Fault Tolerant K-Center Problems

Samir Khuller [*]
Dept. of Computer Science and UMIACS
University of Maryland
College Park, MD 20742

Robert Pless [†]
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Yoram J. Sussmann [‡]
Dept. of Computer Science
University of Maryland
College Park, MD 20742

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 a 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 ($\alpha \leq K$) 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. 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. We also provide polynomial time approximation algorithms for all the above problems for generalizations when cost and weight functions are defined on the set of vertices.

[*]Research supported by NSF Research Initiation Award CCR-9307462, and NSF CAREER Award CCR-9501355. Email addr: `samir@cs.umd.edu`

[†]Research supported by NSF Grant IRI-90-57934 Email addr: `pless@cs.umd.edu`

[‡]Research supported by NSF CAREER Award CCR-9501355. Email addr: `yoram@cs.umd.edu`

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 $\mathcal{S} \subseteq V$ of size at most K such that each vertex in V is "close" to some vertex in $\mathcal{S}$. More formally, it is defined as follows:

$$\min_{\mathcal{S} \subseteq V} \max_{u \in V} \min_{v \in \mathcal{S}} d(u, v)$$

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 [4].

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 [3, 5, 6, 7]. 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, 6, 9], including costs [6, 9] and weights [1, 9].

The α-neighbor K-center problem is discussed in a recent paper by Krumke [8]. The problem is formally defined as follows: given an edge-weighted graph $G = (V, E)$ find a subset $\mathcal{S} \subseteq V$ of size at most K such that each vertex in $V - \mathcal{S}$ is "close" to a set of α vertices in $\mathcal{S}$. Formally,

$$\min_{\mathcal{S} \subseteq V} \max_{u \in V - \mathcal{S}} \delta^{(\alpha)}(u, \mathcal{S})$$

where

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

where d is the distance function. Krumke [8] 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 [10].

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 $\mathcal{S} \subseteq V$ of size at most K such that each vertex in V is "close" to a set of α vertices in $\mathcal{S}$. Formally,

$$\min_{\mathcal{S} \subseteq V} \max_{u \in V} \delta^{(\alpha)}(u, \mathcal{S}).$$

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

$$\min_{\mathcal{S} \subseteq U} \max_{u \in V} \delta^{(\alpha)}(u, \mathcal{S}).$$

For all the problems considered in the paper, we also address the generalizations when vertices have *costs* and *weights*. The cost generalization is formally defined as follows: given a graph G and a cost function $c(v)$ defined on V, find a subset $\mathcal{S}$ of vertices of total cost at most K such that each vertex that needs to be covered by a center is "close" to a set of α vertices in $\mathcal{S}$. Formally, we have to pick a set $\mathcal{S}$ satisfying

$$\sum_{s \in \mathcal{S}} c(s) \leq K.$$

The weight generalization is formally defined as follows: given a graph G and a weight function $w(v)$ defined on V, find a subset $\mathcal{S}$ of vertices of size at most K such that each vertex that needs to be covered by a center is "close" to a set of α vertices in $\mathcal{S}$, where the distance from vertex u to vertex v depends on the weight of v. Formally, we change the definition of the distance measure to

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

where d is the distance function.

Finally, we study the most general case, when the vertices have weights and costs. The results are summarized in the table given below.

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 [7]. The algorithm is a very natural extension of the method given by Hochbaum and Shmoys [6] 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).

In addition, we provide constant approximation bounds for generalizations of the problem involving costs and weights, as well as for the α-neighbor K-suppliers problem. For the K-supplier problem, Hochbaum and Shmoys [6] give a proof (originally due to Howard Karloff) showing that the factor of 3 is the best possible unless $P = NP$. Thus 3 is a lower bound for all the K-supplier generalizations that we consider. Since the basic K-center problem with weights and costs is a generalization of the K-supplier problem, a factor of 3 is also the best possible. The results are summarized in the table below. A † indicates that the bound is the best possible unless $P = NP$, while a ‡ indicates that this matches the best known bound for the basic K-center problem. Let β denote the ratio of the maximum and minimum weights.

