Available online at http://scik.org
J. Math. Comput. Sci. 11 (2021), No. 4, 4464-4496
https://doi.org/10.28919/jmcs/5683
ISSN: 1927-5307

A STUDY ON VEHICLE ROUTING PROBLEM WITH INTER LOADING FACILITIES

B. MAHABOOB[1,*], T. NAGESWARA RAO[1], V.B.V.N. PRASAD[1], Y. HARI KRISHNA[2],

P. VENKATESWARARAO[3], K. PRASAD[1]

[1]Department of Mathematics, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P, India

[2]Department of Mathematics, ANURAG Engineering College, Ananthagiri (v), Kodad, Suryapet, Telangana-508 206, India

[3]KL Business School, Koneru Lakshmaiah Education Foundation, Vaddeswaram,Guntur, India

Abstract: In this article, we discuss variant of the multi-depot vehicle routing problem where depots can act as intermediate replenishment facilities along the route of a vehicle. This problem is a generalization of the Vehicle Routing Problem (VRP). 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 paper, we consider a real-world problem with a loaded vehicle. The vehicle is available for pickup and delivery service. Single-commodity cargo, that is, same cargo type, is present at the customer sites. The vehicle is able to carry out a Pickup and delivery service of such cargo among the customers. By starting and ending at the origin depot, the vehicle follows a route without having a predefined sequence of Pickup and delivery services.

---

*Corresponding author
E-mail address: bmahaboob750@gmail.com
Received March 12, 2021

Keywords: vehicle routing problem; traveling salesman problem; Lexi-search algorithm; NP-hard; pattern recognition technique.

2010 AMS Subject Classification: 68Q25.

1. INTRODUCTION

As energy costs increases, driver shortages continue and hours of service regulations get tighter, transportation providers are motivated to use the capacity available to them as efficiently as possible. Excess capacity that is not utilized can result in declined service requests or additional costs, as vehicles that would otherwise not be necessary must be dispatched. Trucking firms are eager to eliminate any such occurrences, whether through finding loads for the backhaul portion of a route or consolidating loads from multiple customers onto a common vehicle. Improving capacity utilization through these methods is rather common. However, a more subtle extension of load consolidation is to allow multiple vehicles to service the same load. Splitting loads such that the delivery of certain loads is completed in multiple trips rather than one trip results in opportunities for a reduction in cost and the number of vehicles used. While this generally requires additional visits to a load's origin and destination, it may eliminate a dedicated trip to deliver the load by apportioning that load to other vehicles with excess capacity. Several studies have shown the benefit of split deliveries for the Vehicle Routing Problem (VRP), Dror et al. (1994), Frizzell and Giffin (1995), Sierksma and Tijssen (1998) Archetti et al. (2006) had revealed in which a vehicle operating out of a depot makes a series of deliveries on each route

In the year 2002 Gendreau, Laporte and Portvin [1], in their article presented mataheuristics for the capacitated VRP and proposed six main types of mataheuristics that have been applied to the VRP. In the year 2006 Hollis, Forbes and Douglas [2] presented a new multi-depot combined vehicle and crew scheduling algorithm and uses it in conjunction with a heauristic vehicle routing algorithm to solve the intra-city mail distribution problem faced by Australia post. Baldacci, Hadjiconstantinou and Mingozzi [3] in the year 2003, in their article described a new integer programming formulation for the travelling salesman problem with mixed deliveries and collections based on a two commodity network flow approach. Anju Gupta, Vanita Verma and Puri [4] in the year 1995, in their research article, studied special type of bicriteriabulk transportation problem involving the trading-off cost against time. In the year 2007, Borovska, Lazarova and Bahudejla [5], in their paper, discussed different strategies for parallel genetic computation of optimization problems. Abuali, Wainwright and Schoenefeld [6], in the year 1995, in their research article, described a new encoding scheme for the representation of spanning trees and this new encoding scheme is based on the factorization of the determinant of the in-degree matrix of the original graph. In the year 2006, Herer,Tzur and Yucesan [7], in their article, consider a supply chain which consists of several retailers and one supplier and demonstrated that how the values of the order-up to quantities can be calculated using a sample-path-based optimization procedure. In 2003, Vijayalakshmi [8], in her paper, introduced Lexiseaarch approach to the travelling salesman problem. In the year 2012, K.Chendra Sekhar[9], in his paper, depicted a problem recognition LexiSearch approach to travelling salesman problem with additional constraints. Zakir Hussain Ahmed [10], in the year 2011, first modified an existing Lexisearch algorithm by incorporating good upper and lower bounds to obtain exact optimal solution to the problem and then presented adata guided Lexisearch algorithm. Shalini Arora [11], in 2017, in their research article, studied time minimizing assignment problem also known as the bottleneck assignment problem and proposed a lexisearch approach to find an optimal feasible assignment of the facilities so as to minimize total time of completion of n’ out of n jobs. In 2011, Amit Kumar, Amarpreet Kaurand Anila Gupta [12], in their paper, proposed new methods for solving Fuzzy transportation problems with some additional transshipments. Kek, Chen and Meng [13], in 2008, in their research article, proposed two new distance-constrained capacitated vehicle rooting problems to investigate for the first time and study potential benefits in flexibly assigning start and end depots. In 2006,Archetti,Hertz and Speranza[14] described a tabu search algorithm for the vehicle rooting problem with split deliveries and in their discussion the insertion of a customer into a route is done by means of the cheapest insertion method.

The two main bodies of routing literature are relevant to this paper addressing the Vehicle Routing Problem (VRP) and the Pickup and Delivery Problem (PDP). The PDP is more relevant to the work presented here; however, split loads have been most extensively applied to the VRP. This special case of the VRP is most commonly referred to as the Split Delivery Vehicle Routing Problem (SDVRP), which occurs when a destination may be serviced by multiple vehicles. The PDPSL is a more complex problem than the SDVRP, primarily because the available capacity of the vehicle changes each time a load is picked up or delivered for the PDPSL, without the vehicle ever returning to a depot. The SDVRP load planning is done with the same fixed capacity prior to a vehicle leaving the depot. With the PDPSL, searching through each instance of available capacity to determine where to insert a split load is significantly more difficult, as all loads that are to be picked up or delivered in between the pickup and delivery of the load to be inserted must be accounted for in evaluating the capacity. Despite the differences, some of the approaches used for solving the SDVRP are applicable to the PDPSL, such as determining how to divide a load that is to be split and finding routing improvements. The PDP is a generalization of the Traveling Salesman Problem (TSP), making it NP-hard in the strong sense. This has focused most current research on heuristic methods.

