International Journal of Engineering &Technology, 7 (3.12) (2018) 915 -918

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Examining Heuristics for the K-Centers Problem

Vikas S. Shetty[1*], Arka Mukherjee[2], K. Senthilkumar[3]

SRMIST, Chennai, India
**Corresponding Author E-mail: [1]shettyvikas209@gmail.com, [2]arka161@gmail.com, [3]senthilkumar.k@ktr.srmuniv.ac.in*

Abstract

In this paper, we discuss and analyze the heuristics existing to solve the K-Center problem. The K-Center problem is used in various practical scenarios such as facility location, load-balancing, ATM mapping, Cloud Server Selection, or even data clustering and image classification. Specifically, we examine a standard Greedy algorithm with an approximation factor of 2, the clustering algorithm introduced by Gonzales in 1985, and the Dominating Set Algorithm(commonly referred to as the elimination heuristic) devised by Jurij Mihelič and Borut Robič. We also propose a new heuristic to solve the specified problem using Tree-Independent Dual-Trees devised by Ryan R. Curtin.

Keywords: *K-Centers, NP-Hard, Graph Theory, Facility Selection, Fast Machine Learning*

1. Introduction

The K-Center problem is extremely important in various practical world scenarios [1,2]. Imagine an E-Commerce business trying to decide in which "k" cities (here, we call them centers) it would place factories in to serve all "n" locations on the map efficiently. An issue thusly can be effectively assessed utilizing estimation algorithms for the K-Centers problem. The K-Centers Algorithm can be applied whenever we are choosing the "best" vertices in a graph of points. Let us now define our problem more formally. Consider a complete, undirected, and acyclic Graph G = (V, E) where V corresponds to the Vertices and E corresponds to the Edges. Also, the edge-weights in this graph satisfy the triangle inequality, that is, the sum of any two edge weights in the graph is always greater than a third edge weight (when the triangle inequality doesn't exist, we do not have a constant approximation ratio for our problem unless P=NP). Consider a positive integer "k" where K<=|V|. Additionally, think about a set X. Our set X is a subset of the nodes in the graph and |X|<=k. We additionally characterize the distance function dist (v, X) where v∈V and X are as defined previously. The distance function gives us the smallest distance from any vertex v to our above set X. Our goal is to find the set X⊆V, such that the maximum value of our function dist (v, X) is minimized. Likewise, our K-Centers problem is NP-Hard, which implies the best way to get a polynomial time solution is utilizing approximation algorithms [5]. There are already several existing techniques to solve the K-Centers problem, and we will just discuss some of the methods. One of the easiest ways to solve the problem would be the Greedy Algorithm with an Approximation Factor of 2 [9]. Also, no p-approximation heuristic exists with p<2 [8]. In this technique, we keep randomly picking centers and adding all the other cities within a specific radius to a given set till that set has all the cities in our graph. We repeat the process till we find "k" vertices out of "n" nodes. A more efficient way to solve the problem would be to apply the Greedy Algorithm formulated by Gonzales [3,4]. We select random centers, then we select the points which are the furthest away from the selected centers to be the new centers. Another new interesting heuristic is the dominating set technique [7]. In this heuristic, we solve a series of dominating set algorithms and apply parametric pruning to obtain our k centers. There are several other approaches to solve the K-Centers problem - Using Integer Linear Programming, the Minimum Cover Technique, Tabu Search,etc [10,1,12,13,11]. However, in this paper, we won't discuss go into a lot of detail to explain those algorithms.

In the next section of this paper, we will explain in detail the concepts involved in the 2 Approximate Greedy Algorithm, the 1985 Gonzales Algorithm, and the Dominating Set Algorithm. In the section after that, we will discuss a new heuristic using Tree-Independent Dual-Trees which were originally formulated by Ryan R. Curtin et. al [14].

2. Examining Existing Heuristics

In this section of the paper, we shall examine the existing approximation heuristics to solve the K-Centers problem.

A. The 2-Approxiomate Greedy Algorithm:

This is one the simplest techniques existing to solve the problem [9]. We start by making a set W, W is a subset of V and the |W| < k. We create another set called C', where the set is initially the collection of nodes in our given graph. We also consider a random positive integer r which has a value on the same scale as our edge weights. Our notations are formally defined in Figure 1 and our Algorithm is stated in Figure 2.

