7					13	37	4	1*	1	R
75			8					13	40	4	6	2	A+
76				9				21	40	2	5	2	A+
77					10			30	40	3	6*	1	R
78					11			31	42	3	2	2	A+
79						12		42	42=VT	1	3	1	A+
80					12			32	44>VT				R
81				10				22	43>VT				R
82			9					14	44>VT				R
83		6						07	38	2	3	1	A+
84			7					14	38	4	1	1	A
85				8				21	48>VT				R
86			8					14	41	4	6	2	A+
87				9				22	41	2	5	2	A+
88					10			31	41	3	6*	1	R
89					11			32	43>VT				R
90				10				23	44>VT				R
91			9					15	45>VT				R
92		7						08	42=VT				R
93	2							02	27	1	5	2	A
94		3						05	34	3	4	1	A+
95			4					09	34	3	3	2*	R
96			5					10	38	2	1	2	A+
97				6				16	38	2	3	1*	R
98				7				17	41	4	1*	1	R
99				8					44>VT				
100			6					11	42=VT				R
101		4						06	39	3	3	2	A+
102			5					11	39	2	1	2	A+
103				6				17	39	2	3	1*	R
104				7				18	42=VT				R
105			6					12	43>VT				R
106		5						07	44>VT				R
107	3							03	32	3	4	1	A+
108		4						07	32	3	3	2*	R
109		5						08	36	2	1	2	A+
110			6					14	36	2	3	1*	R
111			7					15	39	4	1	1	A
112				8				22	49>VT				R
113			8					15	42=VT				R

114		**6**						**09**	**40**	**2**	**3**	**1**	**A+**
115			**7**					**16**	**40**	**4**	**1**	**1**	**A**
116				**8**				**23**	**50>VT**				**R**
117			**8**					**16**	**43>VT**				**R**
118		**7**						**10**	**44>VT**				**R**
119	**4**							**04**	**37**	**3**	**3**	**2**	**A+**
120		**5**						**09**	**37**	**2**	**1**	**2**	**A+**
121			**6**					**15**	**37**	**2**	**3**	**1***	**R**
122			**7**					**16**	**40**	**4**	**1***	**1**	**R**
123			**8**					**16**	**41**	**4**	**6**	**2**	**A+**
124				**9**				**24**	**41**	**2**	**5**	**2**	**A+**
125					**10**			**33**	**41**	**3**	**6***	**1**	**R**
126					**11**			**34**	**43>VT**				**R**
127				**10**				**25**	**46>VT**				**R**
128			**9**					**17**	**47>VT**				**R**
129		**6**						**10**	**41**	**2**	**3***	**1**	**R**
130		**7**						**11**	**45>VT**				**R**
131	**5**							**05**	**42=VT**				**R**

At the end of the search the current value of **VT** = **42** and it is the value of the feasible word ***L_6=(1, 5, 8, 9, 11, 12),*** it is given in **79th** row of search table and the corresponding ordered tipples are ***(1, 4, 1), (2, 1, 2), (4, 6, 2), (2, 5, 2), (3, 2, 2)*** and ***(1, 3, 1).*** For this optimal feasible word the arrays L, IR, IC, and SX are given in the following table.

Table 5:

	1	2	3	4	5	6	7
L	1	5	8	9	11	12	
IC	1	1	1	1	1	1	-
IR	0	1	0	0	-	-	-
SX	1	2	2	2	-	-	-

The pattern represented by the above optimal feasible word is represented in the following indicator matrix [*i.e.* the first plant is supplying its available capacity to meet requirement of goods at 3rd and 4th warehouses by utilizing the first facility. Similarly, the second plant is transported its capacity of products to 1st and 5th warehouses utilizing the second facility, and so on].

$$X(i, j, 1) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad X(i, j, 2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The diagrammatic representation of the final optimal solution for this numerical example of MPBTP is shown below. The narration of the diagram is similar as explained in fig. 2.