Fault-Tolerant K-Center Approximation Factors				
	Basic	Weights	Costs	Weights + Costs
α-All-Neighbor K-Center	$3\ (2^{\dagger})^{*}$	3	$3^{\ddagger}$	$3^{\dagger}$
α-Neighbor K-Center	$2^{\dagger}$	3	4	$4\beta + 1$
α-Neighbor K-Suppliers	$3^{\dagger}$	$3^{\dagger}$	$3^{\dagger}$	$3^{\dagger}$

* if $\alpha = 2$ or 3

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 introduced by Edmonds and Fulkerson in [2] and used for the K-center problem by Hochbaum and Shmoys in [6]. 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 $w(e_i)$. 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 [6].) 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 $w(e_i)$. 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 a 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 [5, 6] and has been used extensively to solve K-center problems. Find a maximal independent set in G_i^2. Note that if the independent set has size I, then any solution with radius $w(e_i)$ must use at least αI 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 $w(e_i)$, 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.

2.1.1. Any α with weights and costs

The α-all-neighbor K-center problem with weights and costs is a generalization of the α-all-neighbor K-center problem where weight and cost functions are defined on the vertices and the objective is to pick a set of centers whose total cost is at most K, such that the radius is minimized, where the distance from u to v is now defined by $d(u, v) \cdot w(v)$, the weight of edge $e(u, v)$ multiplied by the weight of node v.

For weighted versions of the problems, consider the list of weighted distances $w_{uv} = d(u, v) \cdot w(v)$. List them in non-decreasing order as $w_1 \leq w_2 \leq \cdots \leq w_{2m}$. For each i, define the directed graph G_i as follows. Edge $e(u, v)$ is included in G_i if $d(u, v) \cdot w(v)$, the weighted distance from u to v, is at most w_i.

A small modification of the above algorithm yields a 3-approximation algorithm for the problem with weights and costs. When choosing the initial set, we always select the highest weight available node v. We then mark all nodes whose weighted distance from v is at most $2w_i$. The directed graph ensures that, if a node u has a directed edge to node v in G_i, and v has a higher weight than u, then v also has a directed edge to u in G_i, and therefore can cover in two steps any node that u can cover in one step. Once we have placed α centers at each node v in the independent set, we simply move all α centers to the α cheapest neighbors of the node (including itself), where a neighbor of v is a node with a directed edge to v in G_i. A simple extension of the proof given in [9] shows that the vertices so obtained cover all vertices in the graph with radius at most $3w_i$. Since the node must have at least α neighbors in any solution, the cost of the solution obtained is a lower bound on the cost of any solution with radius w_i.

2.2. $\alpha = 2, 3$

If α is 2 or 3, we can obtain an approximation factor of 2. The algorithm gives an approximation ratio of 3 for any α. We can prove that for $\alpha < 4$ the obtained ratio is actually 2.

The algorithm consists of α iterations. Consider the graph G_i^2. Every node is assigned a "covering number" $C(v)$ (initially 0). The multiset of centers is $\mathcal{S} = \emptyset$. In each iteration we pick an independent set of nodes. 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.

