ELSEVIER

Theoretical Computer Science 242 (2000) 237–245

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fault tolerant K-center problems ☆

Samir Khuller [a,*,1], Robert Pless [b,2], Yoram J. Sussmann [c,3]

[a] *Department of Computer Science and UMIACS, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

[b] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*

[c] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*

Received September 1996; revised May 1998

Communicated by D. Peleg

Abstract

The basic K-center problem is a fundamental facility location problem, where we are asked to locate K facilities in a graph, and to assign vertices to facilities, so as to minimize the maximum distance from a vertex to the facility to which it is assigned. This problem is known to be NP-hard, and several optimal approximation algorithms that achieve an approximation factor of 2 have been developed for it.

We focus our attention on a generalization of this problem, where each vertex is required to have a set of α ($\alpha \leqslant K$) centers close to it. In particular, we study two different versions of this problem. In the first version, each vertex is required to have at least α centers close to it. In the second version, each vertex that *does not have a center placed on it* is required to have at least α centers close to it. For both these versions we are able to provide polynomial time approximation algorithms that achieve constant approximation factors for *any* α. For the first version we give an algorithm that achieves an approximation factor of 3 for any α, and achieves an approximation factor of 2 for $\alpha < 4$. For the second version, we provide algorithms with approximation factors of 2 for any α. The best possible approximation factor for even the basic K-center problem is 2, assuming $P \neq NP$. In addition, we give a polynomial time approximation algorithm for a generalization of the K-supplier problem where a subset of at most K supplier nodes must be selected as centers so that every demand node has at least α centers close to it. For this version our approximation factor is 3. The best possible approximation factor for even the basic K-supplier problem is 3, assuming $P \neq NP$.

Keywords: Approximation algorithms; Facility location; K-center; Fault-tolerance

☆ An earlier version of this paper appeared in the Proceedings of the Third Italian Conference on Algorithms and Complexity (CIAC), 1997.

* Corresponding author.

E-mail address: samir@cs.umd.edu (S. Khuller).

[1] Research supported by NSF Research Initiation Award CCR-9307462, and NSF CAREER Award CCR-9501355.

[2] Research supported by NSF Grant IRI-90-57934.

[3] Research supported by NSF CAREER Award CCR-9501355.

0304-3975/00/$ - see front matter
PII: S0304-3975(98)00222-9

1. Introduction

The basic K-center problem is a fundamental facility location problem and is defined as follows: Given an edge-weighted graph $G = (V, E)$, find a subset $\mathscr{S} \subseteq V$ of size at most K such that each vertex in V is "close" to some vertex in $\mathscr{S}$. More formally, it is defined as follows:

$$\min_{\mathscr{S} \subseteq V \ |\mathscr{S}| \leqslant K} \max_{u \in V} \min_{v \in \mathscr{S}} d(u, v),$$

where d is the distance function. For example, one may wish to install K fire stations and minimize the maximum distance (response time) from a location to its closest fire station. The problem is known to be NP-hard [5].

An approximation algorithm with a factor of ρ, for a minimization problem, is a polynomial time algorithm that guarantees a solution with cost at most ρ times the optimal solution. Approximation algorithms for the basic K-center problem have been very well studied and are known to be optimal [6–9]. These schemes present natural methods for obtaining an approximation factor of 2. Several approximation algorithms are known for interesting generalizations of the basic K-center problem as well [1, 3, 8, 11, 13, 15], including costs [8, 13, 15], weights [3, 13, 15], and capacities [1, 11]. A related problem of placing as few centers as possible so that each vertex without a center has at least k vertex-disjoint paths to centers is also studied in [1].