Notations:
k: The number of centers to be chosen.
C': The set which initially contains all the nodes/vertices in the graph.
W: The final set which will contain K Centers, initially equals 0.
V: The set of all the vertices

Fig. 1: Notations involved in the 2 Approximate Greedy Algorithm

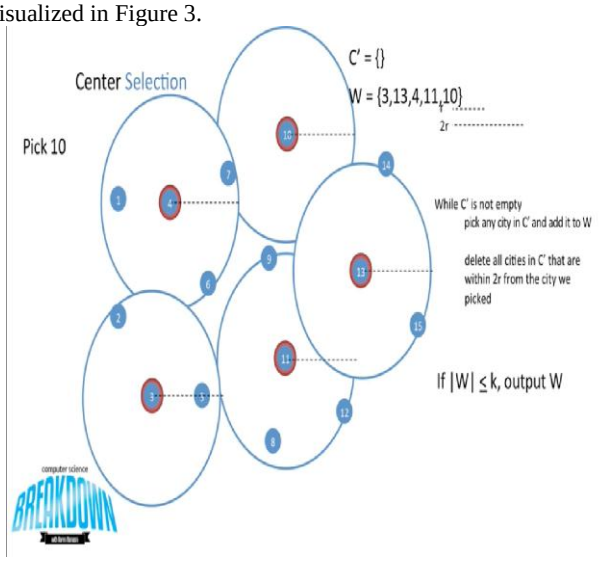
We keep selecting vertices within a (2*r) radius and popping them from the C' set till the set is empty. When it is empty, we check if we have the correct number of centers. If we do not have the needed nodes in W, we repeat the entire algorithm with a different value of 'r' till we get the right number of centers.

Input: Graph G=(V,E)
Output: A set containing K Centers as described above
Algorithm:

1. While C' is not Empty:
2. Pick any city in C' and add it to the set W
3. Delete all cities from C' within the (2*r) radius
4. if (|W| <= k)
5. Output W as the selected K-Centers chosen, return
6. else
7. No set containing K-Centers exist within chosen radius
8. Repeat the algorithm with new value of r

Fig. 2: The 2-Approximate Greedy Heuristic

This Algorithm might not give the exact result in each run, but it comes very close for several cases. A sample center selection is visualized in Figure 3.



(Image Courtesy: CSBreakdown)

Fig. 3: Sample Center Selection

B. The Gonzalez Algorithm (1985):

This is similar to the previous approximation heuristic discussed, for the most part. However, we do not choose a random value of "r". In this algorithm, we choose the furthest point from our initial random selected center as the next center point [3,4].
This Algorithm has an approximation factor of roughly 1.8 and it guaranteed to create an outcome where the final clustering cost is at most twice-optimal. Let us go over some notation in figure 4, write down the algorithm in figure 5.This algorithm is also called the farthest point algorithm because we select the furthest point to be the new center every time.

Notations:
d_i = Distance from p_i to the center nearest to it
r_j = the maximum value of di after j centers are selected
r_k=The final clustering cost

Fig. 4: Notations involved in the Gonzalez Algorithm

Input: A set of nodes/vertices, called P, a number: 'k'
Output: A set W, containing all K-Centers
Algorithm:
1. for i ← 1 to n
2. $d_i \leftarrow \infty$
3. $c_1 \leftarrow p_1$
4. for j ← 1 to k
5. $r_j \leftarrow 0$
6. for i ← 1 to n
7. $d_i \leftarrow \min\{d_i, |p_i c_j|\}$
8. if $r_j < d_i$
9. $r_j \leftarrow d$; $c_{j+1} \leftarrow p_i$
10. return {c1,c2,c3…..ck}

Fig. 5: Algorithm for the farthest point algorithm

In this algorithm, we choose all the 'K' points one by one starting with a random point as our first center. After the first center is chosen, we choose the furthest point away from the center as the next center and repeat our algorithm till all the vertices in our graph are enclosed inside a hypothetical circle.
This above algorithm has a total running time of O(kn). However, we can improve this algorithm to an asymptotic upper bound of O(nlogk) using a slightly more complicated algorithm devised by Tomas Feder and Daniel Greene[15]. However, their faster algorithm does not work in arbitrary metric spaces and the algorithm also assumes that the asymptotic time complexity for computing the distance between any two points is O(1). But, computing the distance can sometimes take quite some time if we are dealing with a large number of points with big distances amongst them.

