Available online at http://scik.org

J. Math. Comput. Sci. 11 (2021), No. 3, 3436-3463

https://doi.org/10.28919/jmcs/5679

ISSN: 1927-5307

LEXI-SEARCH ALGORITHM USING PATTERN RECOGNITION TECHNIQUE FOR SOLVING A VARIANT CONSTRAINED BULK TRANSSHIPMENT PROBLEM

B. MAHABOOB*, J. PETER PRAVEEN, V.B.V.N. PRASAD, T. NAGESWARA RAO,

G. BALAJI PRAKASH, D. SATEESH KUMAR

Department of Mathematics, Koneru Lakshmaiah Education Foundation,

Deemed to be University, Green Fields, Vaddeswaram, Guntur-522 502, India

Abstract: In this article we present a variant transportation problem called "A Variant Constrained Bulk Transshipment Problem". The purpose of this paper is to propose an efficient Lexi-Search Algorithm using pattern recognition technique for solving "A Variant Constrained Bulk Transshipment Problem" on a scalable multicomputer platform and to obtain an optimal solution. Our results show that the proposed algorithm is highly competitive on a set of benchmark problems. The objective of the problem is to minimize the total bulk cost of supplying the required products to the destinations with the restriction that any destination should get its requirement from one source only, even when it gets from a destination. In the sequel we developed a Lexi-Search algorithm based "Pattern Recognition Technique" to solve this problem which takes care of simple combinatorial structure of the problem and computational results are reported. In this discussion we have studied a variation of the transportation problems called "A Variant Constrained Bulk Transshipment Problem". It also comes under combinatorial programming problems.

Keywords: bulk-transportation cost; fixed replenishment costs; destinations; sources; total bulk cost; feasible solution.

2010 AMS Subject Classification: 65Y10.

---

*Corresponding author

E-mail address: bmahaboob750@gmail.com

Received March 11, 2021

1. 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. Gendreau, Laporte and Potvin [1] in their article discussed the VRP monographs on discrete mathematics and applications. In 2006 Hollis, Forbes and Douglas [2] in their paper depicted vehicle routing and crew scheduling for metropolitan mail distribution at Australia post. An exact algorithm for the travelling salesman problem with deliveries and collections was presented by Baldacci, Hadjiconstantinou and Mingozzi [3] in the year 2003.Gupta, Verma and Puri [4] ,presented Time-cost trade-off relations in bulk transshipment problem.Borovska,Lazarova and Bahudejla [5] proposed strategies for parallel genetic computations of optimization problems.In 1995 Abuali,Wainwright and Schoenefeld[6] introduced determinant factorization in which a new coding scheme for spanning trees applied to the probabilistic minimum spanning tree problem was discussed.Here,Tzur and Yucesan[7] in their article proposed the multi-location transshipment problem. More significant work regarding Lexi-search approach to travelling salesman problem was done by Vijayalakshmi [8].Sekhar, Balakrishna and Purushotham[9] in their paper proposed a pattern recognition Lexi search approach to travelling salesman problem with additional constraints.Ahmed[10] presented a data guided lexi-search algorithm for the bottleneck travelling salesman problem.Arora,Puri [11] proposed a lexi-search algorithm for a time minimizing assignment problem.Kumar,Kaur and Gupta[12] in 2011,in their paper introduced new methods for solving fuzzy transpotation problems with some additional transshipments .Kek,Cheu and Meng [13] in their research article discussed distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots.Archetti,Speranza and Hertz[14] presented a tabu search algorithm for the split delivery vehicle routing problem

Research efforts have generally viewed transshipments as an emergency recourse when unexpected circumstances have caused a surplus at one location and a shortage at another. One reason for considering only this reason for transshipments is the general lack of consideration of

fixed replenishment costs. When these costs are present, we may want to replenish at one location and transship items to another location, in order to save on the fixed costs. Another reason for transshipments which was not discussed previously in the literature is to save on the holding costs, exploiting cases where different locations have different holding costs. (The latter reason may indeed be more important in settings where fixed replenishment costs exist, as in such cases more inventories is held).

Suresh Babu (2013) studied the problem called A Variant Bulk Transportation Problem with Multiple Bulk Cost Constraint. In the bulk transportation problem C (i, j) is the cost of transport of the requirement of the destination J (j) from I (i) and it is independent of the units of the products, hence it is called the bulk cost. Hence in the pattern X (i, j)=0 or 1; if it is 1 it means the source i is supplying destination j, otherwise 0.i.e, X (i, j) takes 0 or 1 values. When quantities are large this model is not considered because C (i, j) will be bulk cost only but the quantity is fixed at say α, as a result C (i, j) will be the bulk cost which is one bulk unit α supplying from source i to destination j. if the destination j requires kα. i.e., k times the bulk unit and it can be supplied from source i subjected to the availability, then the cost is k×C (i, j), then X (i, j) = k. In many practical cases k = 1, 2 or 3. If k is more than 3 then the source person is will supply to extra quantity freely, because of competition. As a result in many cases k is restricted to some finite number 1, 2, 3 or 4. The model where X (i, j) can take 1, 2, 3 or 4 is more practical and useful. So this model we call it as A Variant Bulk Transportation Problem with Multiple Bulk Cost Constraint.

