Available online at http://scik.org

J. Math. Comput. Sci. 11 (2021), No. 4, 4766-4798

https://doi.org/10.28919/jmcs/5686

ISSN: 1927-5307

SOLVING THE CAPACITATED VEHICLE ROUTING PROBLEM WITH LEXI-SEARCH APPROACH

B. MAHABOOB[1,*], P. VENKATESWARARAO[2], V.B.V.N. PRASAD[1], G. BALAJI PRAKASH[1],

T. NAGESWARA RAO[1], D. SATEESH KUMAR[1]

[1]Department of Mathematics, Koneru Lakshmaiah Education Foundation, Deemed to be University, Green Fields, Vaddeswaram, Guntur-522 502, India

[2]KL Business School, Koneru Lakshmaiah Education Foundation, Deemed to be University, Green Fields, Vaddeswaram, Guntur-522 502, India

Copyright © 2021 the author(s). This is an open access article distributed under the Creative Commons Attribution License,which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract: In this paper we present a variant vehicle routing problem called “Solving the Capacitated Vehicle Routing Problem with Lexi-Search Approach” (CVRP). The purpose of this article is to propose an efficient Lexi-Search Algorithm using pattern recognition technique for solving CVRP 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. In this paper we focus our investigation on solving the capacitated VPR (CVPR) and considered a variant vehicle routing problem called as **“**Solving the Capacitated Vehicle Routing Problem with Lexi-Search Approach”. 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.

Keywords: transportation cost; waiting time; meta heuristics; infeasible solutions; Lexi-search approach.

2010 AMS Subject Classification: 68U05.

---

*Corresponding authors

E-mail addresses: bmahaboob750@gmail.com

Received March 12, 2021

1. INTRODUCTION

The general trend in the transportation sector is that transportation companies are merging to larger units which can be provided a large number of delivery services. In order to get the most possible benefit from the vehicle fleet, it can be attractive to serve conceptually different transportation tasks by the same fleet, thus models are needed that can handle all additional constraints associated with a transportation task, for example, provide a unified approach for several Vehicle Routing Problems with Time Windows.

There are objectives other than minimizing the transportation cost that may arise in vehicle routing problems such as minimizing the number of vehicles required to serve all customers, balancing the routes, or minimizing the waiting time of a customer.

The Vehicle Routing Problem (VRP) is a generic name given to a whole class of problems in which a set of routes for a fleet of vehicles based at one or several depots must be minimized for a number of geographically dispersed cities or customers. In the VRP, the road network is represented by a graph with arcs and vertices. Arcs represent roads and vertices represent road intersections, junctions, customer locations, and the depot. Each arc has an associated cost. Each customer location vertex has an associated number of goods to be delivered. Each vehicle has its own capacity and cost associated with its utilization.

Another well-known generalization of the VRP is the Multi-Depot Vehicle Routing Problem (MDVRP). In this extension every customer is visited by a vehicle based at one of several depots. In the standard MDVRP every vehicle route must start and end at the same depot. There exist only a few exact algorithms for this problem. Laporte *et al.* (1984) as well as Laporte *et al.* (1988) have developed exact branch-and-bound algorithms but, as mentioned earlier, these only work well on relatively small instances. Several heuristics have been put forward for the MDVRP. Early heuristics based on simple construction and improvement procedures have been developed by Tillman (1969), Tillman and Hering (1971), Tillman and Cain (1972), Wren and Holliday (1972), Gillett and Johnson (1976), Golden *et al.* (1977) and Raft (1982). More recently, Chao *et al.* (1993) have proposed a search procedure combining Dueck's (1993) record-to-record local method for the reassignment of customers to different vehicle routes, followed by Lin's 2-opt procedure (1965) for the improvement of individual routes. Renaud *et al*. (1996) described a Tabu search heuristic in which an initial solution is built by first assigning every customer to its nearest depot. A Petal algorithm has developed by the latter authors (1996) is then used for the solution of the VRP associated with each depot. It finally applies an improvement phase using either a subset of the 4-opt exchanges to improve individual routes, swapping customers between routes from the same or different depots, or exchanging customers between three routes. The Tabu search approach of Cordeau *et al.* (1997) is probably the best known algorithm for the MDVRP. An initial solution is obtained by assigning each customer to its nearest depot and a VRP solution is generated for each depot by means of a sweep algorithm. Improvements are performed by transferring a customer between two routes incident to the same depot, or by relocating a customer in a route incident to another depot. Reinsertions are performed by means of the GENI heuristic (1992). One of the main characteristics of this algorithm is that infeasible solutions are allowed throughout the search. Continuous diversification is achieved through the penalization of frequent moves. In this paper we focus our investigation to solving the capacitated VPR (CVPR).

As in most NP-hard problems, three approaches are typically employed to solve these types of problems: heuristics, approximation methods and exact methods. While heuristics do not provide guarantees about the solution quality, they are useful in practical contexts because of their speed and ability to handle large instances. A special class of heuristics is Meta-Heuristics, which are general frameworks for heuristics. Approximation algorithms are a special class of heuristic that provides a solution and an error guarantee.

We are aware of a number of VRP algorithms based on this approach. One of the first attempts to apply Tabu search to the VRP is due to Willard (1989). Here, the problem is first transformed into a TSP by replication of the depot, and the search is restricted to neighbour solutions that can be reached by means of 2-opt or 3-opt interchanges while satisfying the VRP constraints. In Pureza and Franca (1991), the search proceeds from one solution to the next by swap-ping vertices between two routes. Osman (1991, 1993) uses a combination of 2-opt moves, vertex reassignments to different routes, and vertex interchanges between routes. Another algorithm was developed by Semet and Taillard (1993) for the solution of a real-life VRP containing several features, and different from the version considered in this paper. Here the basic Tabu move consists of moving a city from its current route into an alternative route. Finally, Taillard (1992) partitions the vertex set into clusters separately through vertex moves from one route to another. Clusters are updated throughout the algorithm. Note that in all these algorithms, a feasible solution is never allowed to be-come infeasible with respect to side constraints. Exact methods guarantee that the optimal solution is found if the method is given sufficiently time and space. The VRP is a hard combinatorial problem, and to this day only relatively small VRP instances can be solved to optimality. Interesting exceptions are the problems solved to optimality by Fisher (1989), using minimum k-trees.

In this paper we present a variant vehicle routing problem called “Solving the Capacitated Vehicle Routing Problem with Lexi-Search Approach” (CVRP). There are ‘n’ cities in that city ‘1’ is head quarter city, each city have positive demand. Multiple vehicles with uniform capacity starts from head quarter and supply according to their demand with total minimum distance and return to head quarter. The purpose of this paper is to propose an efficient Lexi-Search Algorithm using pattern recognition technique for solving CVRP 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 remainder of this paper is organized as follows.

