Approximation Algorithms for the k-center Problem: An Experimental Evaluation

Article · January 2003

DOI: 10.1007/978-3-642-55537-4_60

2 authors:

Jurij Mihelič
University of Ljubljana

Borut Robic
University of Ljubljana

Approximation algorithms for the k-center problem: an experimental evaluation

Jurij Mihelič[1] and Borut Robič[1]

Faculty of Computer and Information Science
University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia
{jurij.mihelic, borut.robic}@fri.uni-lj.si

Abstract. In this paper we deal with the vertex k-center problem, a problem which is a part of the discrete location theory. Informally, given a set of cities, with intercity distances specified, one has to pick k cities and build warehouses in them so as to minimize the maximum distance of any city from its closest warehouse. We examine several approximation algorithms that achieve approximation factor of 2 as well as other heuristic algorithms. In particular, we focus on the clustering algorithm by Gonzalez, the parametric pruning algorithm by Hochbaum-Shmoys, and Shmoys' algorithm. We discuss several variants of the pure greedy approach. We also describe a new heuristic algorithm for solving the dominating set problem to which the k-center problem is often reduced. We have implemented all the algorithms, experimentally evaluated their quality on 40 standard test graphs in the OR-Lib library, and compared their results with the results found in the recent literature.

1 Introduction

Problems of finding the best location of facilities in networks or graphs abound in practical situations. One of the well known facility location problems is the *vertex k-center* problem, where given n cities and distances between all pairs of cities, the aim is to choose k cities (called centers) so that the largest distance of a city to its nearest center is minimal. More formally, the vertex k-center can be defined as follows. Let $G = (V, E)$ be a complete undirected graph with edge costs satisfying the triangle inequality, and k be a positive integer not greater than $|V|$. For any set $S \subseteq V$, and vertex $v \in V$, we define $d(v, S)$ to be the length of a shortest edge from v to any vertex in S. The problem is to find such a set $S \subseteq V$, where $|S| \leq k$, which minimizes $\max_{v \in V} d(v, S)$. The vertex k-center problem is NP-hard [5].

A popular way to solve the k-center problem consists of solving a series of *set cover* problems [2,4,9,10]. At each step, a threshold for the cover distance is chosen and it is checked whether all vertices can be covered within this distance using at most k centers; if so, the threshold is decreased, otherwise it is increased. (One can also use the *dominating set* problem instead of the set cover problem [7].) For example, Minieka [10] solved the k-center problem as a series of set cover problems. More elaborate versions of this approach were described by Daskin [2,3], where also the *maximum cover* problem was used, and Ellumni et al. [4] and Ilhan et al. [9], which applied more efficient definition of the problem. Usually, these set cover problems were solved with integer programming. Another way to solve the k-center problem was recently given by Mladenović et al. [11], where the *tabu search*, *variable neighborhood search* and various greedy methods were used. The greedy method was also applied by Gonzalez [6], Hochbaum and Shmoys [8], and Shmoys [12]. The last three describe 2-approximation algorithms which are the *best possible* in the sense that no r-approximation algorithm exists with $r < 2$, unless $P = NP$ [7]. (No approximation algorithm exists in case the triangular inequality does not hold, unless P=NP.)

In the following we briefly describe various heuristics for the k-center problem that are not based on the integer programming (Section 2). We then describe a new heuristic, which combines the greedy approach with solving the dominating set problem, and returns surprisingly good results (Section 3). We have experimentally evaluated all these heuristics as well as the new one. The experimental results are given in Section 4.

2 Heuristics

By using greedy heuristics we often locate centers one by one until there are k centers. For the selection of the first center there may be several possibilities. For example, the center can be located at random, it can be the result of the 1-center problem, or we can apply a heuristic n-times, $n = |V|$, each time with different starting vertex, and then choose the best of the solutions. These approaches will be called *random*, *1-center*, and *plus* version, respectively.

A very simple heuristic is the *pure greedy method*, where centers are located one by one so that the objective function is each time reduced as much as possible. For the selection of the first center we have implemented random, 1-center and plus version. It is easy to see that the time complexity of this pure greedy method is $O(kn^2)$.

Another greedy heuristic for the k-center problem was described by Gonzalez [6], who was able to prove the approximation factor of 2. The algorithm builds final solution in k steps so that, given a partial solution C_{i-1}, it forms a new partial solution C_i by extending C_{i-1} with the vertex v which is the farthest from the C_{i-1}, i.e. the vertex v which maximizes $d(v, C_{i-1})$ at step i. We have implemented random, 1-center, and plus version of this algorithm. The time complexity of Gonzalez's algorithm is $O(kn)$.

Shmoys [12] briefly describes 2-approximation algorithm for the decision version of k-center problem, i.e. where radius r is also given and the aim is to decide if there exist k vertices so that the coverage distance from these vertices is at most r. The algorithm repeatedly chooses one of the remaining vertices v, adds it to the partial solution, and deletes all vertices whose distance to v is at most $2r$. At the end, if the size of the solution exceeds k, the algorithm outputs "no", otherwise "yes". We implemented two versions where either random vertex or vertex with maximum degree can be chosen on each step. The algorithm for the optimization version of problem runs the algorithm for the decision version several times with increasing value of r. Time complexity of this algorithm is $O(kn^3)$.