The α-neighbor K-center problem is discussed in a recent paper by Krumke [12]. The problem is formally defined as follows: given an edge-weighted graph $G = (V, E)$ find a subset $\mathscr{S} \subseteq V$ of size at most K such that each vertex in $V - \mathscr{S}$ is "close" to a set of α vertices in $\mathscr{S}$. Formally,

$$\min_{\mathscr{S} \subseteq V \ |\mathscr{S}| \leqslant K} \max_{u \in V - \mathscr{S}} \delta^{(\alpha)}(u, \mathscr{S}),$$

where

$$\delta^{(\alpha)}(u, \mathscr{S}) = \min_{A \subseteq \mathscr{S}, |A| = \alpha} \max_{a \in A} d(u, a),$$

where d is the distance function. Krumke [12] gives an algorithm with an approximation factor of 4, by generalizing the notion of an independent set of vertices.

The main motivation to study this problem is to provide some notion of fault-tolerance. Namely, if we are concerned with the placement of emergency facilities, then providing "backup" centers when one center fails to respond is useful [14].

We consider a variation of this problem as well, called the α-all-neighbor K-center problem that is formally defined as follows: given an edge-weighted graph $G = (V, E)$ find a subset $\mathscr{S} \subseteq V$ of size at most K such that each vertex in V is "close" to a set of α vertices in $\mathscr{S}$. Formally,

$$\min_{\mathscr{S} \subseteq V \ |\mathscr{S}| \leqslant K} \max_{u \in V} \delta^{(\alpha)}(u, \mathscr{S}).$$

1.1. Our results

We improve Krumke's result, and show that we can obtain an approximation factor of 2 for the problem considered in his paper. This matches the bound for the basic K-center problem, which is the best possible assuming P $\neq$ NP [9]. The algorithm is a very natural extension of the method given by Hochbaum and Shmoys [8] for the basic K-center problem.

We also show that for the α-all-neighbor K-center problem, we can obtain an approximation factor of 3 for any α, and a similar algorithm gives an approximation factor of 2 for $\alpha < 4$ (perhaps the practically interesting case).

For the α-neighbor K-suppliers problem, we obtain an approximation factor of 3. For the K-suppliers problem, Hochbaum and Shmoys [8] give a proof (originally due to Howard Karloff) showing that the factor of 3 is the best possible unless P = NP.

In [10] we provide algorithms for several generalizations of these problems where we also include the notion of weights and costs [8, 13, 15].

Recently, Chaudhuri et al [2] independently came up with a different algorithm with a matching approximation factor for the α-neighbor K-center problem. Their algorithm modifies Krumke's approach and is different from the methods used in our paper.

2. α-all-neighbor K-center problems

We may assume for simplicity that G is a complete graph, where the edge weights satisfy the triangle inequality. (We can always replace any edge by the shortest path between the corresponding pair of vertices.)

The algorithm uses the threshold method (see [4]) used for the K-center problem by Hochbaum and Shmoys in [8]. Sort all edge weights in non-decreasing order. Let the (sorted) list of edges be $e_1, e_2, \ldots e_m$ (where $m = \binom{n}{2}$). For each i, let the threshold graph G_i be the subgraph obtained from G by including edges of weight at most $d(e_i)$. Run the algorithm below for each i from 1 to m, until a solution is obtained. (One can also use binary search to speed up the computation as suggested by Hochbaum and Shmoys [8].) In each iteration, we work with the subgraph G_i and view it as an unweighted graph. Since G_i is an unweighted graph, when we refer to the distance between two nodes, we refer to the number of edges on a shortest path between them. In iteration i, we find a solution using some number of centers. If the number of centers exceeds K, we prove that there is no solution with cost at most $d(e_i)$. If the number of centers is at most K, we return an approximate solution.

Let G_i^2 denote the graph obtained by adding edges to G_i between nodes that have at least one common neighbor.

2.1. Any α

We give an algorithm that obtains an approximation factor of 3 for any value of α. The following technique was introduced by Hochbaum and Shmoys [7, 8] and has

been used extensively to solve K-center problems. Find a maximal independent set I in G_i^2. Note that if the independent set has size $|I|$, then any solution with radius $d(e_i)$ must use at least $\alpha|I|$ centers, because nodes in the independent set cannot be assigned a common center. We therefore place α centers at each node in the independent set. At this point, every node in the graph is at distance at most 2 (in G_i) from α centers.