2. PROBLEM DESCRIPTION

In this paper we have studied a variation of the transportation problems called “A Variant Constrained Bulk Transshipment Problem”. It also comes under combinatorial programming problems. Here the products are supplied simultaneously to the destinations according to their requirement from the sources. The cost of transportation or Transshipment of products from the sources to destinations is given. Objective of the problem is to minimize the total bulk

transshipment cost subjected to the availability and requirement constraints. The Classical Transshipment 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 usual transshipment consist a unit cost for supplied goods to destinations from the sources. But in bulk transshipment the cost is independent of number of goods supplied to destinations, it is practical. In this paper we investigated a "variant of constrained bulk transshipment problem". Let there are m-sources and n-destinations. The destinations can get its complete requirement from a source directly or through some destination. The practical constraint is considered as only fewer destinations are allowed to supply its availability to some limited destinations. The cost of transportation of products from the sources to destination and destinations to destination is given. In this problem we take care of the restriction of availability and requirement of product between source and destinations. i.e., the total availability of the product at the source is greater than or equal to the total requirement of the product at the destinations. Generally movement of a product from source to source or destinations to source is not natural or practical, hence these possibilities are avoided and movement from destinations to destination is only considered. This is more generalized problem and comes under combinatorial programming problem. Often, the model is expressed as a zero-one programming problem. The objective of the problem is minimize the total bulk cost of supplying the required products to the destinations with the restriction that any destination should get its requirement from one source only, even when it gets from a destination.

In this A Variant Constrained Bulk Transshipment Problem, in general any source i supply its product to destinations subject to its availability once. There is a set of *S= {1, 2, 3,....., m}* sources which produces a particular product and set of$D = \{1, 2, 3,.....,n\}$ destinations. The requirement of place j ϵ D is DR (j) and the capacity of the source i ϵ S is SA (i). Let $D^1=\{\propto_1, \propto_2, \propto_3,\ldots\ldots,\propto_k\}$ where k<n(i.e.,$D^1 \subset D$)be the sub set of destinations, which supply the product to destinations subject to its availability of product. Let S^1 be a set of effective sources including destinations (D^1) because we allowed supply the product from destination to destination also i.e., $S^1=SUD^1=\{1, 2, 3,\ldots\ldots\ldots,m^1\}$, where $m^1=m+k$. The cost of bulk

transshipment from source *i* to destination *j* is C(i,j); i ϵ S^1,jϵD.The objective is to minimize the total bulk transshipment cost subjected to the availability and requirements of constraints. The model can be built as 0 or1 programming problem.

3. MATHEMATICAL FORMULATION

$$Minimize \quad Z = \sum_{i \in S^1} \sum_{j \in D} D(i, j)X(i, j) \qquad \text{---------- (1)}$$

Subject to the constraints:

$$\sum_{i \in S^1} X(i, j) = 1, \qquad j \in D \qquad \text{---------- (2)}$$

$$\sum_{i \in S^1} \sum_{j \in D} X(i, j) = n \qquad \text{---------- (3)}$$

$$\sum_{i=1}^{m} SA(i) \geq \sum_{j=1}^{n} DR(j) \qquad \text{---------- (4)}$$

$$X(i, j) = 0 \text{ or } 1 \quad \forall i \in I,\ j \in J \qquad \text{---------- (5)}$$

The constraint (1) describes the minimization of the total bulk transshipment cost subjected to the constraints. The constraint (2) represents that a destination should get its complete requirement exactly once (i.e., from a source or via shipment node).The constraint (3) indicates the supply schedule to the n destinations. The constraint (4) represents that the sum of the requirements at different destinations should less than or equal to the sum of availabilities at various sources. The last constraint (5) indicates that if there is a transportation from i to j then *X (i, j) = 1,* Otherwise it is equals to 0.

In the sequel we developed a Lexi-Search algorithm based on the “Pattern Recognition Technique” to solve this problem which takes care of simple combinatorial structure of the problem.

4. NUMERICAL FORMULATION

The concepts and the algorithm developed will be illustrated by a numerical example. In which we have taken number of Sources as m=4 i.e., S={1,2,3,4}and number of Destinations as n=7 i.e., D={1,2,3,4,5,6,7}. Let SA is the availability of a product at sources and DR is the requirement of a product at the destinations. Let D^1={1,4} and S^1 be the number of sources including destinations (D^1) because the transshipment problem allow supplying product from destination to destination also. So the total number of sources will be increase to 6(4+2).

But in this numerical example we consider, only two destinations can supply the product to the destinations subject to its availability of product. Let D^1={1, 4} be the set of destinations, which supply the product to the destinations subject to its availability of product. This is subset of set D. Hence the total number of sources increases from 4 to 6. Let S^1 be the total number of sources such that S^1=SUD^1= {1,2,3,4,5,6}.Where $S^1(5)=D^1(1)$ and $S^1(6)=D^1(4)$. Then the cost matrix D is given below in **Table-1**. (For convenience same notation D is taken for the distance matrix).

Table- 1

D(i,j)=	1	2	3	4	5	6	7	SA
1	1	17	24	21	30	41	8	**120**
2	18	3	12	11	47	16	21	**100**
3	2	1	20	5	15	7	44	**90**
4	6	4	17	28	39	32	2	**80**
5	-	19	5	23	4	54	59	**–**
6	27	13	49	-	50	6	3	**–**
DR	**40**	**30**	**35**	**45**	**30**	**45**	**50**	

In the above numerical example given in **Table – 1**, D (i, j), where D (i, j)'s taken as non negative integers it can be easily seen that this is not a necessary condition and the cost can well be any positive quantity. Suppose D (3, 4) = 5 means that the cost of the product supply from

source 3 to destination 4 is 5. Similarly D (6, 7) = 3 means that the cost of the product supply from source 6 (i.e., destination 4) to destination 7is 3.

5. FEASIBLE SOLUTION

Consider an ordered pair set {(1,1), (3,2), (4,7), (5,5), (3,4), (5,3), (2,6)} represents a feasible allocation and gives the feasible solution. In the following **figures-1,** shape of cylinder represents the sources and the boxes of rectangular shapes represent destinations. The values at cylinders represent the capacity of source availability and the values at boxes represent the requirements of destinations. The values at arrows indicate that the bulk cost from the corresponding source i to destination j. Then the **figure – 1** represent the feasible solution as follows.