2. PROBLEM DESCRIPTION

In this discussion we considered a variant vehicle routing problem called as “Solving the Capacitated Vehicle Routing Problem with Lexi-Search Approach”. There are some vertices available. Vertex ‘1’ considered a depot and the remaining vertices are considered as cities, each city has known demand. Few vehicles with uniform capacity are available at a depot. Each vehicle starts at depot and supply according to their demand in different routes with minimum cost/distance and return to depot. Each vehicle visits each city only once. Here, there is a restriction that the vehicle capacity is always greater than or equal to total demand of cities in that vehicle route.

Let G = (N, A) be a directed graph where N = {1, 2, 3………, n} is a vertex set, and A = {(i, j): i ≠ j]} is an arc set. Vertex '1' denotes depots at which **'m'** identical vehicles are based with uniform capacity **'Q'** are known and the remaining vertices of **N** represent (n-1) cities. Every city **i** have a requirement $\mathbf{Q_i}$ which is known .The value of **m** is given. Every arc (i, j) is associated a positive distance $\mathbf{c_{ij}}$. (For the sake of simplicity, the terms "distances," "travel times," and "travel costs" will be used interchangeably.) The VRP consists of designing a set of total least cost vehicle routes in such a way that every route starts and ends at the depot. Every city of N - {1} is visited exactly once by exactly one vehicle and every city is associated with a positive demand $\mathbf{Q_i}$. The total demand of any vehicle route will not exceed the vehicle capacity **Q**. The aim of the problem is to find feasible solution which meets the above conditions such that the total cost/distance is minimum. A 0-1 programming formulation of the problem of routing to minimize the cost subject to vehicle capacity constraint is given below.

3. MATHEMATICAL FORMULATION

$$V(R) = \text{Minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad \text{..} \quad (1)$$

Subject to

$$\sum_{i} x_{ij} = 1 \text{ for each } j \in N,\ j=2, 3\ldots n \qquad \text{..} \quad (2)$$

$$\sum_{j} x_{ij} = 1 \text{ for each } i \in N,\ i=2, 3\ldots n \qquad \text{..} \quad (3)$$

$$\sum_{i} x_{i1} = m \qquad \text{..} \quad (4)$$

$$\sum_{j} x_{1j} = m \qquad \text{..} \quad (5)$$

$$\sum_{i} \sum_{j} x_{ij} = n+m-1 \qquad \text{..} \quad (6)$$

$X_{ij} \in \{1, 0\}$ for each i,j ∈ N .... (7)

$$m = \left\{ \left[\frac{\sum Qi}{Q} \right], if \ \frac{\sum Qi}{Q} is \ an \, \text{int} \, eger \, or \left[\frac{\sum Qi}{Q} \right] + 1, \ if \ \frac{\sum Qi}{Q} is \, non - \text{int} \, eger \right\} \text{ .... (8)}$$

Constraint (1) represents the objective function i.e., a fixed fleet of delivery vehicles of uniform capacity must service to known customer demands of a single commodity with the minimum transportation cost.

Constraint (2) and (3) represents that each city is visited exactly only once, except the home city.

Constraint (4) and (5) represents that **'m'** vehicles starts at home city visits cities according to the requirements and returns to home city.

Constraint (6) represents that the number of connectivity's on supplying the required demands from the home city.

Constraint (7) represents that a vehicle travels from i^{th} city to j^{th} city is denoted by 1 otherwise 0.

Constraint (8) represents that the minimum number of vehicles.

In the sequel we developed a Lexi-Search algorithm using "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 with no of cities n=8, each vehicle capacity Q =30, no of vehicles m =3, demand at the cities 2, 3, 4, 5, 6, 7, 8 are 10, 8, 12, 10, 6, 14, 10 units respectively. The cost/distance matrix C (i, j) of CVRP is given as follows.

Table – 1

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	∞	0	30	2	32	9	2	17
2	63	∞	4	39	3	62	50	9
3	4	73	∞	68	74	56	19	38
4	6	75	83	∞	12	54	59	53
5	3	64	54	10	∞	60	42	8
6	5	31	15	23	16	∞	62	4
7	12	58	17	0	70	24	∞	34
8	11	52	10	25	7	72	26	∞

In the above numerical example given in **Table – 1**, C (i, j) = ∞, where (i=j) these cost of distance pairs are not relevant in the solution paths for finding routs for vehicles. Though the values of C (i,j)'s taken positive integers and the cost can be any positive quantity. Suppose C (3, 4) = 68 means that the cost/distance of the product supply from city 3 to city 4 is 68 units.

Table - 2

$\mathbf{Q_i}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	-	10	8	12	10	6	14	10

From the above **Table – 2**, $\mathbf{Q_i} = \boldsymbol{\alpha}$ means demand at particular city **i** is $\boldsymbol{\alpha}$. Suppose Q_5 = 10 means demand at city 5 is 10 units. Here $\mathbf{Q_1}$= **'-'**means demand at depot is zero. The total number of vehicles/routes **(m)** = 3.

5. Feasible Solution

Consider an ordered pair set { (1, 2), (7, 4), (1, 7), (2, 5), (5, 1), (3, 1), (6, 8), (4, 1), (1, 6), (8, 3) } represents a feasible allocation and gives the feasible solution. In the following **figure- 1,** circles represent cities, value in it represents name of the city, square represent head quarter city, the value in the diamond shape represents requirement at that particular city and value on arc represents cost/distance between those particular cities. Then **figure – 1** represents feasible solution as follows.