We now have to distribute the centers so that no two centers are placed on a common node. Note that if there is a solution with radius $d(e_i)$, then every node has degree at least $\alpha - 1$ in G_i. We can therefore move $\alpha - 1$ centers from each node in the independent set to a subset of its neighbors in G_i. Since every node in the graph is at distance at most 2 from a node in the independent set, we must have that every node in the graph is at distance at most 3 from α centers, which implies that this approach gives an approximation factor of 3.

2.2. $\alpha = 2, 3$

Here we give another algorithm for the α-all-neighbor K-center problem. The algorithm essentially chooses α independent sets in G_i^2. The algorithm gives an approximation factor of 3 for any α, and we prove that it achieves an approximation ratio of 2 if α is 2 or 3.

The algorithm consists of α iterations. In each iteration we choose an independent set in G_i^2. At the end of each iteration $j = 1, 2, \ldots, \alpha$, we guarantee that each node is covered by at least j centers within two steps. We ensure this by defining a "covering number" $C(u)$ (initially 0) for each node u. When we add a node v to the independent set in iteration j, we increase the covering number of v and all nodes u adjacent to v in G_i^2 such that $C(u) < j$. At any time during the execution of the algorithm, let the set $\mathcal{S}$ refer to the set of nodes picked in the independent set of any previous iteration.

The independent set in each iteration j is composed of nodes v such that just before v was picked, we had $C(v) < j$. Each iteration consists of two phases. In the first phase, we choose nodes not in $\mathcal{S}$ to add to the independent set. In the second phase, if there are still nodes v remaining with $C(v) < j$, we allow nodes already in $\mathcal{S}$ to be picked as well. We maintain a count of the number of iterations in which a node v was selected in the independent set in phase two as $extra(v)$. $Extra(v)$ is the number of centers assigned to v throughout the algorithm that cannot be placed at node v. These centers will be shifted at the end of the algorithm to neighboring locations not in $\mathcal{S}$, to make centers distinct. Define $helps(v)$ as the set of neighbors u of v in G_i^2 such that v was chosen in phase two of iteration j and $C(u) < j$ when v was picked in this iteration. These are the nodes we must consider when shifting extra centers from v. We prove that if $\alpha < 4$ then we can shift the extra centers to nodes within distance 2 in G_i for all nodes in $helps(v)$. (This last step ensures that we place centers on distinct vertices.)

α-ALL-NEIGHBOR K-CENTER ALGORITHM(G_i).

```
1  for all v
2      C(v) = 0.
3      extra(v) = 0.
4      helps(v) = ∅.
5  for j = 1 to α do
       // Phase I
6      while ∃v ∉ S with C(v) < j do
7          create new center at v and set S = S ∪ {v}.
8          C(v) = C(v) + 1.
9          C(u) = C(u) + 1 if (u, v) ∈ E(G_i^2).
       // Phase II
10     while ∃v with C(v) < j do
11         create new center at v and set S = S ∪ {v}.
12         C(v) = C(v) + 1.
13         extra(v) = extra(v) + 1.
14         for all u with C(u) < j and (u, v) ∈ E(G_i^2)
15             C(u) = C(u) + 1.
16             Set helps(v) = helps(v) ∪ {u}.
17 for all v ∈ S with extra(v) ≥ 1 do
18     if helps(v) = ∅
19       Shift extra(v) centers to neighbors of v in G_i that are not in S.
20     else
21       Shift one center to a common neighbor of any node in helps(v) and
                                                          v in G_i not in S.
22       Shift extra(v) − 1 centers to neighbors of v in G_i that are not in S.
23 end-proc
```

Lemma 2.1. *The above algorithm uses no more centers than the optimal solution.*

Proof. In each iteration we select an independent set in G_i^2. Let I^* be the size of the largest independent set picked in any iteration. Any solution with radius $d(e_i)$ must use at least αI^* centers, and we must have that $|\mathcal{S}| \leqslant \alpha I^*$. □

