Anticipatory Bound Selection Procedure (ABSP) for Vertex K-Center Problem

Rattan Ran and Deepak Garg
Computer Science and Engineering Department, Thapar University, India

Abstract: *Vertex k-center problem introduces the notion to recognize k locations as centers in a given network of n connected nodes holding the condition of triangle inequality. This paper presents an efficient algorithm that provides a better solution for vertex k-center problem. Anticipatory Bound Selection Procedure (ABSP) is deployed to find the initial threshold distance (or radius) and eradicating the gap between minimum distance (lower bound) and maximum distance (upper bound). The jump based scheme is applied to find the optimal coverage distance. Main aspect of this algorithm is to provide optimal solution with less iteration which has not been achieved so far.*

Keywords: *Facility planning and design, facility location allocation, k-center, set covering problem.*

Received February 21, 2012; accepted March 19, 2013

1. Introduction

Facility location is a crucial problem having an extensive range of sub problems that have been scrutinized within various fields including operation research, computational geometry, data analysis, and graph theory. This area of research has a long history and persistent activity. The motive is to achieve the best location of facilities in a network in realistic situations such as placing the ambulance services, fire stations, workstations and many more. It covers both the public and private sectors.

The present world scenario reveals the technological revolution which results the industrial and economical expansion. There is a challenge to identify appropriate locations in a wide area, where such facilities can be located. Keeping in view the necessity of this subject a well defined area of research has been established in last decades.

Facility location problems proposed in operations research provide mathematical formulations of the common optimization aspects of these problems. As depicted in Figure 1, the problem under consideration i.e. k-center problem is the sub-model of discrete location model [6]. The main objective of k-center problem is to minimize the maximum distance between a demand node and its nearest facility. There are two types of *k*-center problem, namely "absolute k-center problem" and "vertex k-center problem" [5]. In former case we locate these facilities anywhere on the network or on any edge whereas later one implies a condition that the facility can be located only on vertex not on edge. This research focuses on vertex *k-center* problem which is part and parcel of the facility location problem.

To understand the facility location problem let us have an example of an automobile manufacturing company, which wants to open its *k* workstations (*k* is an integer) over a given wide area to provide the services to their customers. Now, the problem is to identify such *k* locations in given area to locate the workstation (facility center) so that the maximum population can avail the facility without traveling a long distance keeping in view that the number of workstations are limited. This can be solved using set covering models but the *k-center* problem reveals the internal complication of facility location problem that is to minimize the largest distance of node to its nearest center.

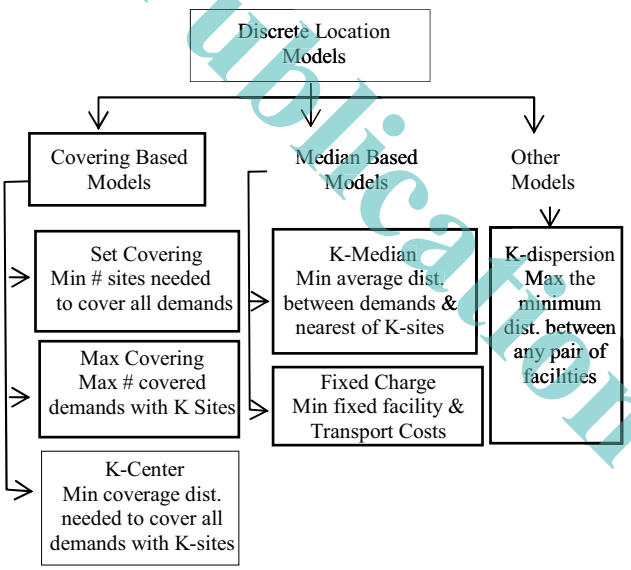

Figure 1. Breakdown of discrete location models [6].

Several techniques have been developed and applied to provide an exact solution of problem. Various approaches have been presented for extending the

solution quality [1, 3, 4, 8], however still there is scope to improve.

Vertex *k-center* problem can be represented as follow: Let G = (V, E) is a weighted graph having $V = \{v_1, v_2, v_3 \ldots v_n\}$ as a set of n vertices and $E = \{e_1, e_2, e_3 \ldots e_m\}$ as a set of m edges.