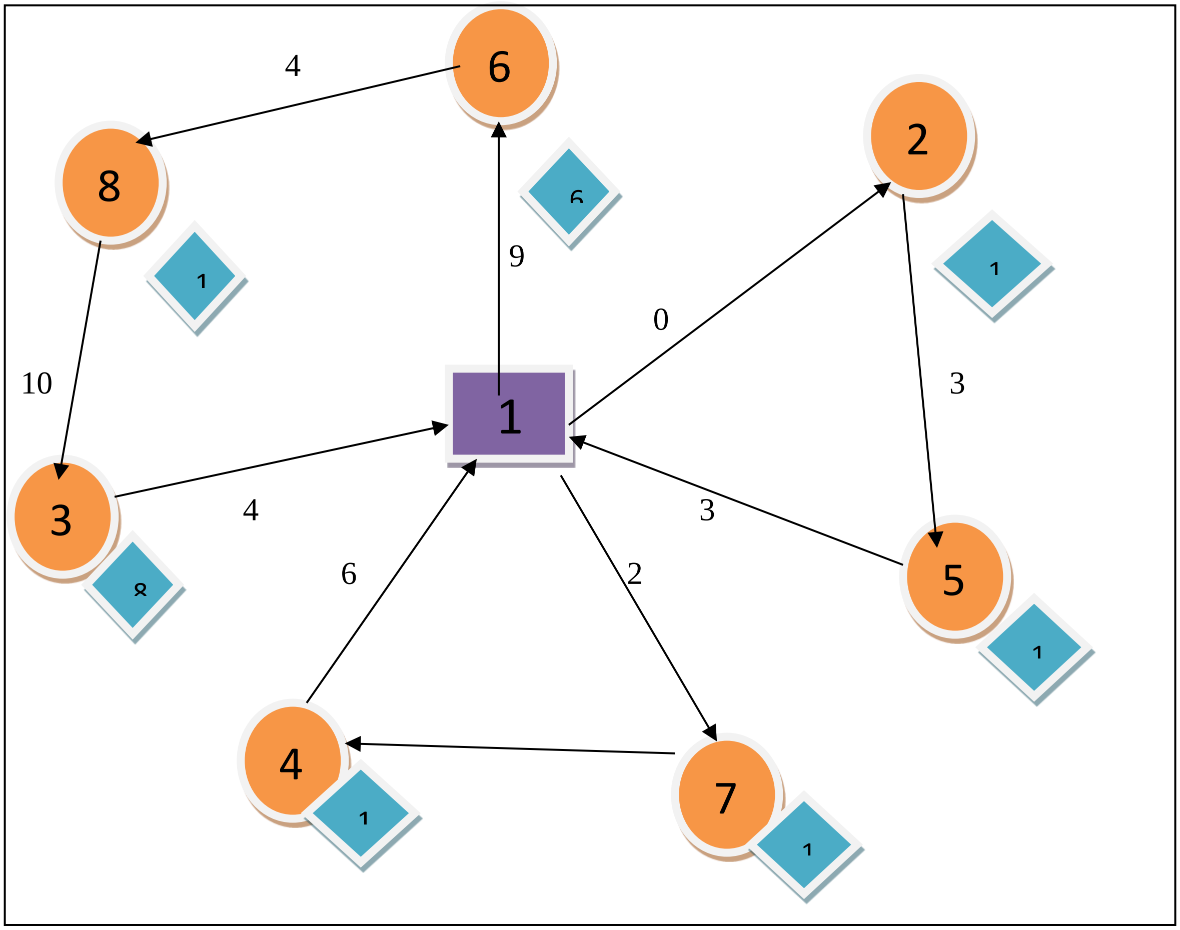


Figure-1 (FEASIBLE SOLUTION)

In the above **Figure-1,** initially, the vehicle has started from the head quarter city 1 to city 2 with capacity of 30 units and there it has supplied the requirement of 10 units. From the city 2, the vehicle travelled to city 5 and supplied the requirement of 10 units and then it has returned to the home city. The 2nd vehicle has started from the home city 1 with capacity of 30 units to city 7; there it supplied the requirement of 14 units. From city 7, the vehicle travelled to city 4; there it has supplied the requirement of 12 units and returned to home city. The 3rd vehicle has started from the home city 1 to destination city 6 with the capacity of 30 units and there it has supplied the requirement of 6 units. From the city 6, the vehicle has travelled to city 8 and supplied the requirement of 10 units. From city 8, the vehicle has reached city 3 and supplied its requirement of 8 units and returned to the home city. The distance/cost of the above mentioned ordered pairs are as follows:

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

= 0 + 0 + 2 + 3 + 3 + 4 + 4 + 6 + 9 + 10

= 41 units.

6. SOLUTION PROCEDURE

In the above **figure-1,** for the feasible solution we observe that **10 ordered pairs** are taken along with the values from the cost/distance matrix for this numerical example in **Table- 1.** The **10 ordered pairs** are selected such that they represents a feasible solution in **figure-1.** So the problem is that we have to select 10 ordered pairs from the cost matrix order [8×8] along with values such that the total cost is minimum and represents a feasible solution. For this selection of 10 ordered pairs we arrange all the ordered pairs with the increasing order of costs/distance 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 ordered pairs { (1, 2), (7, 4), (1, 4), (1, 7), (2, 5), (5,1), (2,3), (3,1), (6, 8), (6, 1) } represents an infeasible solution.

From the above **figure-2**, initially one vehicle has started with capacity of 30 units from the home city and reached to city **2**, there it has supplied the requirement of **10** units. From city **2**, the vehicle travelled to city **5** and supplied the requirement of **10** units and finally returned to the home city. Another vehicle has started from the home city with capacity of **30** units to city **7** and supplied the requirement of **14** units and it has reached city **4** and there it has supplied the requirement of **12** units. One more vehicle has started from the home city with capacity of **30** units and reached city **4** which is contradict hypothesis, because city **4** has got the required capacity of **12** units by the previous vehicle.

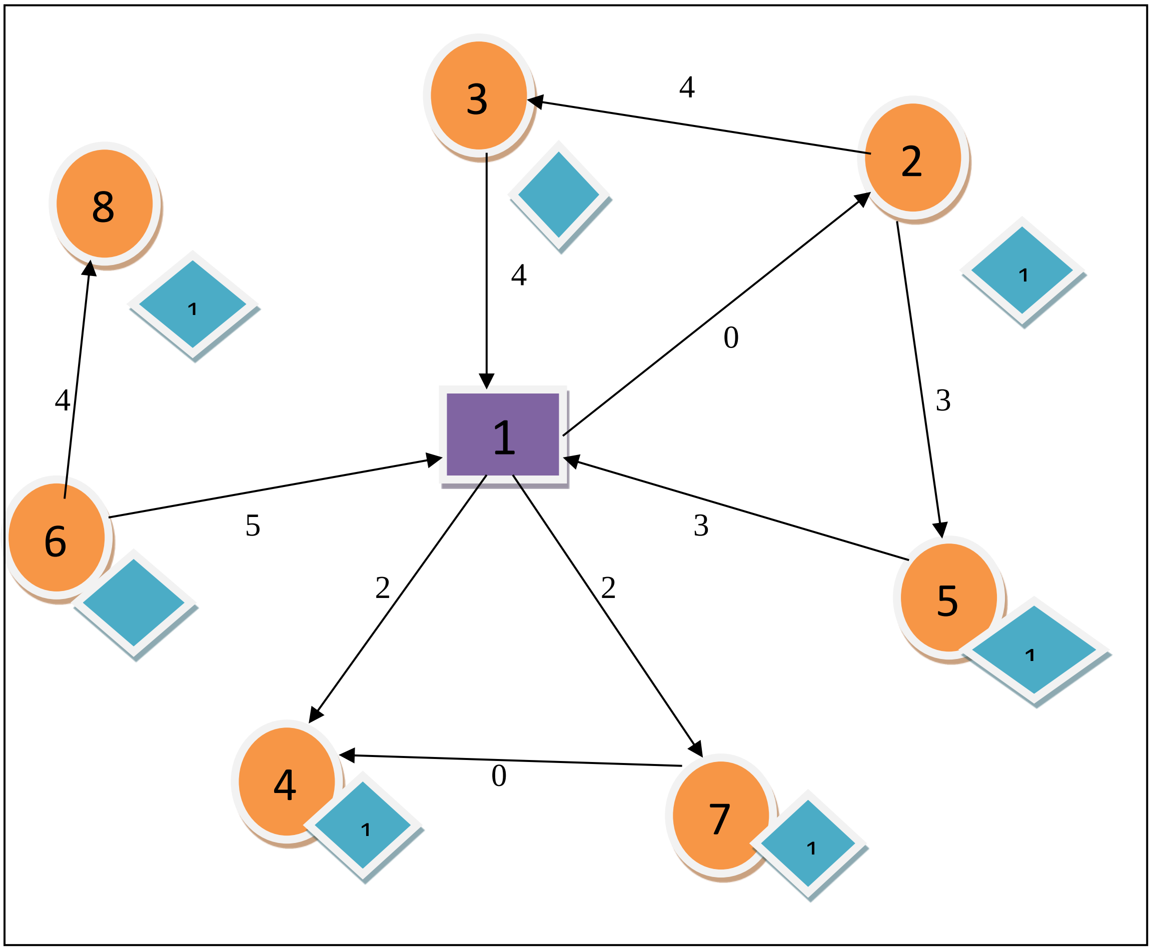


FIGURE-2 (INFEASIBLE SOLUTION)