In the above solution, source 1 supplies its product to destination 1. Source 5 (destination 1) supplies its product to the destinations 5 and 3. Source 3 supplies its product to the destinations 2 and 4. Source 4 supplies its product to the destinations 7 and source 2 supplies its product to the destinations 6.So the solution gives the feasible solution.

Then the total bulk transshipment cost from given 6 sources to 7 destinations with respective source is as follows.

Total cost Z = D (1, 1)+D (3, 2)+D (4, 7)+D (5, 5)+D (3, 4)+D (5, 3)+D (2, 6)

= 1 + 1 + 2+4+5+5+16=34 units.

6. SOLUTION PROCEDURE

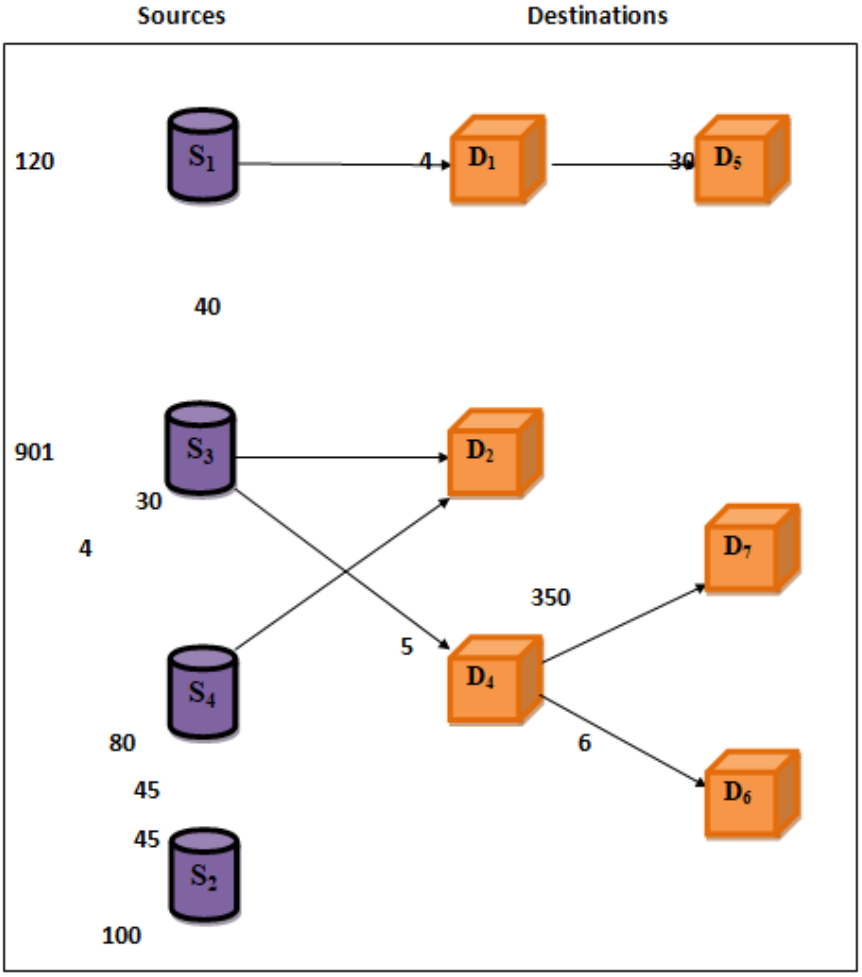
In the above figure-1, for the feasible solution we observe that 7 ordered pairs are taken along with the values from the cost/distance matrix for the numerical example in Table- 1. The 7orderedpairs are selected such that they represent a feasible solution in figure-1. So the problem is that we have to select 7 ordered pairs from the cost matrix order [6×7] along with values such that the total cost is minimum and represents a feasible solution. For this selection of 7 ordered pairs we arrange all the ordered pairs with the increasing cost and call this formation as alphabet

table and we developed an algorithm for the selection along with checking for the feasibility.

7. INFEASIBLE SOLUTION

Consider an ordered pair set {(1,1), (3,2), (6,7), (4,2), (5,5), (3,4), (6,6)} represents an infeasible allocation and gives the infeasible solution.

Figure –2 (INFEASIBLE SOLUTION)



From the above figure-2, source 1 supplies its product to destination 1. The destination 2 gets its required product from source 3. Source 6 (i.e., destinsation4) supplies its product to destination 7. Source 4 supplies its product to destination 2. But destination 2 already satisfied by source 3. Source 5 (i.e., destination1) satisfies the requirement of destination 5. Source 3 supplies its

product to destination 4. But the total sum of the requirement of the destinations 2, 4 and 7 is greater to the availability of the source 3, i.e., the total amount of supply is greater than actual amount of availability of source 3. Source 6 (i.e., destination 4) supplies its product to destination 6. Here the all destinations do not satisfy with the respective requirements by the above allocation. So the solution gives the infeasible solution. The cost for the above mentioned ordered pairs are

Total cost Z = D (1, 1) +D (3, 2)+D (6, 7)+D (4, 2)+D (5, 5)+D (3, 4)+D (6, 6)

= 1 + 1 +3+4 + 4+5+6

= 24 units.

8. CONCEPTS AND DEFINITIONS

8.1. Definition of a Pattern

An indicator two dimensional arrays *X* which is associated with an allocation is called a "pattern". A pattern is said to be feasible if *X* is a feasible solution.

$$V(X) = \sum_{i \epsilon S^1} \sum_{j \epsilon D} D(i,j) X(i,j)$$

The value V(x) is gives the total cost of the tour for the solution represented by X. The pattern represented in the **Table-2** is a feasible pattern. The value V(X) gives the total transportation cost for the solution represented by X. Thus the value of the feasible pattern gives the total cost 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 pairs [(i,j)] for which X (i, j)=1 with understanding that the other X(i,j)'s are zeros.