For any two vertices v_i and v_j in V the distance $d(v_i, v_j)$ is defined to be the length of the shortest path linking v_i and v_j. The objective is to find a subset S of V which implies that $|S| \leq K$, where K is the number of facilities to be identified. The objective is shown in equation 1.

$$Minimize \ \max_{v \in V} d(v,S) \qquad (1)$$

This paper is structured into seven major sections: section 2 presents literature survey and highlights substantial efforts made by numerous researchers in the direction to acquire an optimal solution. Section 3 provides the formulation of vertex *k-center* problem. The section 4 illustrates proposed algorithm and section 5 describes the methodology opted to solve the problem. Section 6 provides an intensive discussion on results. Last but not the least section 7 provides conclusion as well as an insight in to the future work.

2. Literature Survey

The vertex *k-center* problem has received considerable attention both in computer science and operation research literature. This section provides a brief review of extensive literature of vertex *k-center* problem. Sylvester was the first person who has worked out on *1-center* problem in [13]. Minieka [10] solved *k-center* problem by choosing a threshold distance (radius) to check whether the all demand points are covered within this distance using no more than P facilities. Hochbaum and Shmoys [8] presented a 2-approximation algorithm for *k-center* problem with triangle inequality and provided an interesting insight to the problem and given $O(|E|log|E|)$ time solution with value no more than twice the *k-center* problem.

Plesnik [11] generalized the results of Hockbaum and Shmoys, and developed a polynomial algorithm with a worst case error ratio of 2. Plesnik described it for the *p-center* problem in connected graphs with edge lengths and vertex weights. In the subsequent publication Plesnik [12] has proved that the ratio can be arbitrarily large in case of multi-centers.

Based on Minieka's idea, Daskin [5] has solved the vertex *p*-center problem using set covering model for locating centers and bisection method to find the optimal coverage distance. Although, this algorithm produced wonderful results but the algorithm consumed considerable amount of CPU time. Further the algorithm could resolve the problem in polynomial time only for the fixed values of P. Though, vertex *p*-center problem is NP hard for the variable values of P. In July 2000, Daskin has developed another algorithm to solve the same problem [4]. Now he has made a few modifications in his previous algorithm like replacement of the set covering model or Set Covering Problem (SCP) with maximal covering model and applied Lagrangian relaxation to solve problem optimally. But Elloumi et al. [7] has solved it by greedy heuristic and the IP formulation of the sub problem of Minieka's. Ilhan and Pinar [9] have developed an appealing 2-phase approach to solve the vertex *p*-center problem optimally for a specific covering radius.

Al-Khedhairi [1, 2, 3] had put forward several improvements to the well-known Daskin's and Ilhan's algorithms. The intention behind these modifications is to reduce the number of iterations needed to find the optimal solution as well as to shrink the gap between lower and upper bounds to solve the vertex *p*-center problem. The modified algorithm has shown encouraging results.

A critical look at the literature presented above indicates that although a lot of work has been done towards finding an optimal solution of vertex *k-center* problem and most of them suffer from the curse of not only huge gap between initial lower and upper bounds but also, slow in execution. Hence an algorithm that shrinks the mentioned gap and also, speed up the process in strongly desired. Therefore, this work aims to achieve the above stated objective by proposing Anticipatory Bound Selection Procedure and Jump Based Approach as illustrated in the upcoming section.

3. Formulation of Vertex K-Center Problem

To formulate the vertex *k-center* problem in adequate manner it's indispensable to presume basic entities listed as follows:

X and Y be the sets of demand nodes and candidate sites respectively i.e. $X=\{1\ldots, N\}$ $Y=\{1\ldots, M\}$.

d_{xy}=is the distance between demand node and candidate site where $x \in X$, $y \in Y$

K=Total number of facilities to be identified

$$F_y = \begin{cases} 1 & \text{if the facility is located at site } y \in Y \\ 0 & \text{otherwise} \end{cases}$$

A_{xy}=1 if demand node $x \in X$ is assigned to open facility at candidate site $y \in Y$, otherwise

A_{xy}=0

H=the maximum distance between a demand node and its nearest facility

Now the formulation of vertex *k-center* problem can be presented as given below:

$$Minimize \ H \qquad (2)$$

Subject to:

$$\sum_{y \in Y} A_{xy} = 1 \qquad \forall x \in X \qquad (3)$$

$$A_{xy} \leq F_y \qquad \forall x \in X, y \in Y \qquad (4)$$

$$\sum_{y \in Y} F_y = K \qquad (5)$$

$$\sum_{y \in Y} A_{xy} d_{xy} \leq H \qquad \forall x \in X \qquad (6)$$

where A_{xy}, F_y can have only binary values such as: A_{xy}, $F_y = \{0, 1\}$ $\forall x \in X$, $y \in Y$ and also, known as integrity constants.