So, city **4** has visited by the vehicles twice which are contradiction to the fact that each vehicle should visit each city only once. Furthermore, the vehicle has travelled from city **2** to city **3**, which is also a contradiction to the hypothesis, because the vehicle travelled from city **2** to city **5** previously. Moreover the vehicle has travelled from city **6** to city **8** and from city **6** to home city, this is also a contradiction to hypothesis because each vehicle should visit each city only once except home city. It is contradiction to feasible condition. The distance/cost for the above mentioned ordered pairs are follows:

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

= 0 + 0 + 2 + 2 + 3 + 3 + 4 + 4 + 4 + 5

= 27 units.

8. CONCEPTS AND DEFINITIONS

8.1. Definition of a pattern

An indicator two dimensional arrays *X* which gives the supply schedule for requirement of cities 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.

$$V(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} c(i, j)\, X(i, j)$$

The value V(X) gives the total cost of the CVRP of the solution represented by X. Thus the value of the feasible pattern gives the total cost. The pattern represented in **Table-3** is a feasible pattern. The value V(X) gives the total cost of the vehicle routs for the solution represented by X. Thus X is 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, which understanding that the other X (i, j) are zeros.

Consider the set of ordered pairs {(1, 2), (7, 4), (1, 7), (2, 5), (5, 1), (3, 1), (6, 8), (4, 1), (1, 6), (8, 3)} represented by 0 or 1 in matrix X(i, j) indicates the pattern. Here **Table – 3** denotes above pattern which is a feasible solution. According to pattern represented in **figure-1,** satisfies all the constraints in Mathematical Formulation.

Table –3

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

The above **Table-3** represents a feasible pattern for the feasible solution. In the above solution **X (1, 2) = 1**, represents that the vehicle starts from head quarter city with capacity 30 units and reaches to city 2 and supply the requirement of 10 units. **X (2, 5) = 1**, represents that from city 2 the vehicle travels to city 5 and supply the requirement of 10 units. In similar way **X (5, 1) = 1,** represents that the vehicle reaches to home city from city 5. Similarly all the ordered pairs of the feasible solution satisfy the constraints of numerical illustration. So the above solution gives a feasible solution and it shown in **figure-1.**

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

Table –4

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

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

8.2. Alphabet Table

There are $M = n \times n$ ordered pairs in the two-dimensional array C. For convenience these are arranged in ascending order of their corresponding distance and are indexed from 1 to M (Sundara Murthy-1979). Let SN= [1, 2, 3… $n \times n$] be the set of n×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–5.** 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 $DC(a) = \sum_{i=1}^{a} D(i)$.

Then the alphabet of the given cost matrix (Table –1) is as follows.

Table-5: ALPHABET TABLE

SN	D	DC	R	C	SN	D	DC	R	C
1	0	0	1	2	**29**	26	303	8	7
2	0	0	7	4	**30**	30	333	1	3
3	2	2	1	4	**31**	31	364	6	2
4	2	4	1	7	**32**	32	396	1	5
5	3	7	2	5	**33**	34	430	7	8
6	3	10	5	1	**34**	38	468	3	8
7	4	14	2	3	**35**	39	507	2	4
8	4	18	3	1	**36**	42	549	5	7
9	4	22	6	8	**37**	50	599	2	7
10	5	27	6	1	**38**	52	651	8	2
11	6	33	4	1	**39**	53	704	4	8
12	7	40	8	5	**40**	54	758	4	6
13	8	48	5	8	**41**	54	812	5	3
14	9	57	1	6	**42**	56	868	3	6
15	9	66	2	8	**43**	58	926	7	2
16	10	76	5	4	**44**	59	985	4	7
17	10	86	8	3	**45**	60	1045	5	6
18	11	97	8	1	**46**	62	1107	2	6
19	12	109	4	5	**47**	62	1169	6	7
20	12	121	7	1	**48**	63	1232	2	1
21	15	136	6	3	**49**	64	1296	5	2
22	16	152	6	5	**50**	68	1364	3	4
23	17	169	1	8	**51**	70	1434	7	5
24	17	186	7	3	**52**	72	1506	8	6
25	19	205	3	7	**53**	73	1579	3	2
26	23	228	6	4	**54**	74	1653	3	5
27	24	252	7	6	**55**	75	1728	4	2
28	25	277	8	4	**56**	83	1811	4	3

From the above **Table – 5**, Let us consider $15 \in SN$. It represents the ordered pair D (R (15), C (15)) = D (2, 8). Then D (15) = D (2, 8) = 9. i.e., the distance travelled by the vehicle from city 2 to city 8 is 9 units, DC (15) =66.

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. Since the words of length greater than n-1 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-1, and it is a full length word if k = n-1, 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}). For our convince k be the value of unit of D (a_k). 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).

Considered the partial work L_4= (1, 2, 4, 5)

$V(L_k) = V(L_{k-1}) + D(a_k)$

$V(L_4) = V(L_3) + D(5) = 0 + 0 + 2 + 3 = 5$

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 $L_k = \{a_1, a_2, \ldots, a_k\}$ can be defined as follows.

$LB(L_k) = V(L_k) + DC(a_k + n+m-1-k) – DC(a_k)$

Consider the partial word $L_4 = (1, 2, 4, 5)$, $V(L_4) = 5$ then

$$LB(L_4) = V(L_4) + DC(a_4 + 8+3-1-4) – DC(a_4)$$
$$= V(L_4) + DC(5+8+3-1-4) – DC(5)$$
$$= 5 + DC(11) – DC(5)$$
$$= 5+33-7 \quad = 31 \text{units}$$

8.6. Feasibility criterion of a 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 at least 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.

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

IR be an array where IR (i) =1, represents vehicle travels from i^{th} city to other city, otherwise IR (i) = 0.

IC be an array where IC (j) =1, indicates that vehicle reaches j^{th} city from some city, otherwise IC (j) = 0.

SW be an array where SW (i) = j, indicates that i^{th} city is connected to a j^{th} city to supply its requirement, otherwise SW (i) = 0.

SWI be an array where SWI (j) = i, indicates the inverse of the array SW.

The values of the arrays L, IR, IC, SW & SWI are as follows

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

IR (R (a_i)) = 1, i = 1, 2, - - - - - , k and IR (i) = 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

SWI(C (a_i)) = R (a_i), i=1, 2,……, k and SWI(j) = 0 for other elements of j

For example consider a sensible partial word L_5 = (1, 2, 4, 5, 6) which is feasible. The array's L, IR, IC, SW & SWI takes the values represented in **Table –6,** given below.