Consider the set of ordered pairs {(1, 1), (3, 2),(4, 7), (5,5), (3, 4), (5, 3), (2,6)} represented by 0 or 1 in matrix X(i, j) indicates the pattern. Here **Table –2**denotes above pattern which is a feasible solution. According to the pattern represented in **figure-1,** satisfies all the constraints in Mathematical Formulation.

The **Table-2** represents a feasible pattern of the feasible solution. In the above solution **X (2, 6) = 1**, represents that source 2 supplies its product to the destinations 6. In similar way **X (4, 7) = 1**, represents source 4 supplies its product to the destinations 7. Similarly all destinations can get its complete requirement from source directly or via some destination. So the above solution gives a feasible solution and it shown in **figure-1.**

Table –2

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

The pattern in **Table-3,** gives an infeasible solution. The ordered pair set {(1, 1), (3, 2),(6, 7), (4, 2), (5, 5), (3, 4), (6, 6)}represents a pattern which is an infeasible solution given below.

Table –3

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

The ordered pair set {(1, 1), (3, 2),(4, 7), (5, 5), (3, 4), (5, 3), (2, 6)} represents the pattern in **Table-2**, which is feasible solution and the ordered pair set {(1, 1), (3, 2), (6,7), (4, 2), (5, 5), (3, 4), (6, 6)} represents the pattern in **Table –3**, which is an infeasible solution .

8.2. Alphabet Table

There are $m^1 \times n$ ordered pairs in the two-dimensional array D. For convenience these are arranged in ascending order of their corresponding distance and are indexed from 1 to mxn (Sundara Murthy-1979). Let SN= [1, 2, 3…$m^1 \times n$] be the set of m×n indices. Let D be the corresponding array of distance. If a, b ∈ SN and a < b then D(a) $\leq_D$ (b). Also let the arrays R, C be the array of row and column indices of the ordered pair represented by SN. The arrays SN, D, R, and C for the numerical example are given in the **Table–4**. If a ∈ SN then (R (a), C (a)) is the

ordered pair and D (a) = D(R (a), C (a)) is the value of the ordered pair and$DC\ (a) = \sum_{i=1}^{a} D(i)$. For convenience same notation D is used for cost matrix and the increment array of cost in alphabet table. Then the alphabet of the given cost matrix (**Table –1**) is as follows.

Table-4: Alphabet Table

SN	D	DC	R	C
1	1	1	1	1
2	1	2	3	2
3	2	4	3	1
4	2	6	4	7
5	3	9	2	2
6	3	12	6	7
7	4	16	4	2
8	4	20	5	5
9	5	25	3	4
10	5	30	5	3
11	6	36	4	1
12	6	42	6	6
13	7	49	3	6
14	8	57	1	7
15	11	68	2	4
16	12	80	2	3
17	13	93	6	2
18	15	108	3	5
19	16	124	2	6
20	17	141	1	2
21	17	158	4	3

22	18	176	2	1
23	19	195	5	2
24	20	215	3	3
25	21	236	1	4
26	21	257	2	7
27	23	280	5	4
28	24	304	1	3
29	27	331	6	1
30	28	359	4	4
31	30	389	1	5
32	32	421	4	6
33	39	460	4	5
34	41	501	1	6
35	44	545	3	7
36	47	592	2	5
37	49	641	6	3
38	50	691	6	5
39	54	745	5	6
40	59	804	5	7

From the above **Table – 2**, Let us consider $14 \in SN$. It represents the ordered pair D(R (14), C (14)) = (1, 7). Then D (14) = D (1, 7) =8. i.e., the transportation cost from source 1 to destination 7 and DC (14) =57.

8.3. Definition of a Word

Let SN = (1,2,…) be the set of indices, D be an array of corresponding distances of the ordered pairs and Cumulative sums of elements in D is represented as an array DC. Let arrays R,

C be respectively, the row, column indices of the ordered pairs. Let $L_k = \{a_1, a_2, - - -- - , a_k\}$, $a_i \in$ SN be an ordered sequence of k indices from SN. The pattern represented by the ordered pairs whose indices are given by L_k is independent of the order of a_i in the sequence. Hence for uniqueness the indices are arranged in the increasing order such that $a_i \leq a_{i+1}$, i = 1, 2, - - - -, k-1. The set SN is defined as the "Alphabet-Table" with alphabetic order as (1, 2, - - - -,$m^1 \times n$) 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 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 pattern. 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. In the partial word L_k any of the letters in SN can occupy the first place. Since the words of length greater than n are necessarily infeasible, as any feasible pattern can have only n unit entries in it. L_k is called a partial word if k < n, and it is a full length word if k = n, 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 at least one feasible word.

8.4. 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).

For example the word L_5 = (1, 2, 4, 8, 9) then value of L_5 is V (L_5) = V (L_4) + D (a_5),

V (L_4) = 8and a_5=9then V (L_5) = 8 + D (9) = 8+5=13

8.5. 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_3, ……,a_k)can be defined as follows LB (L_k) = V (L_k) + DC (a_k + n- k) – DC (a_k)

Consider the partial word $L_5 = (1, 2, 4,8,9)$, $V(L_5) = 13$ then

$$LB(L_5) = V(L_5) + DC(a_5 + 7\text{-}5) - DC(a_5)$$

$$= V(L_5) + DC(9+2) - DC(9)$$

$$= 13 + 36 - 25 = 24$$

8.6. Feasibility Criterion of a Partial Word

A feasibility criterion is developed, in order to check the feasibility of a partial word $L_{k+1} = (a_1, a_2 \ldots a_k, a_{k+1})$ given that L_k is a feasible word. We will introduce some more notations which will be useful in the sequel.

IC: An array where IC (j) = 1, $j \epsilon J = \{1, 2, \ldots\ldots, n\}$ represents that j^{th} type of destinations are getting its requirements from some sources, otherwise IC (j) = 0.

L: An array where L (*i*) is the letter in i^{th} position of the word, L(i)=0.