Hochbaum and Shmoys [8] introduced the algorithmic technique called *parametric pruning* for solving k-center problem. Initially, edge costs are sorted in nondecreasing order. For each edge cost t the graph is pruned by removing edges with cost greater than t. The aim is to find a *minimum dominating set* in the pruned graph, i.e. the smallest set S of vertices such that every vertex not in S is adjacent to one of the vertices in S. If the cardinality of the minimum dominating set of the pruned graph is at most k, then such a dominating set is also the optimal solution for k-center problem.

Unfortunately, to compute the minimum dominating set is NP-hard optimization problem [5]. Consequently, instead of searching for the minimum dominating set we rather search for the *maximal independent set*[1], i.e the subset S of V such that no two vertices of S are connected in G and no vertex can be added to S while S retaining this property.

Define the *square* of the graph G to be the graph G^2 containing an edge (u, v) whenever G has a path of at most two edges between u and v, $u \neq v$. It is well known that every maximal independent set is also dominating set. The fact that the cardinality of the maximal independent set of G^2 is at most the cardinality of the minimum dominating set of G can be used to construct a 2-approximation algorithm for solving the k-center problem. More specifically, instead of searching for the minimum dominating set of G the algorithm constructs the maximal independent set of G^2. The overall time complexity of the algorithm is estimated to be $O(kn^5)$.

3 Elimination heurustic

We designed a new algorithm for the k-center problem. The algorithm is based on the standard approach that solves a series of dominating set problems. First, edge costs are sorted in a nondecreasing list which is used for getting the threshold values r and for solving the series of dominating set problems. When the cardinality of the dominating set S becomes at most k, the set S is returned as the result of the k-center problem and the algorithm is completed.

For solving the dominating set problem we developed a new heuristic algorithm. Informally, a pair of numbers $(c(v), s(v))$ is initially assigned to each vertex $v \in V$, where $c(v)$ (*cover count*) is the number of vertices that can cover v within distance r, while $s(v)$ (*vertex score*) is used in the following selection process ($s(v)$ is initially set to $c(v)$). At each step of the selection process the vertex v with the smallest $s(v)$ is chosen. If there is a vertex $u \in V$ such that $d(u, v) \leq r \wedge c(u) = 1$, then v is added to the set S of centers;

[1] *Maximum independent set* is NP-hard problem, while *maximal* independent set is one of the suboptimal solutions.

otherwise, $s(u)$ is incremented for all $u \in V$ for which $d(u, v) \leq r$. Next, the cover count $c(u)$ is decremented for all vertices $u \in V$ with $d(u, v) \leq r$. These steps are repeated until all the vertices of the graph have been processed.

Notice, that we use $c(v)$ to ensure that every vertex is covered at least once. In addition, the way $c(v)$ is used makes it possible to easily adapt the algorithm for solving the (*fault-tolerant*) α-neighbor k-center problem where every node must be covered by at least α centers. Moreover, one can adapt the algorithm to solve the minimum set cover problem instead of the minimum dominating set problem. This is useful when solving the so-called k-supplier problem, where k centers must be chosen from a predefined set of vertices.

4 Experimental Results

We tested the described algorithms on 40 OR-Lib test problems, which were originally designed for testing p-median problems [1]. The number of vertices ranges from 100 to 900 while k ranges from 5 to 90. The preprocessing phase runs the all shortest paths algorithm (time complexity $O(n^3)$).

All the algorithms were implemented in Borland Delphi 6.0, and were tested on a computer with Intel processor running at 1.7 GHz with 512MB of system memory. Designations and names of algorithms appear in Table 1. Although our primary aim was to compare the quality of the solutions, let us mention that Gonzalez algorithms were the fastest (running below 1 second). The average time of `HS` was about 100 seconds. The pure greedy methods were quite fast (about 2 seconds on average), but their execution time was very variable and dependent on the parameter k, Shmoys' variants were also fast as well as our `Scr`. (Notice that plus variants run much slower due to the algorithm which tries all vertices for the first center.)

Recall that approximation factor is the ratio between approximated and the optimal objective value. Since sometimes we do not know the optimal solution, we take the best known so far. We call such a ratio an *approximation degree*. Nevertheless, in our case most of the best known objective values were proved to be optimal (see for example [4,9]). Approximation degrees for each algorithm are given in Table 1 below, where we also included the results for Daskin's and tabu search approach.

Objective values for all of the 40 problems are in Table 2. The pure greedy method is the worst, while only slightly better results were with the plus variant. The solution quality strongly depends on the parameter k, and is much better for low values of k. Gonzalez algorithms are very fast and solutions are about 50% worse than best known. The algorithms `HS, ShR, ShD` exploit very similar problem properties, and consequently return very similar results. Our algorithm `Scr` proved to be quite competitive since it achieved better results than any of the implemented algorithms, with the exception of the integer programming approaches [3,4,9] and tabu search and variable neighborhood search [11].