In this paper, we consider a real-world problem with a loaded vehicle. The vehicle is available for pickup and delivery service. Single-commodity cargo, that is, same cargo type, is present at the customer sites. The vehicle is able to carry out a Pickup and delivery service of such cargo among the customers. By starting and ending at the origin depot, the vehicle follows a route without having a predefined sequence of Pickup and delivery services. This problem is called a one-commodity vehicle routing pickup and delivery problem (1-VRPPD). The main feature of 1-VRPPD is that delivery customers can be served with a cargo gathered from Pickup customers. This problem was first defined by Hernández-Pérez and Salazar-González (2004) as a one-commodity Pickup and delivery traveling salesman problem (1-PDTSP), which is the same as 1-VRPPD, when the number of vehicles is equal to 1. In general, the VRP problem and its variations are NP- hard. 1-VRPPD is NP- hard problem in the strong sense it specializes VRP.

The main VRP feature that differs from traveling salesman problem (TSP) is vehicle capacity constraint.

In this article, we discuss variant of the multi-depot vehicle routing problem where depots can act as intermediate replenishment facilities along the route of a vehicle. This problem is a generalization of the Vehicle Routing Problem (VRP). 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.

2. PROBLEM DESCRIPTION

In this paper we consider ‘Vehicle Routing Problem with Inter Loading Facilities’. There are some cities/stations available. Among them some of the cities act as sources including head quarter and remaining cities act as destinations. All sources have some availability of goods/commodity/load and all destinations have the requirements of goods. The vehicle starts from the head quarter with given load capacity and supply requirements of some destinations. If all the destination requirements are satisfied then the vehicle comeback to the head quarter city. Suppose the load/goods of a vehicle is low while supplying the destinations, then there is a facility that the vehicle fill the sufficient load by visiting the near source station and supply the destinations. Here the vehicle need not visit all source cities while supplying the destination requirements, but all the destinations must be satisfied. The availability of goods at the source cities are always greater than or equal to vehicle capacity. The aim of the problem is to find minimum distance/cost for satisfy the all destinations subject to the above conditions.

Let N be the set of n stations defined as N= {1, 2, 3, 4,……,., n} and here the city ‘1’ taken as the home city/head quarter city. Among them let S be the set of k sources including city 1 and defined as S= {α1, α2, α3 ,…., αk}. Let N1be the set of n-k destinations defined as N1= {β1, β2, β3 ,….,βn-k}. Let the requirement of destination j ϵ N1 is DR (j) and the capacity of the source i ϵ S is SC (i). Let the vehicle load capacity be ‘α’.The vehicle starts its tour from the home city (say 1) and come back to it after supplying the requirement of all n-k destinations. The vehicle may or may not visit all the ksource stations in its tour. While supplying the destination requirements, if the load of a vehicle is less than the requirement of destination requirements then the vehicle must visit the nearest source city to filling the load up to its capacity and supply the destination requirements. Hence the objective of the problem is to find a minimum total distance in its tour while completing all the destination requirements subject to the conditions. For this we developed an algorithm called as Lexi-Search algorithm using pattern recognition Technique.

3. MATHEMATICAL FORMATION

Main results.

Min (Z)≡ $\sum_{i\in N}\sum_{j\in N} D(i,j)X(i,j), S\cup N^1 = N$, $1\in S$&S$\cap N^1=\emptyset$ ……. (1)

Subject to the constraints:

$\sum_{i\in N}\sum_{j\in N} X(i,j) = m = |M|$, $n-k \le m \le n$, $M \subseteq N$ ……. (2)

$\sum_{i\in S} SC(i) \ge \sum_{j\in N^1} DR(j)$ …… (3)

$$\left.\begin{array}{l} \alpha,\beta \in S \quad Let\ \alpha,\gamma_1,\gamma_2,\gamma_3,\dots\dots\dots\dots..,\gamma_p,\beta \\ \text{be the sequence of cities } \alpha to \beta \text{ in the tour} \\ \sum_{i=1}^{p} DR(\gamma_i) \le VL, \text{Where VL } = \boldsymbol{\alpha} \end{array}\right\} \quad \text{…… (4)}$$

X (i, j) =1 or 0 …… (5)

The equation **(1)** describes the objective function of the problem i.e., to minimize the total distance/cost subjected to the constraints. The constraint **(2)** represents that the trip includes m cities. The constraint **(3)** takes care of the restriction of availability and requirement of the product between sources and destinations, i.e., the sum of the available capacities at sources is more than the sum of the demands at destinations of a product. The constraint **(4)** assumes that the demand at the destination from source to source must be lesser than or equal to the vehicle capacity. The constraint **(5)** denotes that if the vehicle is traversed from i to j its value is 1 otherwise it is 0.

4. NUMERICAL FORMULATION

The concepts and the algorithms will be illustrated by a suitable numerical example. In which we have taken number of cities as **n=9(N = {1, 2, 3, 4, 5, 6, 7, 8, 9})** among them **1** be the head quarter (**HQ**), number of loading points/source facilities **k = 3 (S = {1, 4, 8})** and remaining **6** cities are destinations **(N^1 = {2, 3, 5, 6, 7, 9}).**Let **SC (i)** is the availability of a product at sources and **DR (j)** is the requirement of a product at the destinations. Then the distance/cost matrix **D** is given below in **Table-1**.