Equation (2) characterizes the objective function that is to minimize the maximum distance between a demand node and its nearest facility. Subject to 1st constraint as in equation (3), that guarantees that each demand node should be assigned to exactly one facility. Although, 2nd constraint provides a check on each demand node that it should be assigned to only selected candidate site as in equation (4). The 3rd constraint ensures that the total number of facilities located must be equal to K as in equation (5). Finally the 4th constraint stipulates that the maximum distance between any demand node and facility should be less than H as in equation (6). For more details readers are referred [3, 4, 8].

4. Proposed Algorithm for Vertex K-Center Problem

The existing algorithm due to Daskin [4] proposes to solve the vertex *p-center* problem using the bisection (binary search) method to find the optimal distance value. But the practical implication of this method is that it generates a swing in results. Whereas the presented algorithm is implemented using such a new and effective approach that make the algorithm to give the desired results smoothly in less iterations. Complete steps of proposed algorithm are given as under

Step 1: Initialize max[];
With maximum value of distance from each row.
Step 2: Sort max[];
Keep unique values only
Step 3: Set L= Minimum value in max[i];
U= Maximum value in max[m]
Where i=0,1,2……m
Step 4: Set D=L initially i=0
Step 5: Solve SCP for the coverage distance D and let p represents the number of facilities found
a. If p>K then set D=(max[i+2]) and repeat step5.
b. If p<K then set U=D, L=D-1, D=L, and repeat the step 5.
c. If p=K then feasible, temp=D; temp is coverage distance and goto stop step 6.
Step 6: Set L=D-1, U=D and D=L. Solve SCP.
a. If p > K then infeasible and stop. temp is the optimal covering distance.
b. If p=K then feasible, temp=D; Go to Step6.

4.1. Description of Proposed Algorithm

The *max*[] an array having all maximum distance values of each row in distance matrix as elements. D is the coverage distance. The variable *temp* keeps the value of D temporarily. The complete flow of algorithm is shown through a state diagram in figure 2.

- *Step 1:* is to initialize the array *max*[], which holds the maximum distance values of each row of distance matrix.

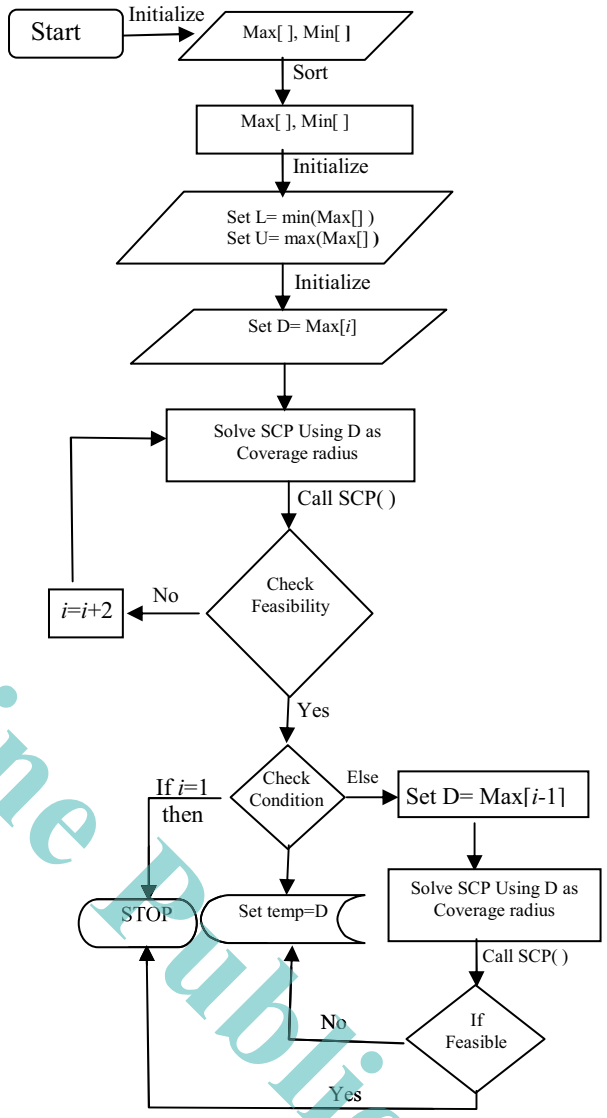

Figure 2. Flow diagram of proposed algorithm.