Algorithm	Level	Deviation	Description
GrR	1,697	0,559	Pure greedy first random
Gr1	1,675	0,570	Pure greedy first 1-center
Gr+	1,512	0,550	Pure greedy plus
GonR	1,495	0,130	Gonzalez first random
Gon1	1,398	0,128	Gonzalez first 1-center
Gon+	1,317	0,139	Gonzalez plus
ShR	1,432	0,112	Shmoys random
ShD	1,343	0,105	Shmoys degree
HS	1,462	0,177	Hochbaum-Shmoys
Scr	1,058	0,043	Elimination heuristic
Das	1,002	0,007	Daskin
TS	1,025	0,045	Tabu search

Table 1. Approximation degrees

References

1. J. E. Beasley. A note on solving large p-median problems. *European J. Oper. Res.*, 21:270–273, 1985.
2. Mark S. Daskin. *Network and Discrete Location: Models Algorithms and Applications*. Wiley, New York, 1995.
3. Mark S. Daskin. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. *Communications of the Operations Research Society of Japan*, 45:9:428–436, 2000.
4. Sourour Elloumi, Martine Labbe, and Yves Pochet. New formulation and resolution method for the p-center problem. 2001.
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. T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science.*, 38:293–306, 1985.
7. Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS publishing company, Boston, 1995.
8. Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.
9. Taylan Ilhan and Mustafa Pinar. An efficient exact algorithm for the vertex p-center problem. 2001.
10. E. Minieka. The m-center problem. *SIAM Rev.*, 12:138–139, 1970.
11. N. Mladenović, M. Labbe, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. 2000. http://smg.ulb.ac.be/Preprints/Labbe00_20.html.
12. David B. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. Technical report, Ithaca, NY 14853, 1995. http://citeseer.nj.nec.com/shmoys95computing.html.

#	n	k	Best	Das	TS	GrR	Gr1	GrP	Gon	Gon1	Gon+	HS	ShR	ShD	Scr
1	100	5	127	127	127	143	133	133	186	162	155	184	188	171	133
2	100	10	98	98	98	117	117	110	131	124	117	160	128	135	109
3	100	10	93	93	93	126	116	106	154	133	124	160	140	120	99
4	100	20	74	74	74	127	127	92	114	99	92	124	109	84	83
5	100	33	48	48	48	87	87	78	71	64	62	77	62	59	48
6	200	5	84	84	84	98	94	89	138	99	98	126	138	106	90
7	200	10	64	64	64	78	79	77	96	87	85	90	88	90	70
8	200	20	55	55	55	72	72	72	82	72	71	84	74	68	60
9	200	40	37	37	37	73	73	63	57	51	49	62	50	52	38
10	200	67	20	20	20	44	44	38	31	29	29	32	28	28	20
11	300	5	59	59	59	68	67	61	73	68	68	82	73	74	60
12	300	10	51	51	51	62	72	56	71	70	66	78	74	70	53
13	300	30	35	36	36	64	64	52	59	51	49	60	54	52	38
14	300	60	26	26	26	60	60	46	40	36	36	44	36	34	27
15	300	100	18	18	18	42	42	40	25	25	23	30	22	20	18
16	400	5	47	47	47	52	51	47	84	55	52	64	83	58	48
17	400	10	39	39	39	50	50	43	56	51	48	56	56	52	41
18	400	40	28	28	28	50	50	42	44	41	39	46	40	38	31
19	400	80	18	18	19	40	40	31	28	28	27	30	26	24	20
20	400	133	13	13	14	32	32	32	19	19	17	22	18	16	14
21	500	5	40	40	40	48	48	42	53	51	45	52	53	45	40
22	500	10	38	38	38	48	49	43	56	54	47	54	54	48	41
23	500	50	22	22	23	41	41	35	34	33	32	36	32	30	24
24	500	100	15	15	16	35	35	32	23	23	21	24	22	20	17
25	500	167	11	11	12	27	27	27	15	15	15	18	16	14	11
26	600	5	38	38	38	44	43	39	50	47	43	52	50	52	41
27	600	10	32	32	32	37	39	35	43	42	55	42	44	44	33
28	600	60	18	18	19	33	33	27	28	28	25	28	28	28	20
29	600	120	13	13	13	34	36	34	19	19	18	22	18	18	13
30	600	200	9	9	11	29	29	29	14	14	13	16	12	12	10
31	700	5	30	30	30	35	34	31	42	38	36	40	42	44	30
32	700	10	29	29	29	35	35	32	45	43	37	40	44	40	31
33	700	70	15	15	16	32	26	24	26	25	23	26	24	22	17
34	700	140	11	11	12	30	30	27	17	17	16	18	16	16	11
35	800	5	30	30	30	37	32	31	38	37	34	40	38	38	32
36	800	10	27	27	27	34	34	30	41	41	34	38	42	38	28
37	800	80	15	15	16	26	26	26	25	24	23	24	22	22	16
38	900	5	29	29	29	42	35	31	36	38	31	38	40	38	29
39	900	10	23	23	24	27	28	25	35	35	28	32	36	34	24
40	900	90	13	13	14	25	22	22	21	20	19	22	20	20	14

Table 2. Objective function values