4.6 Experimental results:

A Computer program for the proposed algorithm is written in C language and is tested on the system COMPAQ dx2280 MT. We tried a set of problems for different sizes. Random numbers are used to construct the Time matrix. The following table - 6 gives the list of the problems tried along with the average CPU time in seconds required for solving them.

Table – 6														
SN	Problem dimensions				AT	CPU run time in seconds of the proposed LSA								
						TYPE 1			TYPE 2			TYPE 3		
	m	*n*	*l*	*P*		Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
1	4	6	2	2	0.0000	0.0000	0.0000	0.0000	0.000	0.0000	.0000	0.000	0.000	0.0000
2	6	5	2	2	0.0005	0.0047	0.0135	0.0091	0.000	0.0018	0.0009	0.0000	0.0000	0.0000
3	8	10	3	3	0.0549	0.0196	0.0885	0.0540	0.0059	0.0214	0.0136	0.0047	0.0153	0.0100
4	10	10	3	5	0.1168	0.1642	0.5219	0.3430	0.2643	0.5024	0.3833	0.1589	0.3126	0.02357
5	10	20	4	5	0.2243	0.8949	1.0716	0.9832	0.8432	1.0126	0.9279	0.6602	0.9431	0.8016
6	10	25	3	6	0.7325	1.1503	1.8420	1.4961	1.0476	1.2435	1.1455	1.0099	1.0894	1.0496
7	15	30	3	5	1.2046	1.7356	2.1638	1.9497	1.5615	1.9830	1.7722	1.3345	1.9187	1.6266
8	15	40	3	4	1.6735	2.3478	2.7034	2.5256	2.1467	2.5609	2.3538	2.0038	2.0164	2.0101
9	20	50	3	3	2.3456	2.9251	3.0027	2.9639	2.4804	2.7132	2.5968	2.2761	2.4895	2.3828
10	30	50	3	3	3.2419	3.1098	3.5663	3.3380	2.9267	3.0146	2.9706	2.1862	2.4533	2.3197

***SN = serial number, m= number of Plants, n = number of warehouses, l = number of facilities, p = number of products. AT represents the CPU run time for generating the Alphabet Table. In the next columns of Type1, Type 2 and Type 3 shows that the minimum, maximum and average CPU run time in seconds of the proposed Pattern Recognition Technique based Lexi – Search Algorithm for obtaining the optimal solution.*

Experiments are carried out on a COMPAQ (dx2280 MT) system 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, k)* are uniformly random in [1,100]
Type 2: a) *C (i, j, k)* are uniformly random in [1,100]
b) VT=0.85VT
Type 3: a) *C (i, j, k)* are uniformly random in [1,100]
b) *Max=(m *n * l)/3*

And the results are tabulated in the above Table. For each type, six data sets are tested. It is seen that time required for the search of the optimal solution is fairly less. 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. The graphical representation of the proposed Lexi Search Algorithm (LSA) for the CPU runs time of the problem instances given in Table – 6 is shown below.

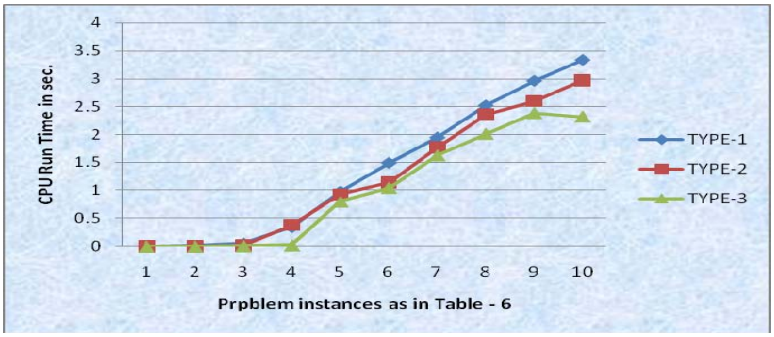

Figure 1

In Fig. 1 the problem instances presented in Table – 8 are taken on X – axis and the average CPU run time of the proposed LSA for three types of data sets as mentioned in the same table are taken on Y – axis. On observing the graph the time to obtain the solution is slightly increasing for higher dimensions in each type of generated data sets. When compared the three types of data sets the average CPU run time of Type-3 taken fairly less computational time than Type-1 and Type-2. Furthermore, the proposed LSA converge to the optimal solution in least time. Hence, we suggested the LSA for solving higher dimensional problems.