Table – 1

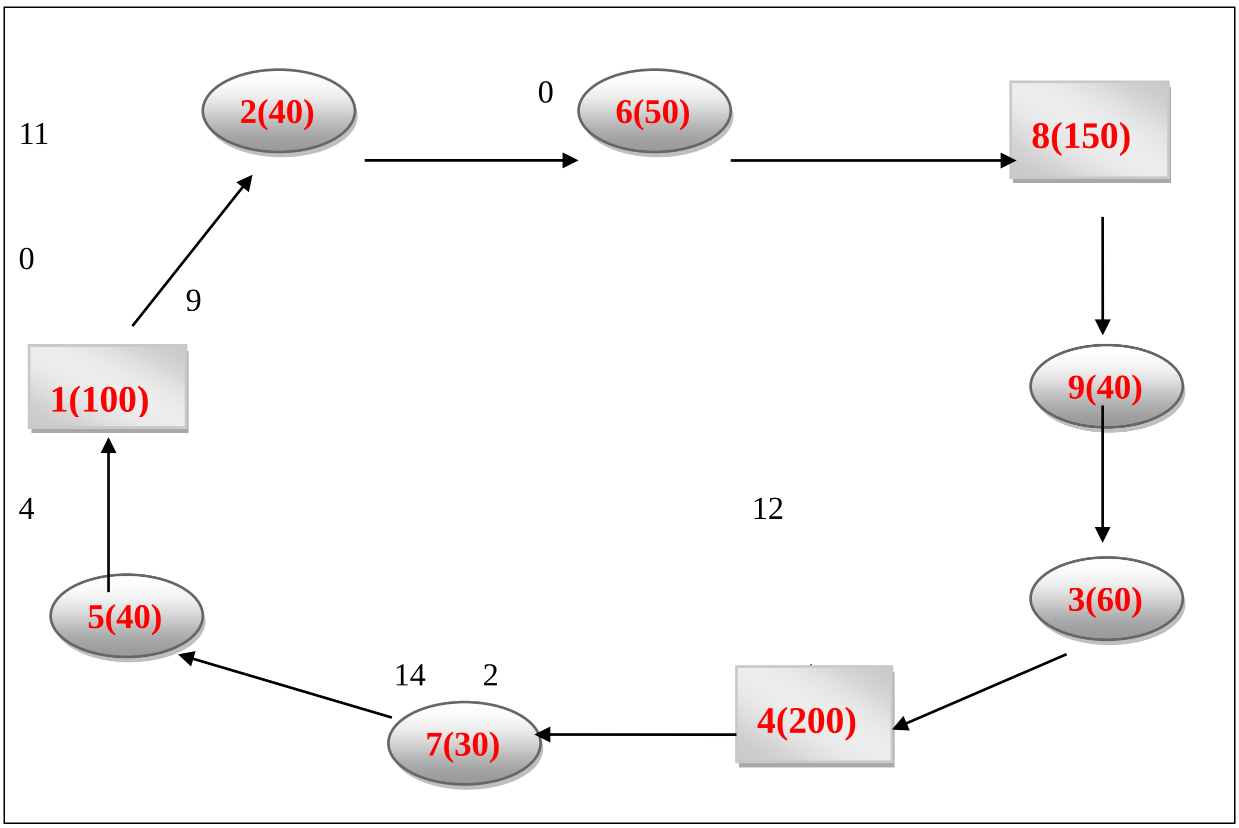
		1	2	3	4	5	6	7	8	9	SC
	1	∞	0	24	40	3	10	29	27	39	**100**
	2	31	∞	40	8	18	0	19	5	22	**-**
	3	19	32	∞	1	38	8	31	10	21	**-**
	4	22	18	24	∞	41	34	2	20	13	**200**
D(i,j)=	**5**	4	24	17	25	∞	31	21	20	32	**-**
	6	26	7	23	1	23	∞	27	11	40	**-**
	7	30	37	3	24	14	29	∞	35	28	**-**
	8	26	29	14	34	28	6	11	∞	9	**150**
	9	27	16	12	37	5	28	38	36	∞	**-**
	DR	**-**	**40**	**60**	**-**	**40**	**50**	**30**	**-**	**40**	

Suppose **D (4, 7) = 2** means that the distance between the cities **4** and **7**is **2**. Moreover, **SC** and **DR** represent that the availability of sources and requirements of destinations. The source and destination arrays **SC** and **DR** are **SC (i) = b** means that the availability of source **i** is **b**and **DR (j) = c** means that the requirement of destination **j** is **c**. Here **SC (4) = 200** means that the availability of source **4** is **200** and **DR (5) = 40** means that the requirement of destination **5** is **40**. Here we have taken the vehicle capacity$\boldsymbol{\alpha} = \mathbf{100}$ units. The objective of the problem is to find a minimum total distance to meet the requirements at the destinations subjected to the constraints.

5. Feasible Solution

Feasible solution is a solution, which satisfies all the constraints in the problem. The constraints are discussed in the mathematical formulation. Consider an ordered pair set {(1, 2), (2, 6), (6, 8), (8, 9), (9, 3), (3, 4), (4, 7), (7, 5), (5,1)} represents a feasible solution. In the following **figure-1**, the values in the ellipse denote name of the destination city and the values in the parenthesis of ellipse denotes the requirement of the destination city. The rectangle box represents sources/pickup points and its available capacity indicated in the respective parenthesis. The values along the arcs indicate the distance between the connected cities.

Figure – 1



In the above **figure – 1,** the vehicle has started its trip from the home city 1 with sufficient load of **100** units. First, it has reached the destination city **2** and supplied the destination's requirement of **40** units, from city **2**the vehicle reached city **6** and supplied its requirement of **50** units. Now, the remaining load in the vehicle is only **10** units which are insufficient to the nearest destination city's requirement. So, the vehicle reached the next nearest source city **8**to reload**90** units of its availability for shortage of vehicle load and reached the destination city **9**andthen city**3,** there it supplied its requirements**40** and **60** units respectively. Now the load in the vehicle is **0** units. As above, the vehicle again reached the nearest source city **4** to reload**100** units of its availability for shortage of vehicle load and reached its destination city**7**andthen city **5**to supply its requirement 3**0** and**40** units respectively. After completing all destinations' requirements, the vehicle has come back to the home city. So, the trip has given a feasible solution. Hence, the value of the solution is:

Z = D(1,2) + D(2,6) + D(6,8) + D(8,9) + D(9,3) + D(3,4) + D(4,7)+ D(7,5) + D(5,1)

= 0+0+11+9+12+1+2+14+4

Z = 53 units.