The values of the arrays IC and L are as follows

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

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

For example consider a sensible partial word $L_4 = (1, 2, 4, 9)$ which is feasible. The array DR, SA, L and IC takes the values represented in **Table –5** given below.

Table – 5

	1	**2**	**3**	**4**	**5**	**6**	**7**
L	1	2	4	9	-	-	-
IC	1	1	0	1	0	0	1

The recursive algorithm for checking the feasibility of a partial word L_p is given as follows. In the algorithm 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_{k+1})** and **CA = C (a_{k+1}).**

9. ALGORITHMS

9.1. Algorithm 1 (checking for feasible)

STEP0 : IX = 0 GO TO1

STEP1 : IS (IC [CA] = 1) IF YES GO TO 13
IF NO GO TO 2

STEP2 : IS (RA > m) IF YES GO TO 3
IF NO GO TO 8

STEP3 : IS (Count [RA]>Q) IF YES GO TO 13
IF NO GO TO 4

STEP4 : RD = DRS [RA] + DR [CA]
IS (IC [SS [RA]] = 1) IF YES GO TO 5
IF NO GO TO 6

STEP5 : IS (RD ≤ (SA [SW [SS[RA]]]) IF YES GO TO 9
IF NO GO TO 13

STEP6 : DRS [RA] = RD IF YES GO TO 7

STEP7 : IS (IC [SS [RA]] =0) IF YES
{DR [SS [RA]] =DR [SS [RA]] +DRS [RA] GOTO 12}
IF NO GO TO 8

STEP8 : IS (DR [CA]≤ SA [RA]) IF YES GO TO 9
IF NO GO TO 13

STEP9 : IS (I=n) IF YES GOTO 12
IF NO GOTO 10

STEP10 : IS (RA>m)
IF YES {SA [SW [SS [RA]]] = SA [SW [SS [RA]]]-DR [CA]
GOTO 12}

STEP11 : SA [RA] =SA [RA]-DR [CA] GOTO 12

STEP12 : IX = 1 GOTO 13

STEP13 : STOP

We start with 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 chain formulation. We will calculate the values of V (L_k) and LB (L_k) simultaneously. Then two situations arises one for branching and 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 under (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 p by taking another letter which succeeds a_k in the position. If all the words of order p are exhausted then we consider the next partial word of order (k-1).
2. LB (L_k) $\geq$ 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 p that succeeds L_k.

To find Optimal Feasible word a Lexi-Search Algorithm using PRT is developed and is given in below.

9.2. Algorithm 2 (Lexi-search Calculation)

STEP0 : Initialization

The arrays SN, IC, R, C, SS, DRS, SW, SA, DR, DC, RA, CA, L, V, LB and values Q, m, n, k are made available. The values I=1, J=0, VT=9999 and Max= m×n–n

STEP1 : J = J + 1

IS J > Max IF YES GOTO 9

IF NO GOTO 2

STEP2 : V [I] = V [I-1] + C[J]; V [0] = 0

LB [I] = V [I] + DC [J + n – I] – DC [J]

IS (LB [I] ≥ VT) IF YES GO TO 9

IF NO GO TO 3

STEP3 : RA = IR [J], CA = IC [J] GOTO 4

STEP4 : Check Feasibility using algorithm 1
IS (IX = 1) IF YES GO TO 5
IF NO GO TO 1

STEP5 : IS (I = n) IF YES GO TO 8
IF NO GO TO 6

STEP6 IC [CA] = 1
SW [CA] = RA
L [I] = J
Count [RA] = count [RA] + 1 GOTO 7

STEP7 : I = I + 1
Max = Max + 1 GOTO 1

STEP8 : VT = V [I], L [I] = J, L [I] is full length word and is feasible
Record L [I] and VT GO TO 10

STEP9 : IS (I = 1) IF YES GO TO 12
IF NO GO TO 10

STEP10 : I = I – 1
J = L [I],RA = R [J], CA = C [J]
IC [CA] = 0
SW [CA] = 0
Count [RA] =count [RA] -1;
L [J] = 0;
IS (RA > m) IF YES GOTO 11 IF NO
{SA [RA] =SA [RA] + DR [CA] GOTO 11}

STEP11 : IS (SW [SS [RA]] ≠ 0) IF YES
{SA [SW [SS [RA]]] = SA [SW [SS [RA]]] + DR [CA] GOTO 12}
IF NO GOTO 12

STEPP12 : IS (DRS [RA] ≠0)
IF YES {DR [SS [RA]] =DR [SS [RA]]-DRS [RA]
DRS [RA] =DRS [RA]-DR [CA]}GOTO 11

STEP13 : SW [CA] = 0 GOTO 1

STEP14 : STOP

The current value of VT at the end of the search is the value of the optimal feasible word. At the end if VT = 999 it indicates that there is no feasible solution.

10. 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-6**. The columns (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 next two columns V and LB are indicate the value and lower bound of the respective partial word. The column R and C gives the row and column indices of the letter. The last column gives the remarks regarding the acceptability of the partial words (i.e. if a partial word is feasible word then accept the letter otherwise reject the letter) and here A indicates the acceptance and R for rejectance of the letter in the respective position.