TABLE-6

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
L	1	2	4	5	6			
IR	1+1	1			1		1	
IC	1	1		1	1		1	
SW	2,7	5			1		4	
SWI	5	1		7	2		1	

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 consider **RA=R (α_k) and CA=C (α_k).**

9. ALGORITHMS

ALGORITHM 1: (Algorithm for feasible checking):

STEP1	:	IS RA = = HQ	IF YES GOTO 5
			IF NO GOTO 2
STEP2	:	IS CA = = HQ	IF YES GOTO 6
			IF NO GOTO 3
STEP3	:	IS IR [RA] = =1	IF YES GOTO 19
			IF NO GOTO 4
STEP4	:	IS IR [RA] = =1	IF YES GOTO 19

			IF NO GOTO 5
STEP5	:	LD=DR (CA)	
		IS X>=LD	IF YES GOTO 6
			IF NO GOTO 17
STEP6	:	W=CA	GOTO 7
STEP7	:	W=SW (W)	
		IS W = =HQ	IF YES GOTO 17
			IF NO GOTO 8
STEP8	:	LD=LD+DR (W)	
		IS X>=LD	IF YES GOTO 9
			IF NO GOTO 19
STEP9	:	IS (W==RA)	IF YES GOTO 16
			IF NO GOTO 10
STEP10	:	IS SW (W)=0	IF YES GOTO11
			IF NO GOTO 7
STEP11	:	W=RA	GOTO 12
STEP12	:	W=SWI (W)	
		IS W = =HQ	IF YES GOTO 17
			IF NO GOTO 13
STEP13	:	LD=LD+DR (W)	
		IS X>=LD	IF YES GOTO 14
			IF NO GOTO 19
STEP14	:	IS (W= =CA)	IF YES GOTO 16
			IF NO GOTO 15
STEP15	:	IS SWI (W) = 0	IF YES GOTO17
			IF NO GO TO 12
STEP16	:	IS IR [HQ] =0	IF YES GOTO 18
			IF NO GOTO 19

STEP 17 : X = 1 GO TO 19

STEP 18 : X=2 GOTO 19

STEP 19 : STOP

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 ak 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 Lk 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.

ALGORITHM 2: (Lexi-Search Algorithm)

STEP 0 : Initialization

The arrays SN, R, C, D, DC and values of N are made available IR, IC, SW and SWI are initialized to zero. The values I=1, J=0, VT=999 and Max=N^2-N, Y = (n-1) + m.

STEP 1 : J=J+1

L [K] = J

LC = 1

IS (J>Max) IF YES GOTO 9

IF NO GOTO 2

STEP 2 : RA= r [J]

CA=c [J];

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

		LB [K] =V [K] +cd [J+n-K]-cd [J]	GOTO 3
STEP 3	:	IS (LB [K] > = VT)	IF YES GOTO 9
			IF NO GOTO 4
STEP 4	:	Check feasible (Using Algorithm 1)	
		IS (IX=0)	IF YES GO TO 1
			IF NO GO TO 5
STEP 5	:	IS (IX=1)	IF YES GO TO 7
			IF NO GO TO 6
STEP 6	:	IS L (K) = = Y	IF YES GOTO 8
			IF NO GOTO 1
STEP 7	:	L [K] = J	
		IC [CA] = 1	
		IR [RA] = 1	
		SWI [CA] = RA	
		SW [RA] = CA	
		K = K + 1	GOTO 1
STEP 8	:	VT = LB [K]	
		L [K] = J	IF NO GOTO 10
STEP 9	:	IS (K= =1)	IF YES GOTO 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 GOTO 1

STEP11 : STOP/END

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 ak 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 Lk 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.

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.

10. SEARCH TABLE

The working details for obtaining an optimal word using above algorithm for the illustrative example is given in the following **Table-7.** The columns named (1), (2), (3), (4), (5), (6), (7), (8), (9) and (10) gives the letters in the first, second, third, fourth, fifth, sixth, seventh, eighth, ninth and tenth 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-7 : Search Table

SN	1	2	3	4	5	6	7	8	9	10	V	LB	R	C	Remark
1	1										0	27	1	2	A
2		2									0	27	7	4	A
3			3								2	27	1	4	R
4			4								2	31	1	7	A
5				5							5	31	2	5	A
6					6						8	31	5	1	A
7						7					12	31	2	3	R
8						8					12	34	3	1	A
9							9				16	34	6	8	A
10								10			21	34	6	1	R
11								11			22	37	4	1	A
12									12		29	37	8	5	R
13									13		30	39	5	8	R
14									14		31	40	1	6	A
15										15	40	40	2	8	R
16										16	41	41	5	4	R
17										17	41	41	8	3	A,VT=41
18									15		31	41	2	8	R,=VT
19								12			23	40	8	5	R
20								13			24	42	5	8	R,>VT
21							10				17	38	6	1	R
22							11				18	42	4	1	R,>VT

23						9					12	38	6	8	A
24							10				17	38	6	1	R
25							11				18	42	4	1	R,>VT
26						10					13	43	6	1	R,>VT
27					7						9	35	2	3	R
28					8						9	39	3	1	A
29						9					13	39	6	8	A
30							10				18	39	6	1	R
31							11				19	43	4	1	R,>VT
32						10					14	44	6	1	R,>VT
33					9						9	44	6	8	R,>VT
34				6							5	35	5	1	A
35					7						9	35	2	3	A
36						8					13	35	3	1	A
37							9				17	35	6	8	A
38								10			22	35	6	1	R
39								11			23	38	4	1	A
40									12		30	38	8	5	A
41										13	38	38	5	8	R
42										14	39	39	1	6	A,=VT
43									13		31	40	5	8	R,>VT
44								12			24	41	8	5	R,>VT
45							10				18	39	6	1	R,=VT
46						9					14	40	6	6	R,>VT
47					8						9	39	3	1	R,=VT
48				7							6	40	2	3	R,>VT
49			5								3	36	2	5	A

50				6							6	36	5	1	A
51					7						10	36	2	3	R
52					8						10	40	3	1	R,>VT
53				7							7	41	2	3	R,>VT
54			6								3	41	5	1	R,>VT
55		3									2	33	1	4	A
56			4								4	33	1	7	A
57				5							7	33	2	5	A
58					6						10	33	5	1	A
59						7					14	33	2	3	R
60						8					14	36	3	1	A
61							9				18	36	6	8	A
62								10			23	36	6	1	R
63								11			24	39	4	1	R,=VT
64							10				19	40	6	1	R,>VT
65						9					14	40	6	8	R,>VT
66					7						11	37	2	3	R
67					8						11	41	3	1	R,>VT
68				6							7	37	5	1	A
69					7						11	37	2	3	A
70						8					15	37	3	1	A
71							9				19	37	6	8	A
72								10			24	37	6	1	R
73								11			25	40	4	1	R,>VT
74							10				20	41	6	1	R,>VT
75						9					15	41	6	1	R,>VT
76					8						11	41	3	1	R,>VT