V. CONCLUSIONS

A commonly occurring problem in distribution system design is the optimal location intermediate distribution facilities between plants and warehouses. A multi-commodity/Product capacitated multi-period version of this problem is formulated as a mixed integer linear program. The major conclusion arising from this study is the remarkable effectiveness of Lexi Search procedure as a computational strategy for static multi-product intermediate location problems. The numerical problem quoted in 4.6 shows that only a few patterns are needed to find an optimal solution. One of the conclusions made on the basis of above discussion is that the problem is to transport the goods with minimum bulk cost from a set of production plants to a set of warehouse points utilizing a given facility. Since the amounts of supply and demand are of approximate values, the results obtained from the utilization of Lexicographic theory, interactive programming method based on the pattern recognition technique is developed.

The problem involving the overwhelming size of the system is quite difficult to solve. This study has utilized reducing to mixed integer linear programming to find the optimal solution, which, aside from eliminating the difficulty associated with the system resolution, can allow the patterns to quickly grasp the information offered by the system and, ultimately, elevate the quality and efficiency of search procedure.

This study has investigated the Lexi-Search Algorithm based on the Pattern Recognition Technique, via the simulation of random numbers in some partition points, to solve the MPBTP with indefinite weights. We considered the random numbers in this study for analyzing the effectiveness and efficiency of the LSA Approach. For better understood the concepts and the steps involved in the algorithm a suitable numerical example is quoted. The Lexi - Search algorithm presented in this chapter, incorporating Pattern Recognition Technique is tested. The same problem sets have been tested with using C language and successfully applied to many real problems. The findings of the model-testing and a wide range of sensitivity analyses using an artificially generated data set are presented. Both solution procedures prove to be efficient and effective in providing close to optimal solutions and proven with a surprisingly small number of patterns. Lexi-search algorithms are proved to be more efficient in many combinatorial problems. Lexi Search strategy stands as a good candidate for being an AI (Artificial Intelligence) search mechanism. Apart from the minimal requirement of memory, the Lexi Search helps to obtain optimality**,** faster than Branch and Bound in many cases. Further, Lexi Search clearly specifies Simpler rules for Branching, Bounding and Termination. Even with the restrictions imposed, the Lexi –Search takes reasonably less time. Its efficiency over the Bounding Procedures (like Lagrangean Relaxation) in Branch & Bound method is also significant. Further it is observed that with the modification of the sort procedure while arranging the alphabet table, the Lexi Search algorithm is becoming more efficient. On the whole, it is felt that Lexi Search algorithm is faster than the Branch and Bound algorithm.

References:

[1] Balas, E. An Additive Algorithm for Linear Programming in Zero-One Variables *Operations Research* 13(4), 1965; 517-546..

[2] Balinski, M. L. and Spielgerg, K. Methods for Integer Programming: Algebraic, Combinatorial and Enumerative, J. S. Aronofsky (ed), Progress in Operations Research, vol. III, Wiley, New York; 1969.

[3] Bhavani & Sundara Murthy M. Truncated M-Travelling Salesmen Problem, OPSEARCH, Vol.43, No.2, 2006.

[4] Ellwein, L. B. and P. Gray (1971) "Solving Fixed Charge Location-Allocation Problems with Capacity and Configuration Constraints," AIIE Transactions, Vol 3, No. 4, pp. 290-298.