- *Step 2*: is to sort the elements of *max*[] in ascending order to obtain the minimum distance value among the stored values . To sort the array quick sort is used. It is set of selected maximum values of distance matrix that helps in smooth search of optimal distance as well as reduces the search efforts unlikely to binary search.
- *Step 3*: initializes the value of lower and upper bounds. Lower bound is initialized with the minimum value in *max*[], and upper bound is initialized with maximum value in *max*[].
- *Step 4*: is to initialize the coverage distance D with first element of *max*[].
- *Step 5*: is used to solve set covering problem with the help of a subroutine SCP() and returns p as

number of facilities found in network. If $p>K$ then not feasible and set D=max[i+2] repeat step 5 until it is feasible, If it is feasible for the D and i=1 then Stop; D is the optimal coverage distance else temp=D go to the step6.

- *Step 6:* checks the feasibility at one step backward if $i \neq 1$. Set D=max[i-1] and perform SCP(), if $p>K$ then not feasible, Stop and *temp* is the optimal coverage distance else *temp*=D and *temp* is the new coverage distance.

5. Methodology Employed

This section deals with a comprehensive description of the entire models and approaches deployed, to provide evidence that the proposed algorithm is as efficient as expected.

5.1. Anticipatory Bound Selection Procedure (ABSP)

Before illustrating the Anticipatory Bound Selection Procedure, it's required to understand the reasons of why ABSP. Why ABSP? The critical literature survey reveals that most of the algorithms are using bisection method to compute the initial threshold distance. It also, presumes zero as lower bound (minimum distance) as shown in figure 3(a). While the bisection method has a very serious problem that it generates simple harmonic motion. Additionally, it also, yields such distance values, that doesn't exist in distance matrix. It takes superfluous time to reach the exact distance value. Eventually, it increases the number of iterations as well as decreases the gap between lower and upper bounds gradually. Hence it's required at the moment to design such a procedure that can yield an initial threshold distance which directs the algorithm to move in either forward or backward direction or nearby the optimal coverage distance. To overcome all these shortcomings ABSP is designed.

The ABSP reduces the substantial gap between lower and upper distance values by computing a distance value (i.e. lower bound), that is neither the zero nor the mean value. It shifts the lower bound from zero to some appropriate distance value as shown in figure 3(b).

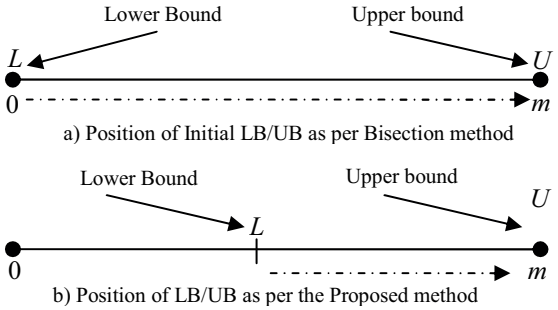

Figure 3. Illustrating position of Initial Lower/Upper Bounds in bisection method and proposed method.

5.1.1.Operational Details of ABSP

The anticipatory bound selection is a two step procedure employed to shrink the gap between lower and upper bounds. This procedure helps out to change the traditional lower bound selection procedure (i.e. zero). The first step is used to lower distance find the maxima i.e. LUB's (least upper bound of each row of the distance matrix). Second step provides minima i.e. GLB (greatest lower bound) it is used to determine the greatest lower bound of all LUB's. Quick sort technique is applied to sort the elements known as GLB's. The operational details of this procedure are illustrated as below:

- *Step 1* Finding maxima (or LUB's):Suppose we have a distance matrix A of order $n \times n$. Also, we have an array max[].

$$A = \begin{vmatrix} d_1^1 & d_2^1 & d_3^1 & d_4^1 \ldots\ldots\ldots & d_n^1 \\ d_1^2 & d_2^2 & d_3^2 & d_4^2 \ldots\ldots\ldots & d_n^2 \\ d_1^3 & d_2^3 & d_3^3 & d_4^3 \ldots\ldots\ldots & d_n^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_1^n & d_2^n & d_3^n & d_4^n \ldots\ldots\ldots & d_n^n \end{vmatrix}$$

Let α_1, α_2, α_3, …….α_n be the maximum elements of 1st, 2nd ,3rd ……n^{th} rows respectively. Therefore, we have a sub set S_d having all the elements with highest distance value in each row of distance matrix A.

$$S_d = \{\alpha_1, \alpha_2, \alpha_3, \ldots\ldots\alpha_n\}$$

Since, this phase unearth the maximum distance value of each row of distance matrix, that's why it is termed as maxima.