Table-6: Search table

SN	1	2	3	4	5	6	7	V	LB	R	C	Remark
1	1							1	16	1	1	A
2		2						2	16	3	2	A
3			3					4	16	3	1	R
4			4					4	18	4	7	A
5				5				7	18	2	2	R
6				6				7	20	6	7	R
7				7				8	22	4	2	R
8				8				8	24	5	5	A
9					9			13	24	3	4	A
10						10		18	24	5	3	A
11							11	24	24	4	1	R
12							12	24	24	6	6	R
13							13	25	25	3	6	R

14							14	26	26	1	7	R
15							15	29	29	2	4	R
16							16	30	30	2	3	R
17							17	31	31	6	2	R
18							18	33	33	3	5	R
19							19	34	34	2	6	A, VT=34
20						11		19	25	4	1	R
21						12		19	26	6	6	R
22						13		20	28	3	6	R
23						14		21	32	1	7	R
24						15		24	36	2	4	R, >VT
25					10			13	25	5	3	A
26						11		19	25	4	1	R
27						12		19	26	6	6	A
28							13	26	26	3	6	R
29							14	27	27	1	7	R
30							15	30	30	2	4	A, VT=30
31						13		20	28	3	6	A
32							14	28	28	1	7	R
33							15	31	31	2	4	R,>VT
34						14		21	32	1	7	R, >VT
35					11			14	27	4	1	R
36					12			14	29	6	6	A
37						13		21	29	3	6	R
38						14		22	33	1	7	R, >VT
39					13			15	34	3	6	R, >VT
40				9				9	26	3	4	A
41					10			14	26	5	3	A
42						11		20	26	4	1	R
43						12		20	27	6	6	R
44						13		21	29	3	6	R
45						14		22	33	1	7	R, >VT

46					11			15	28	4	1	R
47					12			15	30	6	6	R, =VT
48				10				9	28	5	3	A
49					11			15	28	4	1	R
50					12			15	30	6	6	R, =VT
51				11				10	31	4	1	R, >VT
52			5					5	21	2	2	R
53			6					5	23	6	7	A
54				7				9	23	4	2	R
55				8				9	25	5	5	A
56					9			14	25	3	4	R
57					10			14	26	5	3	A
58						11		20	26	4	1	R
59						12		20	27	6	6	A
60							13	27	27	3	6	R
61							14	28	28	1	7	R
62							15	31	31	2	4	R, >VT
63						13		21	29	3	6	A
64							14	29	29	1	7	R
65							15	32	32	2	4	R, >VT
66						14		22	33	1	7	R, >VT
67					11			15	28	4	1	R
68					12			15	30	6	6	R, =VT
69				9				10	27	3	4	R
70				10				10	29	5	3	A
71					11			16	29	4	1	R
72					12			16	31	6	6	R, >VT
73				11				11	32	4	1	R, >VT
74			7					6	26	4	2	R
75			8					6	28	5	5	A
76				9				11	28	3	4	A

77					10			16	28	5	3	A
78						11		22	28	4	1	R
79						12		22	29	6	6	R
80						13		23	31	3	6	R, >VT
81					11			17	30	4	1	R, =VT
82				10				11	30	5	3	R, =VT
83			9					7	31	3	4	R, >VT
84		3						3	19	3	1	R
85		4						3	22	4	7	A
86			5					6	22	2	2	A
87				6				9	22	6	7	R
88				7				10	24	4	2	R
89				8				10	26	5	5	A
90					9			15	26	3	4	A
91						10		20	26	5	3	A
92							11	26	26	4	1	R
93							12	26	26	6	6	A, VT=26
94						11		21	27	4	1	R, >VT
95					10			15	27	5	3	R, >VT
96				9				11	28	3	4	R, >VT
97			6					6	24	6	7	R
98			7					7	27	4	2	R, >VT
99		5						4	25	2	2	A
100			6					7	25	6	7	A
101				7				11	25	4	2	R
102				8				11	27	5	5	R, >VT
103			7					8	28	4	2	R, >VT
104		6						4	28	6	7	R, >VT
105	2							1	19	3	2	A
106		3						3	19	3	1	A
107			4					5	19	4	7	A
108				5				8	19	2	2	R

109				6				8	21	6	7	R
110				7				9	23	4	2	R
111				8				9	25	5	5	R
112				9				10	27	3	4	R, >VT
113			5					6	22	2	2	R
114			6					6	24	6	7	A
115				7				10	24	4	2	R
116				8				10	26	5	5	R, =VT
117			7					7	27	4	2	R, >VT
118		4						3	22	4	7	A
119			5					6	22	2	2	R
120			6					6	24	6	7	R
121			7					7	27	4	2	R, >VT
122		5						4	25	2	2	R
123		6						4	28	6	7	R, >VT
124	3							2	23	3	1	A
125		4						4	23	4	7	A
126			5					7	23	2	2	A
127				6				10	23	6	7	R
128				7				11	25	4	2	R
129				8				11	27	5	5	R, >VT
130			6					7	25	6	7	R
131			7					8	28	4	2	R, >VT
132		5						5	26	2	2	R, =VT
133	4							2	26	4	7	R, =VT

11. COMMENTS

The shaded rows in the above **Table-6**, gives optimal solution of the taken numerical example and at the end of the search the current value of **VT** is **26.** Then the partial word is $\mathbf{L_7 = (1, 4, 5, 8,}$

9,10, 12) is an optimal feasible word. It is given in the **93rd row** of the search table. For this word the arrays L, DR, SA and IC are given in the following **Table – 7**.

Table –7

	1	2	3	4	5	6	7
L	1	4	5	8	9	10	12
IC	1	1	1	1	1	1	1

At the end of the search table the optimum solution value of **VT** is **26** and is the value of optimal feasible word $\mathbf{L_7}$ **= (1, 4, 5, 8, 9, 10, 12)**. Then the following