Theorem 2.2. *The above algorithm returns a solution to the α-all-neighbor K-center problem with an approximation ratio of 2 if $\alpha = 2$ or 3.*

Proof. Call a node v *satisfied* in iteration j if $C(v) \geq j$. Although in each iteration we prefer to pick nodes not previously chosen as centers, after the first phase all nodes remaining with $C(v) < j$ are in $\mathcal{S}$. Define H_j to be subgraph of G_i^2 induced by these (unsatisfied) nodes in round j. Now consider the structure of H_j. The graph H_2 is a collection of singleton nodes, disconnected in G_i^2 (because they were all picked in the independent set in the first iteration). Therefore all nodes in H_2 will be added to $\mathcal{S}$.

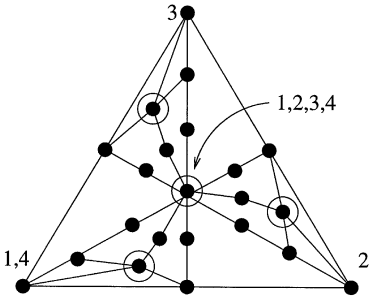

Fig. 1. The circled nodes are cliques of four nodes, and edges to circled nodes represent edges to each of the four nodes in the clique.

First, suppose $\alpha = 2$. Since the nodes in H_2 form an independent set, $helps(v) = \emptyset$ for all nodes in H_2. Therefore, we can shift all but one center to unassigned neighbors of v in G_i. Such neighbors must exist because v must have at least one neighbor in G_i and at most two centers total are placed in the neighborhood of v in G_i^2.

Now let $\alpha = 3$. Consider a node v that was assigned as a center multiple times. If $helps(v) = \emptyset$, then we can shift all but one center to unassigned neighbors of v in G_i, by the above argument.

If $helps(v) \neq \emptyset$ then we must have $|helps(v)| = 1$, because H_3 is a graph with maximum vertex degree of 1 (since any node in H_3 with degree 2 must be satisfied). Assume $helps(v) = \{u\}$. Note that only 2 centers are assigned to v. This follows from the fact that the center on u covers v. We must shift the extra center so that it covers both u and v within distance 2. If u and v are adjacent in G_i, then we can shift the extra center on v to any neighbor of v in G_i. Otherwise, there must exist a node w adjacent to both u and v in G_i. Node w does not have any centers assigned to it because it already has 3 centers adjacent to it. Therefore we can shift the extra center to w.

Any node which does not have a center placed on it has at least α centers adjacent to it in G_i^2. As shown above, a node which has a center placed on it also has at least α centers adjacent to it in G_i^2. Therefore all nodes have at least α centers within radius $2d(e_i)$. □

The following example (see Fig. 1) shows that this algorithm fails when $\alpha = 4$. Our algorithm may do the following. In the first three rounds, it chooses a center from the central clique and one of the corners – this forms a maximal independent set in G_i^2. In the fourth round, it places a center on the final remaining vertex in the central clique, and the only nodes that then remain unhappy are the corner vertices, all of whom have been picked in earlier rounds. It now picks one of the corners on which to place a second center. This center would have to be shifted off to a node which covers all three corners, and there is no vertex that is distance 2 from all corners. It is important to note that the algorithm fails not because it places too many centers. In fact, in this example the optimal solution uses 16 centers (one on every node in each of the four

cliques) while our algorithm places eight and leaves one vertex unsatisfied. While in this case we can see where to add the extra center, it is not clear how to automate this process.

3. α-neighbor K-center problems

In this section, we describe an algorithm which gives an approximation factor of 2 for the α-neighbor K-center problem.

We assume that G is a complete graph with edges satisfying the triangle inequality. Iterate for each i from 1 to m until a solution is obtained.