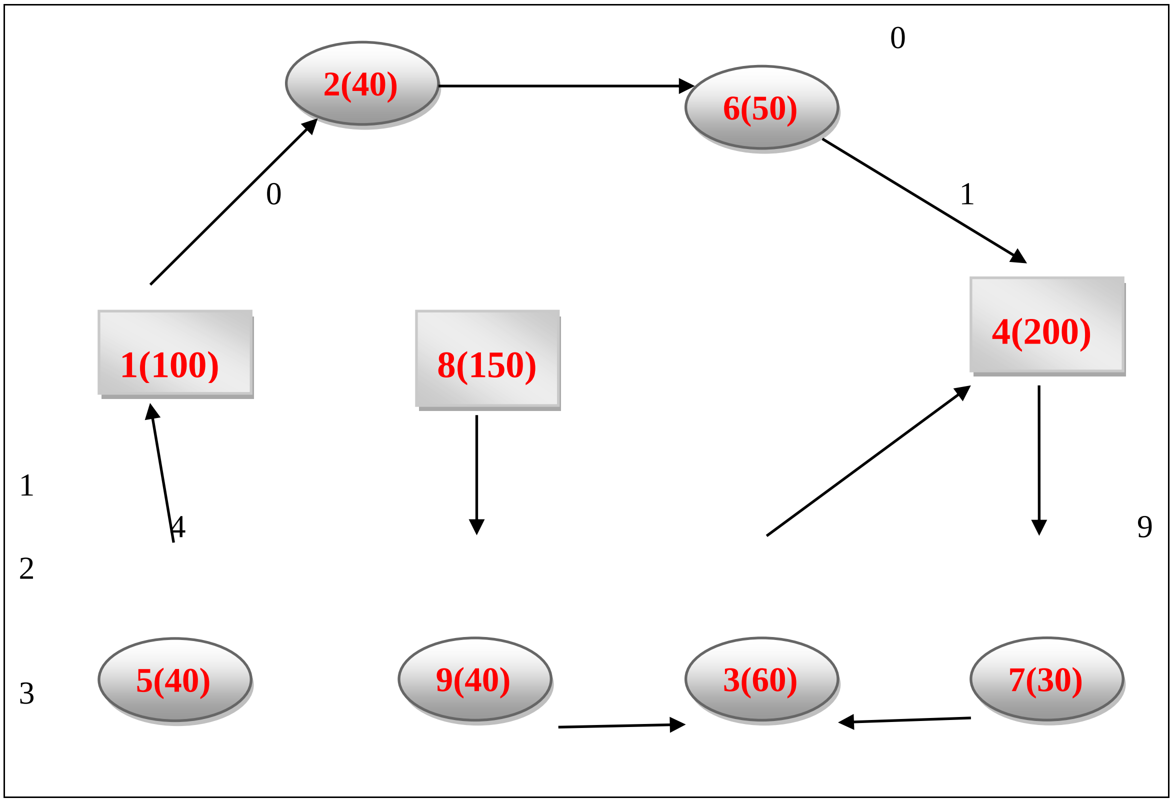
6. SOLUTION PROCEDURE

In the above **figure-1,** for the feasible solution we observe that **9 ordered pairs** are taken along with values from distance matrix for numerical example in **Table- 1.** The 9 **ordered pairs** are selected such that we they represent a feasible solution in **figure-1.** So, the problem is that we selected9 ordered pairs from distance matrix order [9×9] along with values such that the total distance is minimum to represents a feasible solution. The number of ordered pairs will be >6, here it is 9, which satisfies the supply to the 6 cities, this 9 is not fixed, it can vary. For this selection of 9 ordered pairs we arrange all the ordered pairs with increasing order and call this formation as **alphabet table** and we developed an algorithm for the selection along with checking for the feasibility.

7. INFEASIBLE SOLUTION

Infeasible solution is a solution which does not satisfies all the constraints in the problem. Consider an ordered pair set {**(1,2), (2,6), (3,4), (6,4), (4,7), (7, 3), (5, 1), (8, 9), (9,3)}** represents an infeasible solution.

Figure-2



From the above **figure-2**, the vehicle has start edits trip from the home city 1 with sufficient load of **100** units. First it has reached the destination city **2** and supplied the destination's requirement of **40** units, from city **2,** the vehicle reached city **6** and supplied its requirement of **50** units. After that the vehicle reached the source city **4**to reload, but the vehicle already approached the source city **4** from city **3**, this leads to contradict the feasible conditions. Because, in the definition of the tour the vehicle should reach each city exactly only once. Now the vehicle reached the destination city **7** from source city **4**, after supplied its requirement, the vehicle reached city **3**. Further the vehicle comeback to head quarter city from city **5.**This is again a contradiction that the vehicle has completed its tour without supplying all destinations' requirements. Finally, the vehicle started its trip from the source city **8**and reached the destination city **9,** there, it has supplied its requirement from the vehicle; again, it has reached another destination city **3** and supplied its requirement. The distance/cost for the above-mentioned ordered pairs are

Z = D(1,2)+ D(2,6) + D(3,4) + D(6,4)+ D(4,7) + D (7,3) +D (5,1) + D (8,9)+ D(9,3)

= 0 + 0 + 1 + 1 + 2 + 3 + 4 +9 + 12

Z = 32 units.

8. CONCEPTS AND DEFINITIONS

8.1. Definition of a Pattern

An indicator two-dimensional array which is associated with an assignment is called a 'pattern'. A Pattern is said to be feasible if X is a solution.

$$V(X) = \sum_{i \epsilon I} \sum_{j \epsilon J} DV(i,j)X(i,j)$$

The value V(x) is gives the total distant 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 distant of the tour for the solution represented by X. Thus, the value of the feasible pattern gives the total distant 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 zeros.

Consider the set of ordered pairs **{(1, 2), (2, 6), (6, 8), (8, 9), (9, 3), (3, 4), (4, 7), (7, 5), (5, 1)}** 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.

Table-2

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

The above **Table-2** represents a feasible pattern to the feasible solution. In the above solution **X**

(1, 2) = 1, represents that the vehicle starts its trip at home city 1 with sufficient load of **70** units and it visits the destination city **2** and supplies the destination requirement of **30** units. In similar way **X (5, 1) = 1**, represents that vehicle returns to head quarter city from city **5.**After fulfill all destination requirements. Similarly, vehicle fulfills all destination requirements by using inter loading facilities. So the above solution gives a feasible solution and it shown in **figure-1.**

The pattern in **Table-3,** gives an infeasible solution. The ordered pair set **{(1, 2), (2, 6), (3, 4), (6, 4), (4, 7), (7, 3), (5, 1), (8, 9), (9,3) }** represents a pattern which is an infeasible solution given below.

Table-3

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

The ordered pair set **{(1, 2), (2, 6), (6, 8), (8, 9), (9, 3), (3, 4), (4, 7), (7, 5), (5, 1) }** represents the pattern in **Table-2**, which is feasible solution and the ordered pair set **{(1, 2), (2, 6), (3, 4), (6, 4), (4, 7), (7, 3), (5, 1), (8, 9), (9,3) }** represents the pattern in **Table –3**, which is an infeasible solution.

8.2. Alphabet table

There are $\boldsymbol{n^2}$ ordered pairs in the two-dimensional array D. For convenience these are arranged in ascending order of their corresponding cost and are indexed from 1 to n^2 (Sundara Murthy-1979). Let SN= [1, 2, 3, ...,n^2] be the set of n^2 indices. Let D be the corresponding array of cost. If a, b$\in$ 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 and DC be the array of cumulative sum of the elements of D. The arrays SN, D, DC, R, and C for the numerical example are given in the **table-4.** If p$\in$SN then (R (p), C (p)) is the ordered pair and D (a)= DC(R(a),C(a)) is the value of the ordered pair and DC (a) $= \sum_{i=1}^{a} D(i)$. The cost assigned is the single array, for convenience it is also named as D.

Table-4: ALPHABET TABLE

SN	D	DC	R	C	SN	D	DC	R	C
1	0	0	1	2	**37**	23	436	6	5
2	0	0	2	6	**38**	24	460	1	3
3	1	1	3	4	**39**	24	484	4	3
4	1	2	6	4	**40**	24	508	5	2
5	2	4	4	7	**41**	24	532	7	4
6	3	7	1	5	**42**	25	557	5	4
7	3	10	7	3	**43**	26	583	6	1
8	4	14	5	1	**44**	26	609	8	1
9	5	19	2	8	**45**	27	636	1	8
10	5	24	9	5	**46**	27	663	6	7
11	6	30	8	6	**47**	27	690	9	1
12	7	37	6	2	**48**	28	718	7	9
13	8	45	2	4	**49**	28	746	8	5
14	8	53	3	6	**50**	28	774	9	6
15	9	62	8	9	**51**	29	803	1	7
16	10	72	1	6	**52**	29	832	7	6
17	10	82	3	8	**53**	29	861	8	2
18	11	93	6	8	**54**	30	891	7	1
19	11	104	8	7	**55**	31	922	2	1
20	12	116	9	3	**56**	31	953	3	7
21	13	129	4	9	**57**	31	984	5	6
22	14	143	7	5	**58**	32	1016	3	2
23	14	157	8	3	**59**	32	1048	5	9
24	16	173	9	2	**60**	34	1082	4	6
25	17	190	5	3	**61**	34	1116	8	4
26	18	208	2	5	**62**	35	1151	7	8
27	18	226	4	2	**63**	36	1187	9	8
28	19	245	2	7	**64**	37	1224	7	2
29	19	264	3	1	**65**	37	1261	9	4
30	20	284	4	8	**66**	38	1299	3	5
31	20	304	5	8	**67**	38	1337	9	7
32	21	325	3	9	**68**	39	1376	1	9
33	21	346	5	7	**69**	40	1416	1	4
34	22	368	2	9	**70**	40	1456	2	3
35	22	390	4	1	**71**	40	1496	6	9
36	23	413	6	3	**72**	41	1537	4	5

From the above **table-4,** Let us consider 21 $\in$ SN. It represents the ordered triple (R(21),C(21))=(4,9).Then D(21)=D(4,9)= 13 and DC(21)=129.