77				7							8	42	2	3	R,>VT
78			5								5	38	2	5	A
79				6							8	38	5	1	A
80					7						12	38	2	3	R
81					8						12	42	3	1	R,>VT
82				7							9	43	2	3	R,>VT
83			6								5	43	5	1	R,>VT
84		4									2	38	1	7	A
85			5								5	38	2	5	A
86				6							8	38	5	1	A
87					7						12	38	2	3	R
88					8						12	42	3	1	R,>VT
89				7							9	43	2	3	R,>VT
90			6								5	43	5	1	R,>VT
91		5									3	44	2	5	R,>VT
92	2										0	33	7	4	A
93		3									2	33	1	4	R
94		4									2	38	1	7	A
95			5								5	38	2	5	A
96				6							8	38	5	1	A
97					7						12	38	2	3	R
98					8						12	42	3	1	R,>VT
99				7							9	43	2	3	R,>VT
100			6								5	43	5	1	R,>VT
101		5									3	44	2	5	R,>VT
102	3										2	40	1	4	R,>VT

11. COMMENTS

The above **Table-7,** gives optimal solution for the numerical example. At the end of the search the current value of **VT** = **39** and it is the value of the feasible word ***L10=(1, 2 ,4, 6, 7, 8, 9, 11, 12, 14),*** it is given in **42th** row of search table and the corresponding ordered pairs are ***(1, 2), (7, 4),(1, 7), (5, 1), (2, 3), (3, 1), (6, 8), (4, 1), (8, 5)***and ***(1, 6,).*** For this optimal feasible word the arrays L, IR, IC, SW and SWI are given in the following **Table-8.**

Table-8

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**-**	**-**
L	1	2	4	6	7	8	9	11	12	14
IR	1+1+1	1	1	1	1	1	1	1	-	-
IC	1+1+1	1	1	1	1	1	1	1	-	-
SW	-	3	1	1	1	8	4	5	-	-
SWI	-	1	2	7	8	1	1	6	-	-

At the end of the search table the optimum solution value of **VT** is **39** and is the value of optimal feasible word ***L10= (1, 2, 4, 6, 7, 8, 9, 11, 12, 14).*** Then the following **Figure-3** represents the optimal solution to the **CVRP.**

From the above **figure-3,** initially, a vehicle has started from the head quarter city with capacity of **30** units and reached city **2**, there it supplied the requirement of **10** units. From city **2**, the vehicle has travelled to city **3** and supplied the requirement of **8** units finally returned to home city. Another vehicle has started from the home city with capacity of **30** units and reached city **7** and there it has supplied the requirement of **14** units. From city **7**, the vehicle has travelled to city **4** and supplied the requirement of **12** units and finally returned to the home city. One more vehicle has started from the home city with capacity of **30** units and reached city **6**, there it has supplied the requirement of **6** units. From city **6**, the vehicle has travelled to city **8** and supplied the requirement of **10** units. From city **8**, the vehicle has travelled to city **5** and supplied the requirement of **10** units and finally returned to home city. The distance/cost for the above mentioned ordered pairs are follows:

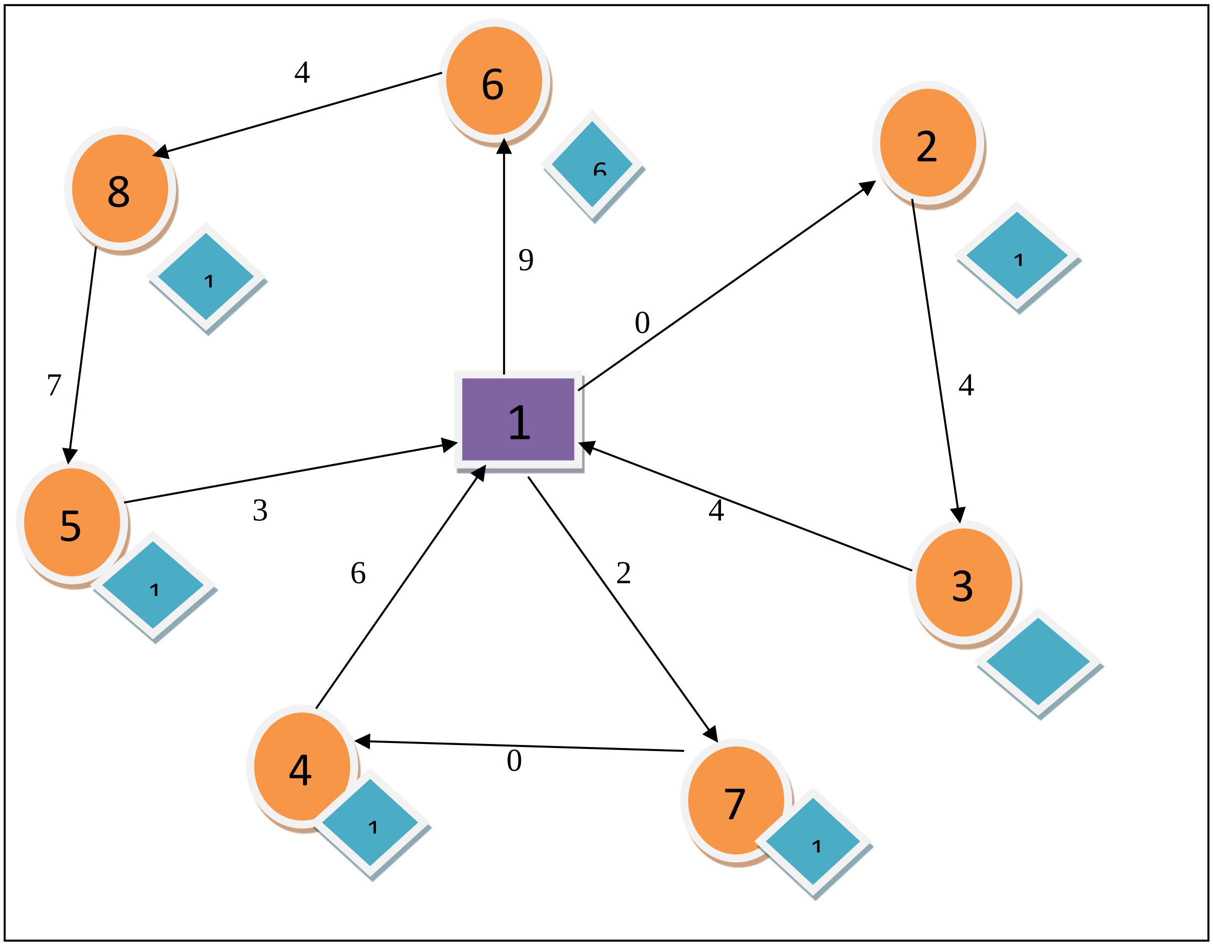


Figure-3 (OPTIMAL SOLUTION)

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

= 0 + 0 + 2 + 3 + 4 + 4 + 4 + 6 + 7 + 9

= 39 units.