C. The Dominating Set Heuristic:

In this heuristic [6,7], we solve a series of dominating set and apply a parametric pruning [17,18,19] technique to calculate our centers. This technique is also known as the elimination heuristic.
The Dominating Set Problem was found out as one of the sub-problems for the K-center problem. A Dominating Set [16] is a subset of our vertices such that every vertex is adjacent to at least one of the vertices in our dominating set. The goal of the dominating set problem is to find a dominating set with the lowest cardinality. The problem is also an NP-Hard problem.
Therefore, the parametric pruning technique makes sense to achieve a better heuristic.
We apply parametric pruning to compute a bottlenecked graph, then we find out the dominating set for our pruned graph. If the cardinality of the obtained set is lesser than or equal to K, we return our dominating set. Or else, we use another value of an edge cost and perform parametric pruning again. The algorithm for this heuristic is given in Figure 6.

Input: A complete Graph G=(V,E)
Output: Set of centers C
Algorithm:
1. Sort edge-weights in ascending order, z = |E|
2. for $c_i = c_1$ to c_z
3. G_i = ParametricBottleneck(G,C_i);
4. C = DominatingSetAlgorithm(G_i)
5. if |C| <= k , return C;
6. Terminate algorithm

Fig. 6: Algorithm for the Dominating-Set Heuristic

The time complexity of this heuristic is $O(n^2 \log n)$ if backtracking

is effectively utilized in the dominating-set subroutine. This heuristic also has lesser deviation when compared to the other approximation algorithms we discussed.

3. The Tree Independent Dual-Tree Heuristic

Tree-Independent Dual-Trees are Dual-Tree algorithms devised by Ryan R. Curtin et al. to make Dual-Trees work with any kind of tree as a base tree [14]. The previous dual-tree algorithms were dependent on the type of tree used for construction (eg: K-D Trees, Octrees, yinyang trees, etc).
In this heuristic, we use randomization, clustering, and the dual-tree traversal together to efficiently solve the K-Centers problem. Dual-Trees can work well here because clustering is easier in a given partitioned space and scoring is faster.

A Dual-Tree algorithm is represented in four parts: A space partitioning tree – a tree which divides our plane into, a dual-tree traversal, a BaseCase() subroutine, and a Score() subroutine. We briefly explain the algorithm in Figure 7.

Input: A set of nodes/vertices
Output: A number of K-Centers
Algorithm:

1. Select a random vertex, and determine a radius.
2. Select a center if it's <= than a cluster parameter.
3. Apply a Dual-Tree Traversal:
4. During the traversal, add unmarked node to a cluster if the distance between the reference point and query point is <= the average computed radius.
5. If the distance is >= the average computed radius, the node is pruned and the children nodes are not visited.
6. After the traversal is complete, add remaining unmarked node to the appropriate cluster based on distance from the center.
7. Print nodes with their marked cluster center.

Fig. 7: The Dual-Tree Heuristic for this problem

The radius is determined by using the BaseCase() function of the K-Nearest Neighbours algorithm in Curtin's paper(Figure 8).

Input: Query coordinate t_q, reference coordinate t_r, vector of k-nearest candidate points N_{pq} and k candidate distances D_{pq} (arranged via ascending distance)
Output: Distance d between t_q and t_r
Algorithm:
1. $d \leftarrow \| t_q - t_r \|$
2. If $d < D_{pq}[k]$ and BaseCase(t_q, t_r) is not yet called
3. Then
4. Insert d into sorted vector D_{pq} and shorten list to Length k
5. Insert t_r into N_{pq} such that N_{pq} is ordered by
6. Distance list; shorten list to length k
7. Return d

Fig. 8: The BaseCase() method for K-Nearest Neighbours

4. Standards of Existing Heuristics

Experimentally, it has been shown that the Gonzalez Algorithm is the fastest [7]. However, those algorithms also have a pretty high approximation factor. Note that the quality of the algorithm is inversely proportional to the Approximation Factor. The Approximation Factor can be defined as a ratio between the approximation solution and the absolute solution [20]. In Figure 9, we look at the graph of the approximation factors of various known Algorithms. The Gonzales Algorithm we discussed is denoted by GnR and the Dominating Set Heuristic we studied is know as Scr(Scoring Heuristic) here[7]. Even though Scr has a larger running time than Gonzalez, we have a pretty good approximation factor which is a good algorithm for practical applications.