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, - - - -, n^2) 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. 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_3 = (1, 2, 3) then value of L_3 is V (L_3) = V (L_2) + D (a_3), In this partial word a_3=3, V(L_2) = 0; Therefore V (L_3) = 0 + D(3) = 0+1 = 1.

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 $L_k = (a_1, a_2, ----, a_k)$ can be defined as follows.

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

Consider the partial word $L_3 = (1, 2, 3)$; Then $V(L_3) = 1$

$$LB(L_3) = V(L_3) + DC(a_3 + n - k) - DC(a_3)$$
$$= 1 + DC(3 + 9 - 3) - DC(3)$$
$$= 1 + DC(9) - DC(3) = 1 + 19 - 1 = 19$$

Where $DC(a_k) = \sum_{i=1}^{ak} D(i)$. It can be seen that LB (L_k) is the value of the complete word, which is obtained by concatenating the first (n-k) letters of SN to the partial word L_k.

8.6.Feasibility criterion of a Partial Word

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

- **IR** be an array where IR (i) = 1, $i \in N$ indicates that the vehicle is visiting some city from city i Otherwise IR (i) = 0
- **IC** be an array where IC (i) = 1, $i \in N$ indicates that the vehicle is coming to city i from another city, otherwise IC (i) = 0
- **SW** be an array where SW (i) = j indicates that the vehicle is visiting city j from city i, Otherwise SW (i) = 0
- **L** be an array where L[i] = α_i, $i \in N$ is the letter in the i^{th} position of a word.

The values of the arrays L, IR, IC, SW and ST are as follows

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

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

SW(R (a_i)) = C (a_i), i = 1, 2, - - - , k and SW (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_5 = (1, 2, 3, 5,8) which is feasible. The arrays IR, IC, L, SW and ST take the values represented in **Table – 5** given below.

Table – 5

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
L	1	2	3	5	8	-	-	-	
IR	1	1	1	1	1	-	-	-	-
IC	1	1	-	1	-	1	1	-	-
SW	2	6	4	7	1	-	-	-	

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}) and CA = C (a_{p+1})

9. ALGORITHMS

ALGORITHM 1: (Algorithm for feasible checking):

STEP1 : IS (IC [CA] ==1) IF YES GOTO 1

IF NO GOTO 2

STEP2 : IS (b [RA] ==1) IF YES GOTO 4

IF NO GOTO 3

STEP3 : IS (b [CA]==1) IF YES GOTO 5

IF NO GOTO 20

STEP4 : IS (b [CA]==1) IF YES

{DR [RA]=DR [RA] +DR [CA]; GOTO 10}

IF NO GOTO 20

STEP5 : IS (SA [RA]>=DR [CA]) IF YES GOTO 6

IF NO GOTO 1

STEP6 : W=CA
A=RA GOTO 7

STEP7 : IS (SW [W] ==0) IF YES

{SA [W] =SA [A]-DR [W]; GOTO 11}
IF NO GOT 9

STEP8 : IS b [W]==1) IF YES GOTO 7
IF NO {SA [A]=SA [RA]-DR [CA]; GOTO 21}

STEP9 : A=W
W=SW [W]
LC=LC+1
IS (W==RA) IF YES GOTO 10
IF NO GOTO 8

STEP10 : IS (X>=DR [RA]) IF YES GOTO11

IF NO {DR [RA] =DR [RA]-DR [CA]; GOTO 1}

STEP11 : DR [RA] =DR [RA]-DR [CA];
W=RA;
A=CA; GOTO 12
STEP12 : IS (SWI [W] ==0) IF YES

{DR [W]=DR[W]+DR[A]; GOTO 11}

IF NO GOTO 13

STEP13 : M=SWI[W];

IS B [M]==1) IF YES GOTO19

IF NO GOTO 14

STEP14 : DR[W]=DR[W]+DR [A]

IS (SA[M]>=DR[W]) IF YES

{QT=SA[M] – DR [W]; GOTO 15}

IF NO GOTO 1

STEP15 : W=CA GOTO 16

STEP16 : IS (SW[W]==0) IF YES

{SA[W]=QT; GOTO 11}

IF NO GOTO 17

STEP17 : IS (b [W]==1) IF YES GOTO 18

IF NO {SA[A]=QT; GOTO 21}

STEP18 : A=W

W=SW[W]

LC=LC + 1

IS (W==RA) IF YES GOTO 10

IF NO GOTO 16

STEP19 : DR[W]=DR[W]+DR [A]

DR[M]=DR[M]+DR [W]

IS (X>=DR[M]) IF YES

{W=M;A=W; GOTO 12}

IF NO GOTO 1

STEP20 : W=CA; GOTO 21

STEP21 : IS (SW[W]==0) IF YES GOTO 11

IF NO GOTO 22

STEP22 : LC=LC+ 1

W=SW[W]

IS (W==RA) IF YES GOTO 10

IF NO GOTO 21

STEP 21 : X = 1 GO TO STOP

STEP 22 : STOP

We start with the partial word $L_1 = (a_1) = (1)$. A partial word L_k is constructed as $L_k = L_{k-1} * (\alpha_p)$. Where * indicates chain formulation. We will calculate the values of V (L_k) and LB (L_k) simultaneously. Then two situations arise 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.

   Now a Lexi-Search Algorithm to find an Optimal Feasible word is developed.