- *Step 2* Finding Minima (or GLB):Now, we have to find the minimum element out of these maxima's that's why this is termed as minima. To find the minima following steps are to be followed:

  a. $max[] \leftarrow S_d$
  b. Remove all duplicate values max[] and maintain the unique distance values in ascending order. and we have $max[] = \{\alpha_1, \alpha_2, \alpha_3, \ldots\ldots\alpha_m\}$, Where $m \leq n$
  c. Now Set L=max[i] and U= max[m] Where i = 0, 1, 2……m

Thus, we can get the lower bound. The entire process exhibits that we have considered only those values, which are present in distance matrix to choose the lower bound as well as initial threshold distance, unlike the bisection method. A tight upper (U) and lower bound (L) is the major strength of proposed algorithm.

5.2. Jump Based Scheme (JBS)

The jump based scheme is deployed to reduce the number of iterations needed to find the optimal coverage distance. Consequently, it speeds up the process. To substantiate the above mentioned statement, the functional detail and results produced by JBS are given below as evidence.

5.2.1. Functional Details of JBS

As stated above that the JBS is employed to reduce the number of iterations. The JBS implied in collaboration of ABSP. ABSP supplies *max*[] an array of elements (containing distinct maximum distance values of each row of distance matrix) as an input to the JBS. Now JBS selects the first element of *max*[] array as initial threshold distance and passes it to the SCP function as distance parameter to check the feasibility.

If solution is infeasible then it jumps to the second next value and repeat the process until feasible solution is reached. As soon as the feasible solution is attained then it checks the feasibility at preceding distance value. Because of jumping nature the preceding value is unchecked so, the feasibility is checked for this distance. If feasible then this process is repeated until infeasible state is reached. As soon as it gets infeasible the last feasible solution is the optimal coverage distance.

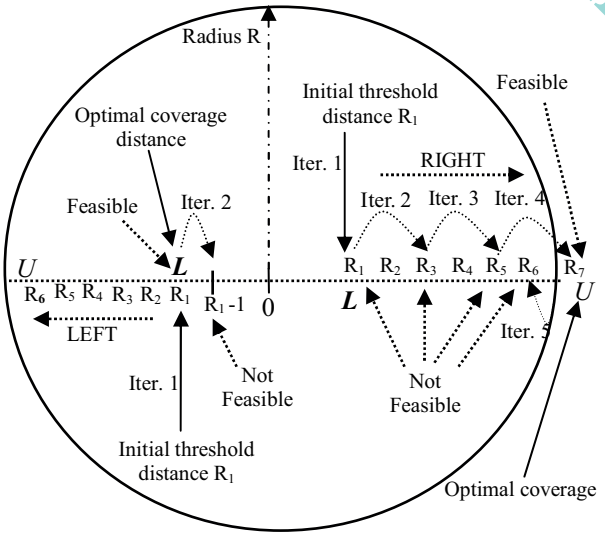

Figure 4. Working of jump based scheme.

Figure 4 illustrates the working traditions of JBS effectively. Figure 4 showing a big circle having LEFT half and RIGHT half. The LEFT half showing the situation when the initial threshold distance R_1 or *L* is passed to the SCP() for feasibility check. Since it is feasible, now it'll check for the preceding distance value i.e., R_1-1. Figure shows that it is not feasible for R_1 -1, therefore the R_1 is the optimal coverage distance. Similarly, the RIGHT half is presenting another look of same scheme as shown in figure 4, initially *L* is set to be the threshold distance and passed to the SCP() to check the feasibility. If it is feasible then the former case will be repeated till it is infeasible. Otherwise, *D* is set to be the second next distance value and procedure is repeated until it is feasible. As soon as it reaches to the feasible state then the preceding distance value is set to *D* and repeat the process until it is infeasible. Where it is infeasible; the former distance value is optimal coverage distance as shown in figure4.