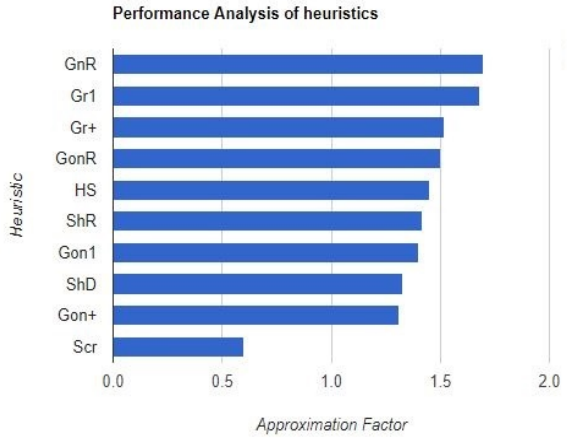


Fig. 9: Approximation factor of current heuristics

5. Conclusion

In this paper, we have gone over some of the existing algorithms for solving the NP-Hard K-Centers problem. We have also presented the algorithm for a new heuristic using the Tree-Independent Dual-Tree Algorithms. Finally, we discuss the approximation factor of the standard existing techniques and successfully cite a graph to analyze the factors.

Acknowledgement

This paper was supported by the Department of Computer Science and Engineering. School of Computing, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India.

References

[1] M.S. DASKIN, "Network and Discrete Location: Models Algorithms and Applications", Wiley, New York, 1995.
[2] CHARLES S. REVELLE AND H.A. EISELT, "Location analysis: A synthesis and survey", European Journal of Operational Research, 165:1–19, 2005.
[3] JEFF ERICKSON, "K-Approximation Algorithms, Lecture Notes, Data Structures and Algorithms", University of Illinois at Urbana–Champaign, 2009
[4] T. GONZALEZ, "Clustering to minimize the maximum intercluster distance", Theoretical Computer Science., 38:293–306, 1985.
[5] M.R. GAREY AND D.S. JOHNSON, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Co., San Francisco, 1979.
[6] J. MIHELIČ AND B. ROBIČ, "Approximation algorithms for the k-center problem: an experimental evaluation", Proc. OR 2002, Klagenfurt, Austria, 2002.
[7] J. MIHELI Č AND B. ROBIČ, "Solving the k-center Problem Efficiently with a Dominating Set Algorithm", Journal of Computing and Information Technology - CIT 13, 3, 225–233, 2005.
[8] D.S. HOCHBAUM, ed., "Approximation Algorithms for NP-hard Problems", PWS publishing company, Boston, 1995.
[9] "Introduction to Approximation Algorithms - K Center Problem", CSBreakdown(https://www.youtube.com/watch?v=dpYZojRuJEI)
[10] E. MINIEKA, "The m-center problem", SIAM Rev., 12:138–139, 1970.
[11] N.MLADENOVI´C, M. LABBE, AND P. HANSEN, "Solving the p-center problem with tabu search and variable neighborhood search", Networks 42(1):48-64, 2003.
[12] S. ELLOUMI, M. LABBE, AND Y. POCHET, "New formulation and resolution method for the p-center problem", http://www.optimization-online.org/DB_HTM, 2001.

[13] T. ILHAN AND M.C. PINAR, “An efficient exact algorithm for the vertex p-center problem”,http://www.optimization-online.org/DB_HTML/2001/10/376.html, 2001.
[14] RYAN R. CURTIN, WILLIAM B. MARCH, PARIKSHIT RAM, DAVID V. ANDERSON, ALEXANDER G. GRAY, CHARLES L. ISBELL, JR, “Tree Independent Dual-Trees”, Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.
[15] Tomas Feder and Daniel H. Greene, “Optimal algorithms for approximate clustering”, Proc. 20th STOC, 1988.
[16] Alikhani, S. and Peng, Y.-H, "Introduction to Domination Polynomial of a Graph”, Ars Combin. 114, 257-266, 2014.
[17] D.S. HOCHBAUM AND D.B. SHMOYS, "A best possible heuristic for the k-center problem" , Mathematics of Operations Research, 10:180–184, 1985.
[18] J. PLESNIK, “A Heuristic for the p-Center Problem in Graphs”, Discrete Applied Mathematics 17:263–268, 1987.
[19] V. VAZIRANI, “Aproximation Algorithms”, Springer, 2001
[20] Guha, S. & Khuller, S. Algorithmica (1998) 20: 374. https://doi.org/10.1007/PL00009201, 1998.