ALGORITHM 2: (Lexi-Search Algorithm)

STEP 0 : Initialization

The arrays SN, R, C, D, DC,B, M, LB, V, L, DR, r, c and values of N are made available IR, IC, SW, SWI and ST are initialized to zero. The values I=1, J=0, VT=999 and Max=N^2- N

STEP 1 : J=J+1;

L[K]=J;

LC=1;

IS (J>Max) IF YES GOT 9;

IF NO GOT 2

STEP 2 : RA=r[J]

CA=c[J];

V[K]=V[K-1]+d[J];

LB[K]=V[K]+DC[J+n-K]-DC [J] GOTO 3

STEP 3 : IS (LB[K]>=VT) IF YES GOTO

IF NO GOTO 4

STEP 4 : Check feasible (Using Algorithm 1)

IS (IX=0) IF YES GO TO 2

IF YES GO TO 5

STEP 5 : IS (LC==n) IF YES GOTO 8

IF NO GOT 6

STEP 6 : IS (b[RA]==1) IF YES

{DR[RA]=DR[RA] – DR[CA];GOTO 1}

IF NO GOTO 1

STEP 7 : L[K]=J

IC[CA]=1

IR[RA]= 1

SWI[CA]=RA

SW [RA] = CA

DR[CA]=0

K=K+1

GOTO STEP 1

STEP 8 : VT=LB[K]

L[K]=J

IS (b[RA]==1) IF YES

{DR[RA]=DR[RA] – DR[CA]; GOTO 10}

IF NO GOT 10

STEP 9 : IS (K==1) IF YES GOT 11

IF NO GOTO 10

STEP10 : K=K-1

J = L [K]

RA=r[J]

CA = c[J]

L [J] = 0

IC[CA]=0

IR [RA] = 0

SWI[CA]=0

SW[RA]= 0

LB [K+1] = 0

L[K+1]=0

V[K+1]=0

IS (b[RA]==1) IF YES

{DR[RA]=DR[RA] – DR[CA]; GOTO 1}

IF NO GOTO 1

STEP11 : STOP/END

The current value of VT at the end of the search is the value of the optimal feasible word. At the end if VT = 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 **Table-6.**The columns (1), (2), (3), (4), (5), (6), (7), (8) and (9) gives the letters in the first, second, third, fourth, fifth, sixth, seventh, eighth and ninth positions 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	8	9	V	LB	R	C	Remarks
1	1									0	19	1	2	A
2		2								0	19	2	6	A
3			3							1	19	3	4	A
4				4						2	19	6	4	R
5				5						3	23	4	7	A

6					6					6	23	1	5	R
7					7					6	26	7	3	R
8					8					7	30	5	1	A
9						9				12	30	2	8	R
10						10				12	33	9	5	R
11						11				13	36	8	6	R
12						12				14	39	6	2	R
13						13				15	42	2	4	R
14						14				15	44	3	6	R
15						15				16	47	8	9	A
16							16			26	47	1	6	R
17							17			26	48	3	8	R
18							18			27	50	6	8	A
19								19		38	50	8	7	R
20								20		39	52	9	3	A
21									21	52	52	4	9	R
22									22	53	53	7	5	A,VT=53
23								21		40	54	4	9	R, >VT
24							19			27	52	8	7	R
25							20			28	55	9	3	R, >VT
26						16				17	49	1	6	R
27						17				17	51	3	8	R
28						18				18	54	6	8	R, >VT
29					9					8	34	2	8	R
30					10					8	37	9	5	A
31						11				14	37	8	6	R
32						12				15	40	6	2	R
33						13				16	43	2	4	R
34						14				16	45	3	6	R
35						15				17	48	8	9	R
36						16				18	50	1	6	R
37						17				18	52	3	8	R
38						18				19	55	6	8	R, >VT
39					11					9	41	8	6	R
40					12					10	45	6	2	R
41					13					11	48	2	4	R
42					14					11	51	3	6	R
43					15					12	54	8	9	R, >VT
44				6						4	27	1	5	R
45				7						4	31	7	3	A

46					8					8	31	5	1	A
47						9				13	31	2	8	R
48						10				13	34	9	5	A
49							11			19	34	8	6	R
50							12			20	36	6	2	R
51							13			21	38	2	4	R
52							14			21	40	3	6	R
53							15			22	42	8	9	R
54							16			23	44	1	6	R
55							17			23	45	3	8	R
56							18			24	47	6	8	A
57								19		35	47	8	7	A
58									20	47	47	9	3	R
59									21	48	48	4	9	A,VT=48
60								20		36	49	9	3	R, >VT
61							19			24	49	8	7	R, >VT
62						11				14	37	8	6	R
63						12				15	40	6	2	R
64						13				16	43	2	4	R
65						14				16	45	3	6	R
66						15				17	48	8	9	R, =VT
67					9					9	35	2	8	R
68					10					9	38	9	5	A
69						11				15	38	8	6	R
70						12				16	41	6	2	R
71						13				17	44	2	4	R
72						14				17	46	3	6	R
73						15				18	49	8	9	R, >VT
74					11					10	42	8	6	R
75					12					11	46	6	2	R
76					13					12	49	2	4	R, >VT
77				8						5	36	5	1	A
78					9					10	36	2	8	R
79					10					10	39	9	5	A
80						11				16	39	8	6	R
81						12				17	42	6	2	R
82						13				18	45	2	4	R
83						14				18	47	3	6	R
84						15				19	50	8	9	R, >VT
85					11					11	43	8	6	R

86					12					12	47	6	2	R
87					13					13	50	2	4	R, >VT
88				9						6	40	2	8	R
89				10						6	44	9	5	A
90					11					12	44	8	6	R
91					12					13	48	6	2	R, =VT
92				11						7	49	8	6	R, >VT
93			4							1	23	6	4	A
94				5						3	23	4	7	A
95					6					6	23	1	5	R
96					7					6	26	7	3	A
97						8				10	26	5	1	A
98							9			15	26	2	8	R
99							10			15	28	9	5	A
100								11		21	28	8	6	R
101								12		22	30	6	2	R
102								13		23	31	2	4	R
103								14		23	32	3	6	R
104								15		24	34	8	9	A
105									16	34	34	1	6	R
106									17	34	34	3	8	A,VT=34
107								16		25	35	1	6	R, >VT
108							11			16	31	8	6	R
109							12			17	33	6	2	R
110							13			18	35	2	4	R, >VT
111						9				11	29	2	8	R
112						10				11	32	9	5	A
113							11			17	32	8	6	R
114							12			18	34	6	2	R, =VT
115						11				12	35	8	6	R, >VT
116					8					7	30	5	1	A
117						9				12	30	2	8	R
118						10				12	33	9	5	A
119							11			18	33	8	6	R
120							12			19	35	6	2	R, >VT
121						11				13	36	8	6	R, >VT
122					9					8	34	2	8	R, =VT
123				6						4	27	1	5	R
124				7						4	31	7	3	A
125					8					8	31	5	1	A

126						9				13	31	2	8	R
127						10				13	34	9	5	R, =VT
128					9					9	35	2	8	R, >VT
129				8						5	36	5	1	R, >VT
130			5							2	28	4	7	A
131				6						5	28	1	5	R
132				7						5	32	7	3	A
133					8					9	32	5	1	A
134						9				14	32	2	8	R
135						10				14	35	9	5	R, >VT
136					9					10	36	2	8	R, >VT
137				8						6	37	5	1	R, >VT
138			6							3	33	1	5	R
139			7							3	38	7	3	R, >VT
140		3								1	24	3	4	A
141			4							2	24	6	4	R
142			5							3	29	4	7	A
143				6						6	29	1	5	R
144				7						6	33	7	3	R
145				8						7	38	5	1	R, >VT
146			6							4	34	1	5	R, =VT
147		4								1	29	6	4	A
148			5							3	29	4	7	A
149				6						6	29	1	5	R
150				7						6	33	7	3	A
151					8					10	33	5	1	A
152						9				15	33	2	8	R
153						10				15	36	9	5	R, >VT
154					9					11	37	2	8	R, >VT
155				8						7	38	5	1	R, >VT
156			6							4	34	1	5	R, =VT
157		5								2	35	4	7	R, >VT
158	2									0	24	2	6	A
159		3								1	24	3	4	A
160			4							2	24	6	4	R
161			5							3	29	4	7	A
162				6						6	29	1	5	A
163					7					9	29	7	3	R
164					8					10	33	5	1	R
165					9					11	37	2	8	R, >VT