Figure –3

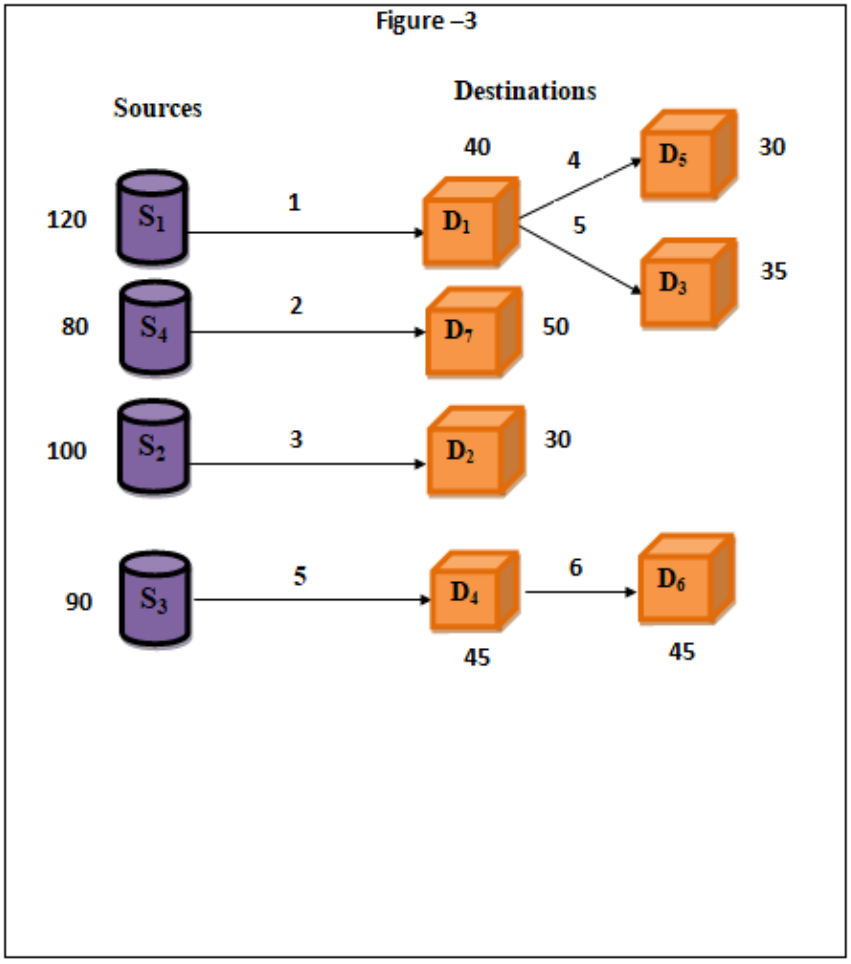


Figure-3 represents the optimal solution to the problem.

In the above **figure-3**, source 1 supplies its product to destination 1. Source 5 (destination 1) supplies its product to the destinations 5 and 3. Source 3 supplies its product to destination 4. Source 6 (destination 4) supplies its product to the destinations 6. Destination 7 and destination 2 gets its requirement of the product from the source 4 and source 2 respectively. The transshipment cost for the above mentioned ordered pairs are

Total cost Z = D (1, 1) + D (4, 7) + D (2, 2) + D (5, 5) + D (3, 4) + D (5, 3) + D (6, 6)

= 1 + 2 + 3 + 4 + 5 + 5 + 6 = 26 units.

Consider the set of ordered pairs {(1, 1) , (4, 7), (2, 2), (5, 5), (3, 4), (5, 3), (6, 6)} represented the pattern given in the **Table-8**, which is a feasible solution. According to the pattern represented in **figure-3**, satisfies all the constraints in Mathematical Formulation.

Table-8

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

From the above optimal feasible solution the total bulk transshipment cost from given sources to 7 destinations as follows.

Total cost = D (1, 1) + D (4, 7) + D (2, 2) + D (5, 5) + D (3, 4) + D (5, 3) + D (6, 6)

= 1+2+3+4+5+5+6 = 26 units.

12. EXPERIMENTAL RESULTS

A Computer program for the proposed Lexi – Search Algorithm is written in C Language and is tested at various hard instances. The experiments are carried out on a COMPAQ (dx2280 MT) system. The inputs like Cost Matrix D (i, j), source capacities (SA), and destination requirements (DR) are randomly generated for different instances. The cost values are uniformly generated in the interval [1, 100]. The values for the availability and requirement of the product

are also randomly generated between [1, 500]. For different values of m, k, n, and Q a set of problems have been tested and their computational run time is recorded in seconds. The obtained results are tabulated in **Table -9**. It is observed that, the time required for the search of the optimal solution is fairly less. In the following table microseconds are represented by zero.

In **Table-9**, SN = serial number, m = number of sources, *k* = number of destinations which are acting as shipment nodes, n = number of destinations. Q = the number of times can supply the requirement of destinations by a destinations which are acting as shipment nodes. AT = CPU run time for formation of the alphabet table, ST=CPU run time for searching an optimal solution. It is seen that time required for the search of the optimal solution is moderately less.

Table-9

SN	m	k	N	Q	Trail Solution	Optimal Solution	CPU RUN TIME
							AT+ST
1	4	2	7	2	200	26	0.000000
2	5	2	8	2	200	58	0.000000
3	5	3	10	2	200	103	0.000000
4	8	2	10	2	200	73	0.000000
5	8	3	12	2	200	123	0.000000
6	10	3	10	3	300	55	0.000000
7	10	5	15	3	300	104	0.109890
8	10	5	20	3	300	147	0.109890
9	15	5	25	3	300	141	0.274725
10	20	3	25	3	300	106	0.274725
11	20	5	30	3	400	113	0.384615
12	25	8	40	3	400	175	0.549445
13	30	5	40	4	400	135	0.604396
14	30	5	50	4	400	187	0.659341
15	40	10	50	4	400	125	0.879121
16	45	8	60	5	500	166	0.824176
17	50	10	55	5	500	109	0.934066
18	55	5	70	5	500	159	1.043956
19	60	10	70	6	500	140	1.153460
20	70	5	80	6	500	140	1.153460