The answer of question is very straight forward that jump based method is used to save the iteration. It has been proved by Al-Khedhairi [1, 3]. The analytical analysis of jump based method can be generalized as, if any algorithm needs 5 or more iteration to perform a particular task then jump based method takes less number of iterations. The result of jump base method shows [1] improvement in performance of algorithm. At the same time Al-Khedhairi said that the jump based method does not shows any improvement in cases where less than 5 iterations are required.

In general wherever large numbers of iterations are required this method has proven its worth to reduce the number of iterations.

6. Results and Discussion

The major achievement of this algorithm is reduced number of iterations required to locate the requisite facilities. Presented algorithm is implemented on more than 30 problems having range of nodes 9 to 90 and facilities 3 to 58. Program executed on dual core 1.6 GHz Microprocessor with 1GB RAM and code is written in Turbo C++. Results are shown in table 1.

Table 1. Comparative Results.

N	K	Proposed Algorithm		Daskin's Algorithms	
		#Iter.	Optimal value	#Iter.	Optimal value
9	3	3	6	6	6
9	4	2	4	5	4
9	6	3	3	5	3
13	3	4	13	6	20
13	4	2	10	6	10
13	7	4	8	6	8
13	9	7	5	9	5
13	10	8	4	6	4
20	6	12	12	6	12
20	7	3	7	6	12
20	10	4	6	6	6
20	12	2	5	5	5
20	17	3	4	6	4
25	7	16	28	7	28
25	8	23	21	8	21
25	9	5	18	6	18
25	10	6	17	7	17
25	12	7	16	7	16
25	15	2	15	6	15
25	18	4	13	6	13
25	22	7	10	6	10
50	22	4	22	6	22
50	23	2	20	7	20
50	28	3	19	6	19
50	29	4	18	7	18
50	32	5	17	7	17
50	34	7	15	7	15
50	42	8	14	7	14
50	45	10	12	6	12
75	45	11	12	13	12
75	48	11	13	13	13
90	50	13	15	15	15
90	55	12	16	15	16
90	58	13	16	14	16

It illustrates the results obtained through execution of proposed algorithm using ABSP and most famous Dsaskin's algorithm. Column 1 shows the number of nodes in network; Column 2 presents the number of centers to be located, column 3 and 5 showing the number of iterations taken by proposed algorithm and Daskin's algorithm to locate the required number of centers while the column 4 and 6 is showing optimal distance value. Excluding a few cases, proposed algorithm has completed most of cases within fewer numbers of iterations than existing algorithm. Figure 7 exhibits the results of table 1 in graphical form.

The overall impact of using Anticipatory Bound Selection Procedure in proposed algorithm makes it robust as well as efficient than the existing methods. Although the value of optimal coverage distance determined by both the algorithms in most of cases is equal but in a few cases, our approach showing better results. In other words, the results of proposed algorithm substantiated the significance of being a real algorithm for vertex *k-center* problem.

6.1. Analysis

It is essential to perform some analytical technique to justify the results obtained and methodology employed. On the basis of results obtained from both the algorithms an iteration frequency analysis is done. The analysis results are shown in table 2 and table 3. This iteration frequency analysis revealed that the proposed algorithm starts providing results just in the second iteration and most of cases are solved within 4 or 5 iterations except a few cases.
While the Daskin's algorithm starts providing optimal distance value by exhausting at least five iterations or more in most of cases.

Table 2. Frequency table for daskin's algorithm.

Daskin's Algorithm				
Iterations	**Freq.**	**Freq. %age**	**Cumulative Frequency**	**Cumulative Frequency Percentage**
5	3	8.82	5	3
6	19	55.88	6	19
7	27	79.41	7	27
8	28	82.35	8	28
9	29	85.29	9	29
13	31	91.18	13	31
14	32	94.12	14	32

Table 3. Frequency table for proposed algorithm.