Consider the set of ordered pairs {**(1, 2), (7, 4),(1, 7), (5, 1), (2, 3), (3, 1), (6, 8), (4, 1), (8, 5)**, **(1, 6,)}** represented the pattern given in the **Table-9,** which is a feasible solution.

According to the pattern represented in **figure-3,** satisfies all the constraints in Mathematical Formulation.

Table –9

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

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 to N, H and D. The results are tabulated in the **Table -10** are given below. For each instance, five to eight data sets are tested. It is seen that the time required for the search of the optimal solution is fairly less. In the following table microseconds are represented by zero.

Table-10

SN	N	H	D	VT	CPU Run Time in seconds
					Avg. AT + ST
1	5	1	4	23	0.0000
2	7	1	6	28	0.0000
3	8	1	7	54	0.0000
4	10	1	9	62	0.1532
5	12	1	12	65	0.1754

6	14	1	13	179	0.1836
7	15	1	14	142	0.1997
8	17	1	16	156	0.2259
9	18	1	17	194	0.3415
10	19	1	18	172	0.4617
11	20	1	19	185	0.5891
12	21	1	20	189	0.6132
13	22	1	21	196	0.9129
14	23	1	22	201	1.7268
15	25	1	24	224	1.9145

. In the above **Table-10**, SN = serial number, N = number of cities, H= Head quarter/depot, D = number of destinations/cities. 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.

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 awarded thesis of Madhu Mohan Reddy (2015)**.** Then **Table-11** shows that the comparative results of different sizes. In the following table microseconds are represented by zero.

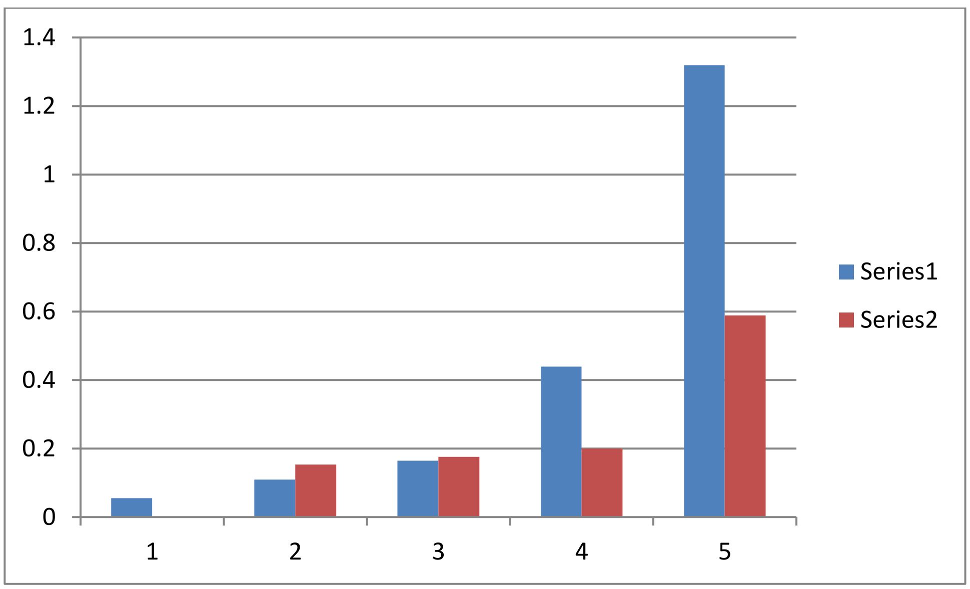
Table-11

SL.NO	N	Published Model	Proposed Model
1	8	0.054945	0.0000
2	10	0.109890	0.1532
3	12	0.164835	0.1754
4	15	0.439560	0.1997
5	20	1.318681	0.5891

In the above **Table- 11**, the last two columns show the CPU run time of published model and proposed model. As compared the two models of sizes N=8, 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, 0.1532sec, 0.1754sec, 0.1997sec & 0.5891 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**, series 2 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 this paper, we have presented an exact algorithm called Lexi-Search algorithm using pattern recognition technique to solve "Solving the Capacitated Vehicle Routing Problem with Lexi-Search Approach**"**. 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] Z.H. Ahmed, A data-guided lexisearch algorithm for the bottleneck travelling salesman problem, Int. J. Oper. Res. 12 (2011), 20-33.

[2] 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.

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

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

[5] 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.

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

[7] M. Gendreau, G. Laporte, J.-Y. 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, (2002), pp. 129–154.

[8] 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.

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

[10] A. Gupta, V. Verma, M.C. Puri, Time-cost trade-off- relations in bulk transportation problem, J. Inform. Optim. Sci. 16 (1996), 317-325.

[11] P. Borovska, M. Lazarova, S. Bahudejla, Strategies for parallel genetic computation of optimization problems, Biotechnol. Biotechnol. Equipment. 21 (2007), 241–246.

[12] 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.

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

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

[15] J. Peter Praveen, R. Donthi, S.V. Prasad, B. Mahaboob B. Venkateswarlu, A glance on the estimation of Cobb-Douglas production functional model, AIP Conf. Proc. 2177 (2019), 020067.

[16] J. Peter Praveen, B. Mahaboob, R. Donthi, S.V. Prasad, B. Venkateswarlu, On stochastic linear regression model selection, AIP Conf. Proc. 2177 (2019), 020068.

[17] B. Mahaboob, J. Peter Praveen, R. Donthi, S.V. Prasad, B. Venkateswarlu, Criteria for selection of stochastic linear model selection, AIP Conf. Proc. 2177 (2019), 020041.

[18] R. Donthi, S.V. Prasad, B. Mahaboob, J. Peter Praveen, B. Venkateswarlu, Estimation methods of nonlinear regression models, AIP Conf. Proc. 2177 (2019), 020081.

[19] R. Donthi, J. Peter Praveen, S.V. Prasad, B. Mahaboob, B. Venkateswarlu, Numerical techniques of nonlinear regression model estimation, AIP Conf. Proc. 2177 (2019), 020082.

[20] B. Mahaboob, S.V. Prasad, J. Peter Praveen, R. Donthi, B. Venkateswarlu, On misspecification tests for stochastic linear regression model, AIP Conf. Proc. 2177 (2019), 020039.

[21] B. Mahaboob, K.A. Ajmath, B. Venkateswarlu, C. Narayana, J.P. Praveen, On Cobb-Douglas Production function model, AIP Conf. Proc. 2177 (2019), 020040.

[22] B. Mahaboob, K.A. Azmath, B. Venkateswarlu, C. Narayana, B.M. Rao, An evaluation in generalized LSE of linearized stochastic statistical model with non-spherical errors, AIP Conf. Proc. 2177 (2019), 020038.

[23] J. Peter Praveen, B. Nageswara Rao, Y. Haranadh, B.V. Appa Rao, C. Narayana, B. Mahaboob, Existence of solution of the nonlinear differential equation in the modeling of eardrum by using homotopy perturbation method, Adv. Math.: Sci. J. 9 (2020), 4853–4862.