Consider the graph G_i^2. Every node is assigned a "covering number" $C(v)$ (initially 0). The set of centers is $\mathscr{S} = \emptyset$. At the end of each iteration $j = 1, 2, \ldots, \alpha$, we guarantee that each node not chosen as a center is covered by at least j centers within distance two. In each iteration, we pick a center that is not covered by at least j centers. We assign a center at the chosen vertex, and increase the covering number for all vertices within distance two in G_i.

α-NEIGHBOR K-CENTER ALGORITHM(G_i^2).
1 **for** all v
2 $\quad C(v) = 0$.
3 **for** $j = 1$ to α **do**
4 $\quad$ **while** $\exists v$ with $C(v) < j$ **do**
5 $\quad\quad$ create center at v and and set $\mathscr{S} = \mathscr{S} \cup \{v\}$.
6 $\quad\quad$ $C(v) = \alpha$.
7 $\quad\quad$ $C(u) = C(u) + 1$ if $(u, v) \in E(G_i^2)$.
8 **end-proc**

We find at most α independent sets in α iterations.

Theorem 3.1. *The above algorithm finds a solution to the α-neighbor K-center problem with an approximation ratio of two.*

Proof. When the algorithm terminates, each vertex has a covering number equal to α. This guarantees that each vertex was either chosen as a center, or is covered by at least α centers within distance 2. We now prove that if there is a feasible solution with K centers in some G_i, then our algorithm will not assign more than K centers in G_i.

Assume that this does not hold. In other words, there is a graph for which there is a solution that uses at most K centers, and our algorithm assigns more than K centers. Consider the smallest value of K for which the algorithm fails, and consider the smallest graph G that is a counter-example for that value of K. Assume that the centers assigned in iteration j have label j. Let S_{OPT} be the set of K vertices in graph G that have centers placed on them by the optimal solution. Note that each vertex in $V - S_{\mathrm{OPT}}$ has at least α neighbors in S_{OPT}.

If our algorithm places centers only on vertices in S_{OPT} then we certainly do not place more than K centers. Assume that j is the highest labeled center placed at $v \in V - S_{OPT}$ by the algorithm. Let $N_{OPT}(v)$ be the neighbors of v in S_{OPT}. Clearly $|N_{OPT}(v)| \geqslant \alpha$. Let $V_{OPT}(v)$ be all the vertices that are adjacent to some vertex in $N_{OPT}(v)$.

We claim that there are at most α centers placed by the algorithm in $v \cup N_{OPT}(v) \cup V_{OPT}(v)$ from G. If v had a center placed on it in iteration j, then at the instant it was placed it had at most $j-1$ centers within distance 2 in G_i. Hence, there were at most $j-1$ centers with label $< j$ in this region. Since all centers with label $> j$ are placed only at nodes in S_{OPT} this implies that we cannot place two nodes with the same label in $N_{OPT}(v)$ (since the nodes placed in a single iteration form an independent set in G_i^2). Thus there can be at most $\alpha - j$ nodes of label $> j$ in $v \cup N_{OPT}(v) \cup V_{OPT}(v)$ from G. Adding gives at most α nodes in this region.

We now claim that if we delete $v \cup N_{OPT}(v) \cup V_{OPT}(v)$ from G, this gives us a smaller counter-example (unless the deleted nodes are exactly G, which is not a valid counter-example as we use only α nodes). □

4. α-neighbor K-suppliers problems

In this section, we give an algorithm that obtains an approximation factor of 3 for the α-neighbor K-suppliers problem.

We assume that $G = (U, V, E)$ is a complete bipartite graph with edges satisfying the triangle inequality. We place centers on the vertices in U and have to ensure that each vertex in V has α centers in its neighborhood in G_i.

Iterate for each i from 1 to m until a solution is obtained. (As before, we will find the smallest i for which we obtain a solution that uses at most K centers.) Define H_i to be the subgraph of G_i^2 induced by V and find a maximal independent set in H_i. This returns a subset $\mathscr{S} \subseteq V$. We shift these to the set $\mathscr{S}' \subseteq U$ by placing a center on each of α neighbors in U of each node in S.