Proposed algorithm				
Iterations	**Freq.**	**Freq. %age**	**Cumulative Frequency**	**Cumulative Frequency Percentage**
2	5	14.71	5	14.71
3	5	14.71	10	29.42
4	6	17.65	16	47.06
5	2	5.88	18	52.95
6	1	2.94	19	55.89
7	4	11.76	23	67.65
8	2	5.88	25	73.53
10	1	2.94	26	76.47
11	2	5.88	28	82.36
12	2	5.88	30	88.24
13	2	5.88	32	94.12
16	1	2.94	33	97.06
23	1	2.94	34	100.00

It is clearly shown in column cumulative frequency percentage of table 2 and table 3 that the Daskin's algorithm completes only 8.82 percent of jobs within the range of 5 iterations while the proposed algorithm completes 52.95 percent. It is noticeable in both the tables that there is no iteration less than 5 in the first column of table 2 while in table 3 there is the minimum iteration number that is 2 with frequency 5. Moreover the cumulative frequency of iteration 4 is 16 in table 3 it's an evidence that more than 47 percent of work is done by the proposed algorithm by exhausting at most of 4 iterations.

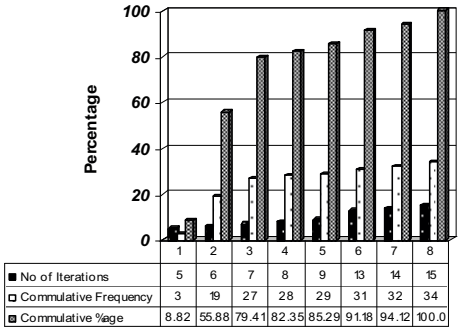

Figure 5. Cumulative frequency analysis for daskin's algorithm.

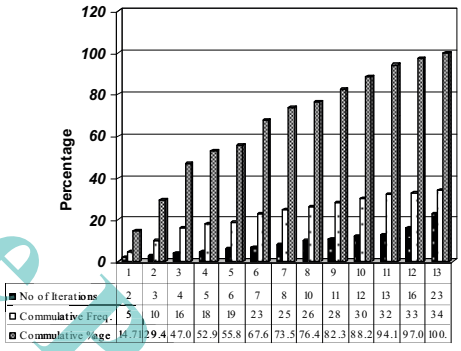

Figure 6. Cumulative frequency analysis for proposed algorithm.

The difference in performance of both the algorithms is visible. Approximately 50% of work has been completed by the proposed algorithm while the Daskin's algorithm could not cover even a single case up to the fourth iteration. Daskin's algorithm exhausts at least five iterations to solve any of given problem. A graphical representation of this analysis is also, presented through the Figure 5 and 6.

6.2. Discussion

As already mentioned, that the success of presented algorithm is in the reduced number of iterations. It is just because of methodology opted to select the initial threshold distance i.e. ABSP & JBS as described in section 4. The overall impact of presented methodology is that many cases are able to complete in smaller number of iterations. Since the bisection method iteratively half the distance values to reach towards optimal distance while the presented methodology selects the maximum distance values and checks the possible feasible solution in just selected

values, additionally the Jump Based Scheme speed up the searching process. Collectively, it returns the result in comparatively less iterations.

Actually this is one aspect of algorithm while the another important aspect is that the selection of these maximum distance values (as described in section 5.1) is a time consuming process, consequently it increases the overall time complexity of algorithm.

Although, this paper highlights only one significant facet of algorithm that it is efficient enough to reduce the iterations. Subsequently, the intention is to reduce the time complexity of algorithm by establishing an appropriate counterpart.

7. Conclusion and Future Work

The proposed algorithm eliminates the swings in distance values (D) with the help of Anticipatory Bound Selection Procedure and provides a smooth approach to move towards the optimal coverage distance. Jump Based Scheme played a remarkable role in reducing iterations. Eventually, the results produced are beyond expectations but not the time complexity is yet to be improved. Even though, the rest of results produced by the proposed algorithm are well simulated and reliable.

The subsequent work will be supplement of present work and cover all remaining aspects belonging to the vertex *k-center* problem. The future work will focus on improving the time complexity of algorithm. It is also, an open problem for the researchers working in same direction.