[24] B. Mahaboob, J. Peter Praveen, B. V. Appa Rao, Y. Haranadh, C. Narayana, G. Balaji Prakash, A study on multiple linear regression using matrix calculus, Adv. Math.: Sci. J. 9 (2020), 1857-8438.

[25] J. Peter Praveen, B. Nageswara Rao, B. Mahaboob, C. Rajaiah, Y. Harnath, C. Narayana, Series decomposition method for asymmetric nonlinear oscillations, Adv. Math.: Sci. J. 9 (2020), 8069–8076.

[26] B. Mahaboob, K. Sreenivasulu, M. Rajaiah, Y. Harnath, C. Narayana, J. Peter Praveen, Jordan generalized derivations in gamma nearrings, Adv. Math.: Sci. J. 9 (2020), 8341–8347.

[27] J. Peter Praveen, B. Nageswara Rao, B. Mahaboob, B.V. Appa Rao, An application of laplace transform, inverse Laplace transform and Pade's approximant in the periodic solution of duffing equation of motion, Int. J. Emerg. Trends Eng. Res. 8 (2020), 5979-5983.

[28] J. Peter Praveen, B. Nageswara Rao, B. Mahaboob, M. Rajaiah, Y. Harnath, C. Narayana, On the simulation of beck column through a simple Xiong-Wang-Tabarrok experimental model of centripetally loaded column, Int. J. Emerg. Trends Eng. Res. 8 (2020), 5100-5103.

[29] B. Mahaboob, B. Venkateswarlu, G.S.G.N. Anjaneyulu, C. Narayana, A discourse on applications of lie groups and lie algebras, Int. J. Sci. Technol. Res. 9 (2020), 6226-6231.

[30] B. Venkateswarlu, B. Mahaboob, K.A. Ajmath, C. Narayana, Application of DEA in super efficiency estimation, Int. J. Sci. Technol. Res. 9 (2020), 4496-4499.

[31] B. Venkateswarlu, B. Mahaboob, C. Subbarami Reddy, C. Narayana, Evaluating different types of efficiency stability regions and their infeasibility in DEA, Int. J. Sci. Technol. Res. 9 (2020), 3944-3949.

[32] B. Venkateswarlu, B. Mahaboob, C. Subbaramireddy, C. Narayana, Multi-criteria optimization techniques in DEA: methods & applications, Int. J. Sci. Technol. Res. 9 (2020), 509-515.

[33] B. Mahaboob, B. Venkateswarlu, K.A. Azmath, C. Narayana, J. Peter Praveen, On OLS estimation of stochastic linear regression model, Int. J. Eng. Adv. Technol. 8 (2020), 1953-1955.

[34] B. Mahaboob, B. Venkateswarlu, C. Narayana, M. Sivaiah, Bias in the maximum likelihood estimation of parameters of nonlinear regression models, Int. J. Sci. Technol. Res. 8 (2019), 1252-1255.

[35] B. Venkateswarlu, B. Mahaboob, C. Narayana, C. Subbarami Reddy, Evaluation of slack based efficiency of a decision making unit, Int. J. Sci. Technol. Res. 8 (2019), 1178-1182.

[36] B. Venkateswarlu, B. Mahaboob, K.A. Azmath, C. Narayana, C. Muralidaran, An application of linear programming in the estimation of technical efficiency of DMU, Int. J. Eng. Adv. Technol. 8 (2019), 1956-1959.

[37] B. Venkateswarlu B. Mahaboob, C. Subbarami Reddy, C. Narayana, New results production theory by applying goal programming, Int. J. Sci. Technol. Res. 8 (2019), 1918-1923.

[38] B. Mahaboob, B. Venkateswarlu, C. Narayana, J. Ravi sankar, P. Balasiddamuni, A treatise on ordinary least squares estimation of parameters of linear model, Int. J. Eng. Technol. 7 (2018), 518-522.

[39] B. Mahaboob, B. Venkateswarlu, C. Narayana, J. Ravi Sankar, P. Balasiddamuni, A monograph on nonlinear regression models, Int. J. Eng. Technol. 7 (2018), 543-546.

[40] B. Venkateswarlu, B. Mahaboob, J. Ravi Sankar, C. Narayana, B. Madhusudhana Rao, An application of goal programming in data envelopment analysis, Int. J. Eng. Technol. 7 (2018), 523-525.

[41] C. Narayana, B. Mahaboob, B. Venkateswarlu, J. Ravi Sankar, A memoir on model selection criterion between two nested and non-nested stochastic linear regression models, Int. J. Eng. Technol. 7 (2018), 529-531.

[42] C. Narayana, B. Mahaboob, B.Venkateswarlu, J. Ravi Sankar, P. Balasiddamuni, A study on misspecification and predictive accuracy of stochastic linear regression models, Int. J. Eng. Technol. 7 (2018), 532-535.

[43] C. Narayana, B. Mahaboob, B. Venkateswarlu, J. Ravi Sankar, P. Balasiddamuni, A discourse on modified likelihood ratio (LR), Wald and Lagrange multipliers (LM) tests for testing general linear hypothesis in stochastic linear regression model, Int. J. Eng. Technol. 7 (2018), 536-538.

[44] C. Narayana, B. Mahaboob, B. Venkateswarlu, J. Ravi Sankar, P. Balasiddamuni, A treatise on testing general linear hypothesis in stochastic linear regression model, Int. J. Eng. Technol. 7 (2018), 539-542.

[45] B. Mahaboob, B. Venkateswarlu, J. Ravi Sankar, Estimation of parameters of constant elasticity of substitution production functional model, IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042121.

[46] B. Mahaboob, B. Venkateswarlu, J. Ravi Sankar, P. Balasiddamuni, Computation of nonlinear least squares estimator and maximum likelihood using principles in matrix calculus, IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042125.

[47] B. Venkateswarlu, B. Mahaboob, C. Subbarami Reddy, J. Ravi Sankar, A study on technical efficiency of a DMU (review of literature), IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042124.

[48] B. Mahaboob, B. Venkateswarlu, G. Mokeshrayalu, P. Balasiddamuni, New methods of testing nonlinear hypothesis using iterative NLLS estimator, IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042126.

[49] B. Venkateswarlu, B. Mahaboob, C. Subbarami Reddy, B. Madhusudhana Rao, Fitting of full Cobb-Douglas and full VRTS cost frontiers by solving goal programming problem, IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042127.

[50] B. Mahaboob, B. Venkateswarlu, G. Mokeshrayalu, P. Balasiddamuni, A different approach to estimate nonlinear regression model using numerical methods, IOP Conf. Ser.: Mater. Sci. Eng. 263 (2017), 042122.

[51] K. Lakshmi, B. Mahaboob, M. Rajaiah, C. Narayana, Ordinary least squares estimation of parameters of linear model, J. Math. Comput. Sci. 11 (2021), 2015-2030

[52] B. Venkateswarlu, M. Mubashir Unnissa, B. Mahaboob, Estimating cost analysis using goal programming, Indian J. Sci. Technol. 9 (2016), 1-10.