13. COMPARISON DETAILS

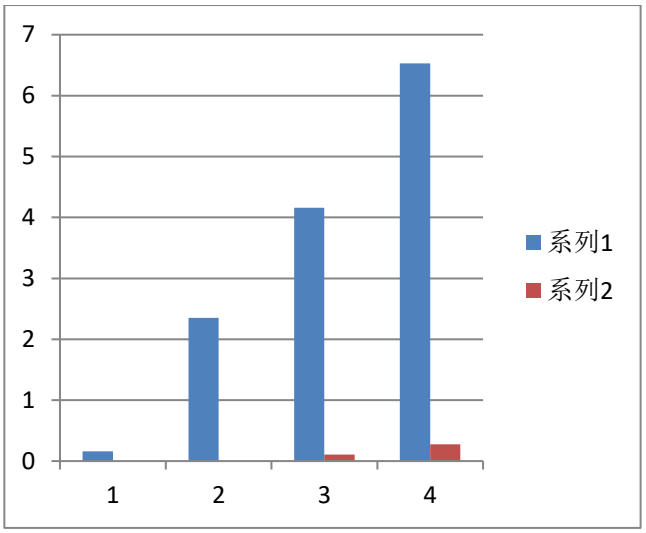
We implement the Lexi-Search algorithm using Pattern Recognition Technique based with C language for this model. We tested the proposed algorithm by different set of problems and compared the computational results with the published Time Dependent Bulk Transportation model in the studies of Nangana (2007) of Sri Venkateswara University; Tirupati. **Table-10** shows that the comparative results of different sizes. In the following table microseconds are represented by zero.

Table-10

S. No.	No. of Sources	No. of Destinations	Published Model	Proposed Model
1	4	7	0.16	0.000000
2	8	10	2.35	0.000000
3	10	15	4.16	0.109890
4	15	25	6.53	0.274725

In the above **Table-10**, last two columns show CPU run time of published model and proposed model. As compares the two models of sizes N=4, 8, 10&15. The runtime of this instance with the existing model are 0.16 sec, 2.35 sec, 4.16 sec & 6.53 sec and the proposed model took 0.0 sec, 0.0 sec, 0.109890 sec & 0.274725 sec, it is reasonably less time. The present model takes very less computational time for finding the optimal solution. Hence, suggested the present model for solving the higher dimensional problems. The graphical representation of the CPU run time for the two models presented in the above 4 instances is given below. In **Graph-1**, X axis taken the SN and Y axis taken the values of CPU run time for the published and proposed models. From **Graph-1,** series 1 represent that CPU run time for getting optimal solution **by published model (awarded)** and series 2 represent that CPU run time for searching the optimal solution by the proposed model. Also the proposed model takes less time than published model for giving the solution.

Graph-1



14. CONCLUSION

In the above discussion, we presented an exact algorithm called Lexi-search algorithm based on pattern recognition technique to solve the **A Variant Constrained Bulk Transshipment Problem**. Lexi-search algorithms are proved to be more efficient in many combinatorial problems. First the model is formulated into a zero-one programming problem. A Lexi-Search Algorithm based on Pattern Recognition Technique is developed for getting an optimal solution. The problem is illustrated with help of suitable numerical example in detail. We planned the proposed algorithm using C-language. The computational details are reported and compare with the published paper. As an observation the CPU run time is reasonably less for higher values of the problem to obtain an optimal solution than the published model. Based on this experience we strongly consider that this algorithm can perform larger size problems.

CONFLICT OF INTERESTS

The authors declare there is no conflict of interests.

REFERENCES

[1] M. Gendreau, G. Laporte, J.-Y. Potvin, Metaheuristics for the capacitated VRP, in: P. Toth, D. Vigo (Eds.), The vehicle routing problem, monographs on discrete mathematics and applications, SIAM, Philadelphia, (2002), pp. 129–154.

[2] B.L. Hollis, M.A. Forbes, B.E. Douglas, Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post, Eur. J. Oper. Res. 173 (2006), 133–150.

[3] R. Baldacci, E. Hadjiconstantinou, A. Mingozzi, An exact algorithm for the traveling salesman problem with deliveries and collections, Networks. 42 (2003), 26–41.

[4] A. Gupta, V. Verma, M.C. Puri, Time-cost trade-off relations in bulk transportation problem, J. Inform. Optim. Sci. 16 (1995), 317–325.

[5] P. Borovska, M. Lazarova, S. Bahudejla, Strategies for Parallel Genetic Computation of Optimization Problems, Biotechnol. Biotechnol. Equip. 21 (2007), 241–246.

[6] F.N. Abuali, R.L. Wainwright, D.A. Schoenefeld, Determinant factorization: a new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem, in: L.J. Eshelman (Ed.), Proceedings of the Sixth International Conference on Generic Algorithms, Morgan Kaufmann Publishers, San Francisco, California, (1995), pp. 470-475.

[7] Y.T. Herer, M. Tzur, E. Yücesan, The multi-location transshipment problem, IIE Trans. 38 (2006), 185–200.

[8] G. Vijaya Lakshmi, Lexisearch approach to travelling salesman problem, IOSR J. Math. 6(4) (2013), 01-08.

[9] K.C. Sekhar, U. Balakrishna, E. Purushotham, A pattern recognition lexi search approach to travelling salesman problem with additional constraints, Int. J. Computer Sci. Eng. 4 (2012), 307-320.

[10] Z.H. Ahmed, A data-guided lexisearch algorithm for the bottleneck travelling salesman problem, Int. J. Oper. Res. 12 (2011), 20-33.

[11] S. Arora, M.C. Puri, A lexi-search algorithm for a time minimizing assignment problem, Opsearch, 35 (1998), 193-213.

[12] A. Kumar, A. Kaur and A. Gupta, New methods for solving fuzzy transportation problems with some additional transshipments, ASOR Bull. 30 (2011), 42-61.

[13] A.G.H. Kek, R.L. Cheu, Q. Meng, Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots, Math. Computer Model. 47 (2008), 140–152.

[14] C. Archetti, M.G. Speranza, A. Hertz, A tabu search algorithm for the split delivery vehicle routing problem, Transport. Sci. 40 (2006), 64-73.