166				7						6	33	7	3	R
167				8						7	38	5	1	R, >VT
168			6							4	34	1	5	R, =VT
169		4								1	29	6	4	A
170			5							3	29	4	7	A
171				6						6	29	1	5	A
172					7					9	29	7	3	A
173						8				13	29	5	1	R
174						9				14	32	2	8	R
175						10				14	35	9	5	R, >VT
176					8					10	33	5	1	R
177					9					11	37	2	8	R, >VT
178				7						6	33	7	3	A
179					8					10	33	5	1	A
180						9				15	33	2	8	R
181						10				15	36	9	5	R, >VT
182					9					11	37	2	8	R, >VT
183				8						7	38	5	1	R, >VT
184			6							4	34	1	5	R, >VT
185		5								2	35	4	7	R, >VT
186	3									1	30	3	4	A
187		4								2	30	6	4	R
188		5								3	36	4	7	R, >VT
189	4									1	36	6	4	R, >VT

11. COMMENTS

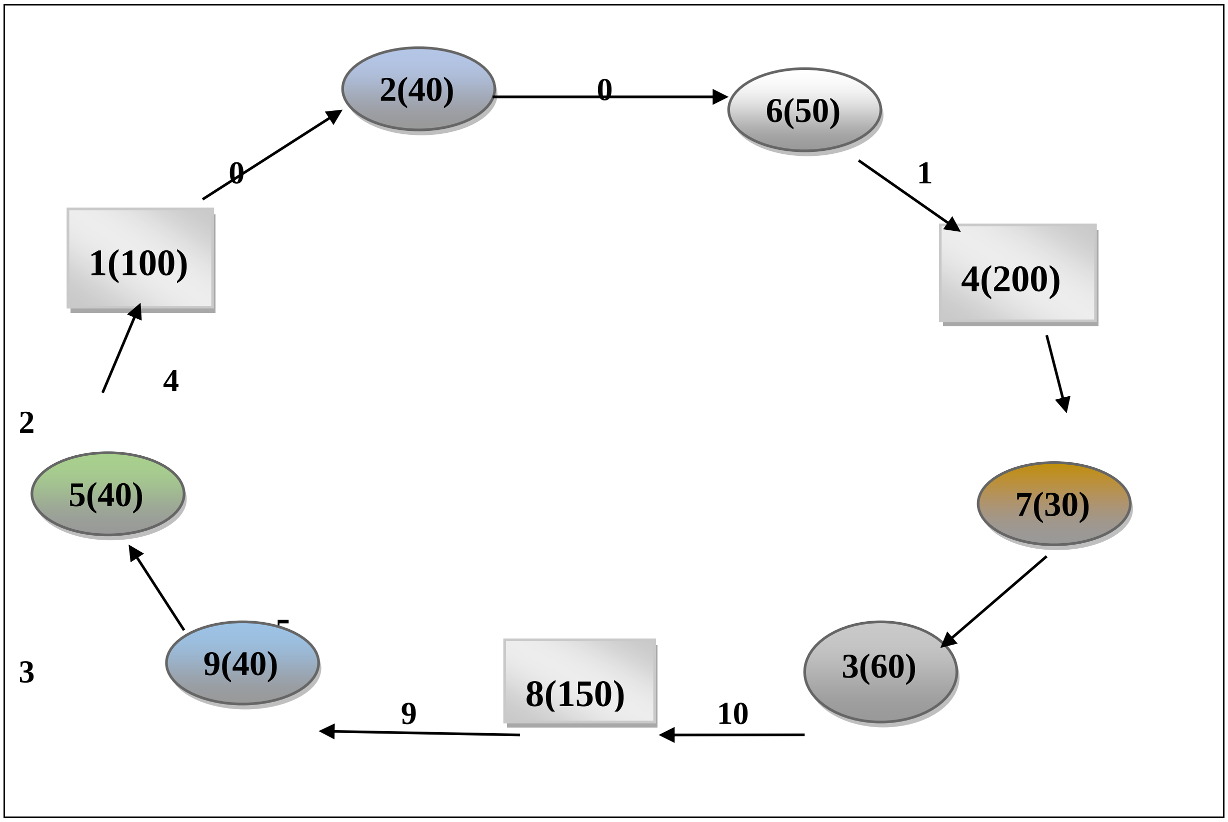
The shaded row in the above **Table-6**, gives optimal solution of the taken numerical example and at the end of the search the current value of **VT** is **34.** Then the partial word is **L_9 = (1, 2, 4, 5, 7, 8, 10, 15, 17)** is an optimal feasible word. It is given in the **106^{th}** row of the search table. For this optimal word the arrays L, IR, IC, SW and ST are given in the following **Table – 7**.

Table -7

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
L	1	2	4	5	7	8	10	15	17
IR	1	1	1	1	1	1	1	1	1
IC	1	1	1	1	1	1	1	1	1
SW	2	6	8	7	1	4	3	9	5

At the end of the search table the optimal solution value of **VT** is **34**and is the value of optimal feasible word **L_9 = (1, 2, 4, 5, 7, 8, 10 15, 17)**. Then the following **figure-3**represents the optimal solution to the problem.

Figure –3



From the above **figure-3**, the vehicle has started its trip from the home city 1 with sufficient load of **100** units. First it has reached the destination city **2** and supplied its requirement of **40** units. From city **2,** the vehicle reached city **6** and supplied its requirement of **50** units. Now, the remaining load in the vehicle is only**10** units which are insufficient for the nearest destination city's requirement. So, the vehicle reached the nearest source city **4** to reload**90** units of its availability for shortage of vehicle load and reached the destination city**7**andthen city **3,** there it supplied its requirement **30** and**60** units respectively. Now the load in the vehicle is only **10** units. As above, the vehicle again approached the nearest source city **8** to reload**90** units of its availability for shortage of vehicle load and reached destination city**9** and then **5,** there it supplied its requirement **40**and**40** units respectively. After completed all destinations requirement, the vehicle has come back to the home city. So, the trip has given a feasible solution. Hence the value of the solution is:

Z = D(1,2)+D(2,6)+D(6,4)+D(4,7)+D(7,3)+D(3,8)+D(8,9)+D(9,5)+D(5,1)

= 0+0+1+2+3+10+9+5+4

= 34 units

Consider the set of ordered pairs **{(1, 2), (2, 6), (6, 4), (4, 7), (7, 3), (3, 8), (8, 9), (9, 5), (5, 1)}** 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

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

12. EXPERIMENTAL RESULTS

The following table shows that the computational results for proposed Lexi-Search algorithm using pattern recognition technique. We presented computer program for this algorithm in C language, and it is verified by the system COMPAQ dx2280 MT. We ensure this algorithm by trying a set of problems for different sizes. We took different random numbers as values in distance matrix. The distance Matrix D (i, j,) takes the values uniformly random in [0, 200]. We tried a set of problems by giving different values for N, S, D. The results are tabulated in the **Table -9** are given below. For each instance, five to seven data sets are tested. It is seen that the time required for the search of the optimal solution is fairly less. In the following microseconds are represented by zero.