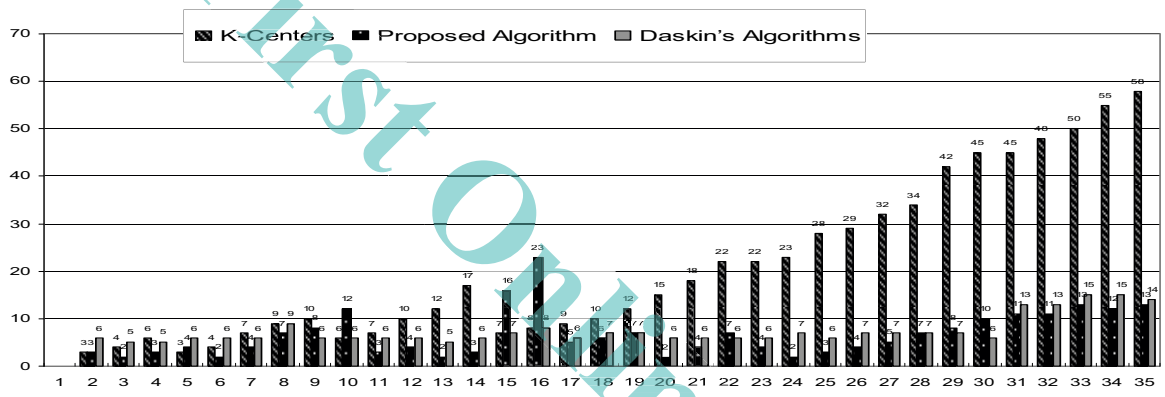

Figure 7: Showing Iterative Deviations for Daskin and Proposed Algorithm.

Acknowledgements

We would like to acknowledge the support of Mr. Vijay Kumar Saini, Dr. Kanwal Garg and Dr. Dimple Juneja.

References

[1] Al-khedhairi A. and Salhi S., "Enhancement to Two Exact Algorithms for Solving the Vertex P-center Problem," *JMMA*, vol. 2, no. 2, pp. 129-147, 2005.

[2] Al-khedhari A., "An Enhancement of Daskin's Algorithm for Solving p-center Problem," *Journal of Approximation Theory and Applications*, vol. 2, no. 2, pp. 121-131, 2007.

[3] Al-Sadi J. and Awaad A., " A New OTIS-Arrangement Interconnection Network," *The International Arab Journal of Information Technology (IAJIT)*, vol. 8, no. 4, pp. 376-382, 2011.

[4] Daskin M., "A New Approach to Solve the Vertex P-center Problem to Optimality: Algorithm and Computational Results," *Communications of the Operations Research Society of Japan*, vol. 45, no. 9, pp. 428-436, 2000.

[5] Daskin M., *Network and Discrete Locations: Models, algorithms, and applications*, John Wiley & Sons, New York, pp. 154-191, 1995.

[6] Daskin M., *What You Should Know About Location Modeling*, Wiley Periodicals, pp. 283-294, 2008.

[7] Elloumi S., Labbe M., Pochet Y., "New Formulation and Resolution Method for the P-center Problem," *http:/ www. Optimization-online.org/DB_HTML/2001/10/394.html*, 2001.

[8] Hochbaum D-S and Shmoys D-B., "A Best Heuristic for the k-center Problem," *Math. sOperations Research*, vol. 10, pp. 180-184, 1985.

[9] Ilhan T., and Pinar M.C., "An Efficient Exact Algorithm for the Vertex p-center Problem," 2001.

[10] Minieka E., "The M-Center Problem," *SIAM Review*, vol. 12, pp. 138-139, 1970.

[11] Plesnik J., "A Heuristic for The P-Center Problem in Graphs," *Discrete Applied Mathematics*, vol. 17, pp. 263-268, North Holland, 1987.

[12] Plesnik J., "Two Heuristics for the Absolute P-Center Problem in Graphs," *Math. Slovaca*, vol. 38, no. 3, pp. 227-233, 1988.
[13] Sylvester J-J., "A Question in the Geometry of Situation," *Quarterly Journal of Mathematics*, pp. 1-79, 1857.

Rattan Rana is a research scholar (Ph.D) in Computer Science and Engineering Department of Thapar University Patiala (India). He is awarded with the Degree of Master of Computer Applications by the IGNOU India in 2003 and Master of Philosophy in Computer Science by the Chaudhary Devi Lal University Sirsa (India) in 2008**.** He has more than 10 research papers published with international Journals. He is having more than 9 years of experience in teaching and IT industry. His research area is efficient and optimization algorithms.

Deepak Garg is currently a faculty in Computer Science and Engineering Department with Thapar University Patiala (India) and having 15 years rich cross-functional experience in continuously delivering in the capacity of teacher and researcher. Undertaken several prestigious research and consultancy assignments. Hands on experience in guiding B.Tech, M.Tech. and PhD students and producing excellent results. Esteemed member of several professional organizations, editorial board of various journals and 88 publications to the credit. Conducted many seminars/ conferences/ workshops. Chair, IEEE Computer Society, India Council.