[5] Glover , F. A Multiphase - Dual Algorithm for the 0-1 Integer Programming Problem, Opns.Res. 13, pp. 879-919, 1965.

[6] Junginer, W. On representatives of multi-index transportation problems. European Journal of Operational Research, 66, 353-371, 1993.

[7] Little, J.D.C. Murthy K.G. Sweeny, D.W. An algorithm for the TSP. Opns. Res., 11, pp.972-982, 1963.

[8] Maio Adriano De and Roveda, C. An all zero-one algorithm for a certain class of transportation problems. Operations Research, 19, 1406-1418. 1971.

[9] Marks, D. H., Liebman, J.C., and Bellmore, M. Optimal Location of Intermediate Facilities in a Trans-shipment Network, paper R-TP3.5 presented at the 37th National ORSA Meeting, Washington, D.C, 1970.

[10] Naganna B. Operations Research. Ph. D thesis, S. V. University, Tirupati, India. (2007).

[11] Pandit, S.N.N. An Intelligent Search approach to TSP - Symposium in OR, IIT, Kharagpur, 1965.

[12] Pandit, S.N.N. Some Combinatorial Search Problems, PhD thesis, IIT Kharagpur, India, 1963.

[13] Picard, J.C and Queyranne, M. The TDTSP and its application to the Tradiness problem in one machine scheduling, Opns.Res.,26,pp.86–110, 1978.

[14] Prakash, S, Kumar, P, Prasad B.V.N.S. and A. Gupta, Pareto optimal solutions of a cost–time trade-off bulk transportation problem, *European Journal of Operational Research* **188** (1) (2008), pp. 85–100.

[15] Purusotham, S. and Sundara Murthy, M A Variant Multi-Dimensional Assignment Problem, presented in the international conference on Operations Research for A Growing Nation in conjunction with 41st Annual convention ORSI held at Tirupati, 2008.

[16] Purusotham, S. and Sundara Murthy,M. A New Approach for Solving the Network Problems, presented in the international conference on Operational Research for Urban Rural Development in conjunction with 43rd Annual convention ORSI held at Madurai, 2010.

[17] Rautman, C.A., Reid, R.A., and Ryder, E.E. Scheduling the disposal of nuclear waste material in a geologic repository using the transportation model. Operations Research 41, 459-469, 1993.

[18] Shila Das. Routing and Allied Combinatorial Problems - Ph.D., Thesis, Dibrugrah University, Dibrugrah Assam, India, 1976.

[19] Sobhan Babu, K. and Sundara Murthy, M. (2010). An Efficient Algorithm for Variant Bulk Transportation Problem. International Journal of Engineering Science and Technology, vol. 2(7), 2595-2600.

[20] Sobhan Baubu, K., Chandrakala, K., Purusotham, S. and Sundara Murthy, M: A New Approach for Variant Multi Assignment Problem. International Journal on Computer Science and Engineering, Vol. 2(5): 1633-1640, 2010.

[21] Srinivasan V. & G.L. Thompson (1973). An Algorithm for Assigning Uses to Sources in a Special Class of Transportation Problems, Op. Res., Vol. 21, No.1.

[22] Sundara Murthy, M. (1979): Combinatorial Programming: A Pattern Recognition Technique Approach, a PhD thesis, REC Warangal, India (Unpublished).

[23] Venkata Ramana V.V. Combinatorial Programming Problems. M.Phil Thesis, S.V.U. College of Engg. Tirupati, A.P., India, 1986.

AUTHORS PROFILE

Dr. S. Purusotham is currently working as a Lecturer, Department of Mathematics, S. V. J. College, TTD, Tirupati, received the doctoral degree in Mathematics from Sri Venkateswara University, Tirupati Andhra Pradesh, India.

Dr. M. Sundara Murthy is currently working as professor in the Department of Mathematics, Sri Venkateswara University, Tirupati, India.