Let S_{OPT} be the set of vertices in U that have centers placed on them by the optimal solution. Let $P = |S_{OPT}|$.

We first prove that $|\mathscr{S}'|$ is at most P. Each node in S has at least α neighbors in G_i that are in S_{OPT}. No other neighbor of these α nodes in S_{OPT} can be picked in S. Therefore we chose at most $\lfloor P/\alpha \rfloor$ nodes in V. Thus the algorithm chooses at most $\alpha \lfloor P/\alpha \rfloor \leqslant P$ centers.

Since each node in the independent set has at least α neighbors in U, and no two nodes in the independent set can share a neighbor in U, each node in the independent set can place a center on α of its neighbors in U.

We now prove the approximation bound of 3. If the optimal solution has cost $d(e_i)$, then in G_i we will find a valid solution as follows. Consider a node $v \in V$. If v was not chosen in S, it has a neighbor v' in H_i that is in S. Since v' has α centers adjacent to it in G_i. Each of these centers is within distance 3 from v in G_i. If v is in S, then it has α centers within distance 1.

5. Extensions and generalizations

In [10] we provide algorithms for several generalizations of these problems where we also include the notion of weights and costs, which are defined as follows. Each node has an associated "cost" for placing a center on it, and rather than limiting the number of centers, we have a limited budget [8, 13, 15]. Other generalizations include cases where the vertices have weights and we consider the weighted distance from a node to its closest center [3, 13, 15]. The methods used by Chaudhuri et al. [2] do not seem to extend to include the notion of weights or cost.

References

[1] J. Bar-Ilan, G. Kortsarz, D. Peleg, How to allocate network centers, J. Algorithms 15 (1993) 385–415.
[2] S. Chaudhuri, N. Garg, R. Ravi, The p-neighbor k-center problem, Inform. Process. Lett. 65 (1998) 131–134.
[3] M. Dyer, A.M. Frieze, A simple heuristic for the p-center problem, Oper. Res. Lett. 3 (1985) 285–288.
[4] J. Edmonds, D.R. Fulkerson, Bottleneck extrema, J. Combin. Theory, 8 (1970) 299–306.
[5] M.R. Garey, D.S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-completeness, Freeman, San Francisco, 1978.
[6] T. Gonzalez, Clustering to minimize the maximum inter-cluster distance, Theoret. Comput. Sci. 38 (1985) 293–306.
[7] D. Hochbaum, D.B. Shmoys, A best possible heuristic for the k-center problem, Math. Oper. Res. 10 (1985) 180–184.
[8] D. Hochbaum, D.B. Shmoys, A unified approach to approximation algorithms for bottleneck problems, J. ACM 33(3) (1986) 533–550.
[9] W.L. Hsu, G.L. Nemhauser, Easy and hard bottleneck location problems, Discrete Appl. Math. 1 (1979) 209–216.
[10] S. Khuller, R. Pless, Y.J. Sussmann, fault tolerant k-center problems, Technical Report CS-TR-3652, University of Maryland, College Park, 1996, Available by ftp at ftp.cs. umd.edu/pub/papers/papers/3652/3652.ps.Z.
[11] S. Khuller, Y.J. Sussmann, The capacitated k-center problem, Proc. 4th Annual European Symp. on Algorithms, Lecture Notes in Computer Science, vol. 1136, Springer, Berlin, p. 1996, 152–166.
[12] S.O. Krumke, On a generalization of the p-center problem, Inform. Process. Lett. 56 (1995) 67–71.
[13] J. Plesnik, A heuristic for the p-center problem in graphs, Discrete Appl. Math. 17 (1987) 263–268.
[14] L. Smith, Volunteers' rescue response rates worsen in Pr. William, The Washington Post, April 17, 1996.
[15] Q. Wang, K.H. Cheng, A heuristic algorithm for the k-center problem with cost and usage weights, Proc. 28th Annual Allerton Conf. on Communication, Control, and Computing, 1990, pp. 324–333.