In the **Table-9**, SN = serial number, N = number of cities, S = number of sources/loading points, D = number of destinations. 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	N	S	D	NPT	VT	CPU Run Time in seconds
						Avg. AT+ ST
1	5	2	3	6	35	0.0000
2	10	4	6	6	55	0.0000
3	12	5	7	8	79	0.0000
4	15	6	9	7	87	0.1035
5	17	6	11	7	98	0.1089
6	20	8	12	6	107	0.2481
7	22	9	13	6	119	0.2735
8	25	10	15	6	124	0.3548
9	28	13	15	6	132	0.5196
10	30	14	16	6	148	0.6743
11	35	17	18	5	167	0.6894
12	40	18	22	5	185	0.8193

13. COMPARISON DETAILS

We implemented Lexi-search Algorithm (LSA) using Pattern Recognition Technique with C language for this model. We tested the proposed algorithm by different set of problems and compared the computational results with the published Vehicle Routing problem by Madhu Mohan Reddy(2015)**.**Then **Table-10** shows that the comparative results of different sizes.In the following table microseconds are represented by zero.

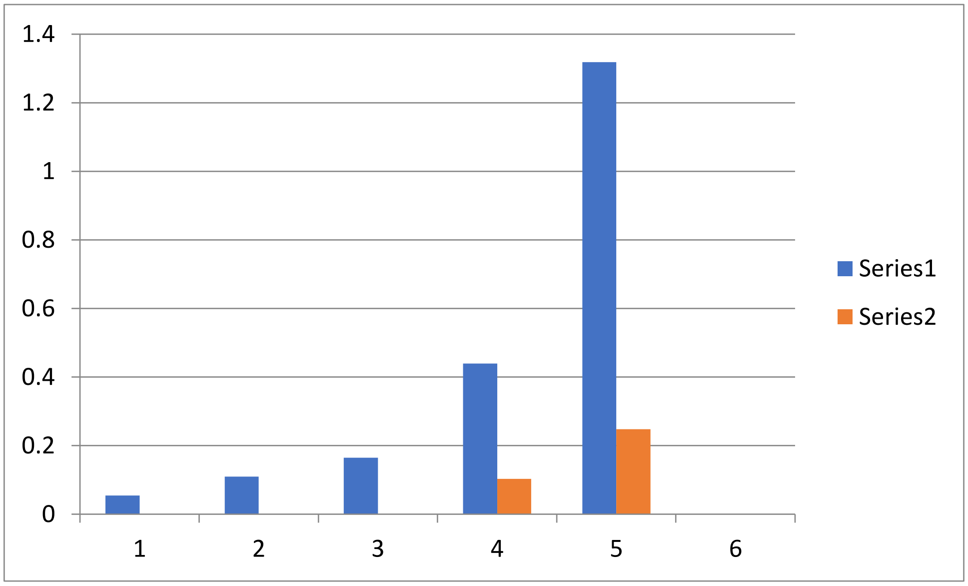
Table-10

S. No.	No. of cities	Published model	Proposed model
1	5	0.054945	0.0000
2	10	0.109890	0.0000
3	12	0.164835	0.0000
4	15	0.439560	0.1035
5	20	1.318681	0.2481

In the above **Table-10**, the last two columns show the CPU run time of published model and proposed model. As compared the two models of sizes N=5,10,12,15 & 20. The runtime of this instance with the existing model are 0.054945sec, 0.109890sec, 0.164835sec, 0.439560sec & 1.318681sec and the proposed model took 0.0sec, 0.0sec, 0.0sec, 0.1035sec & 0.2481 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 also.

The graphical representation of the CPU run time for the two models presented in the above 5 instances is given below. In the **Graph-1**, X axis taken the SN and Y axis taken the values of CPU run time for the published and proposed models.

Graph-1



From the above **Graph-1**, series2 represent that CPU run time for getting optimal solution by proposed model and series 1 represent that CPU run time for searching the optimal solution by the published model. Also, the proposed model takes less time than published model for giving the solution.

14. CONCLUSION

In the above conversation, we have presented an exact algorithm called Lexi-Search algorithm using pattern recognition technique to solve "Vehicle Routing Problem with Inter Loading Facilities". First the model is formulated into a zero-one programming problem. A Lexi-Search Algorithm using Pattern Recognition Technique is developed for getting an optimal solution. The problem is discussed with suitable numerical illustration. We have programmed the proposed algorithm using C-language. The computational details are reported. As an observation the CPU run time is fairly less for higher values to the parameters of the problem to obtain optimal solutions.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] M. Gendreau, G. Laporte, J. Potvin, Metaheuristics for the capacitated vehicle routing problem, in: P. Toth, D. Vigo (Eds.), The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, SIAM, Philadelphia. (2015), 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(1) (2006), 133–150.

[3] R. Baldacci, E. Hadjiconstantinou, A. Mingozzi, An exact algorithm for the traveling salesman problem with deliveries and collections, Networks. 42(1) (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(2) (1996), 317-325.

[5] P. Borovska, M. Lazarova, S. Bahudejla, Strategies for parallel genetic computation of optimization problems, J. Biotechnol. Biotechnol. Equipment. 21(2) (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), 470 - 475.

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

[8] G. Vijayalakshmi, Lexi-search approach to travelling salesman problem, IOSR J. Math. 6 (2013), 1-08

[9] K. Chendra Sekhar et.al., A Problem Recognition Lexisearch Approach to traveling salesman problem with additional constraints, Int. J. Computer Sci. Eng. 4(2) (2012), 307-320.

[10] Z.H. Ahmed, A data-guided lexisearch algorithm for the bottleneck travelling salesman problem, Int. J. Oper. Res. 12(1) (2011), 20-33.

[11] S. Arora et.al., A Lexisearch Algorithm for a time minimizing assignment problem, Opsearch, 35(2017), 193-213.

[12] A. Kumar, A. Kaur, A. Gupta, New methods for solving fuzzy transportation problems with some additional transshipments, ASOR Bull. 30(1) (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, A. Hertz, M.G. Speranza, A tabu search algorithm for the split delivery vehicle routing problem, Transport. Sci. 40 (2006), 1-132.