In each iteration, when there is a choice for a vertex to be chosen as a center, we *prefer* picking vertices that do not have a center already assigned to them. We assign a center at the chosen vertex, and increase the covering number for all vertices within distance two in G_i.

```
α-ALL-NEIGHBOR K-CENTER ALGORITHM($G_i^2$).
1   for all $v$
2       $C(v) = 0$.
3       $extra(v) = 0$.
4       $helps(v) = \emptyset$.
5   for $j = 1$ to $\alpha$ do
        // Phase I
6       while $\exists v \notin \mathcal{S}$ with $C(v) < j$ do
7           create new center at $v$ and set $\mathcal{S} = \mathcal{S} \cup \{v\}$.
8           $C(v) = C(v) + 1$.
9           $C(u) = C(u) + 1$ if $(u, v) \in E(G_i^2)$.
        // Phase II
10      while $\exists v$ with $C(v) < j$ do
11          create new center at $v$ and set $\mathcal{S} = \mathcal{S} \cup \{v\}$.
12          $C(v) = C(v) + 1$.
```

```
13              $extra(v) = extra(v) + 1$.
14              if $\exists u$ with $C(u) < j$ and $(u, v) \in E(G_i^2)$
15                 $C(u) = C(u) + 1$.
16                 Set $helps(v) = u$.
17 for all $v \in \mathcal{S}$ with $extra(v) \geq 1$ do
18     if $helps(v) = \emptyset$
19        Shift $extra(v)$ centers to neighbors of $v$ in $G_i^2$ that are not in $\mathcal{S}$.
20     else
21        Shift one center to a common neighbor of $v$ and $helps(v)$ in $G_i^2$ not in $\mathcal{S}$.
22        Shift $extra(v) - 1$ centers to neighbors of $v$ in $G_i^2$ that are not in $\mathcal{S}$.
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 $w(e_i)$ must use at least αI^* centers, and we must have that $|\mathcal{S}| \leq \alpha I^*$.

□

Notice that this algorithm produces a multiset of centers. We now show how to make the centers distinct.

Theorem 2.2: *The above algorithm returns a solution to the α-all-neighbor K-center problem with an approximation ratio of two 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 $\mathcal{S}$.

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

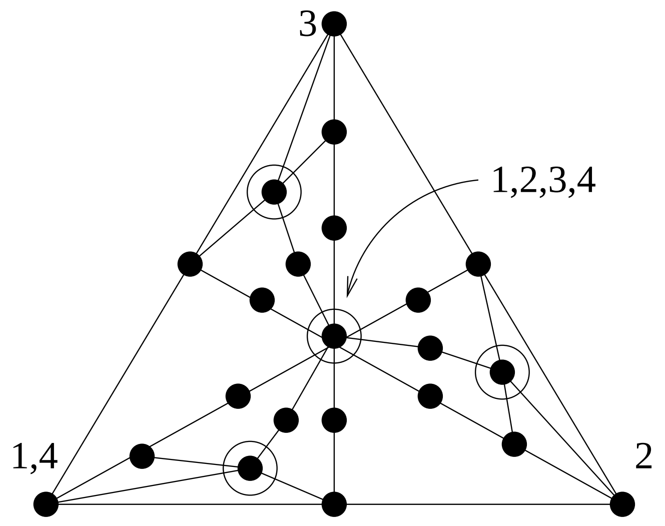

Figure 1: The circled nodes are cliques of 4 nodes, and edges to circled nodes represent edges to each of the four nodes in the clique.

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 $w(e_i)$.

□

The following example (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 8 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

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.

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

Consider the graph G_i^2. Every node is assigned a "covering number" $C(v)$ (initially 0). The set of centers is $\mathcal{S} = \emptyset$. 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 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      $C(v) = 0$.
3  for $j = 1$ to $\alpha$ do
4      while $\exists v$ with $C(v) < j$ do
5          create center at $v$ and and set $\mathcal{S} = \mathcal{S} \cup \{v\}$.
6          $C(v) = \alpha$.
7          $C(u) = C(u) + 1$ 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 two. 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 $V - S_{OPT}$ 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 $|N_{OPT}(v)| \geq \alpha$. 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).

$\square$

3.1. Any α with weights

Using a different algorithm, we can obtain an approximation factor of three for the α-neighbor K-center problem with weights. The algorithm repeatedly selects a node which is not at least α-covered as a center and increments the covering number of all nodes within distance 3 of the center.

Consider the list of weighted distances $w_{uv} = d(u,v) \cdot w(v)$. List them in non-decreasing order as $w_1 \leq w_2 \leq \cdots \leq w_{2m}$. For each i, define the directed graph G_i as follows. Edge $e(u,v)$ is included in G_i if $d(u,v) \cdot w(v)$, the weighted distance from u to v, is at most w_i.

```
α-NEIGHBOR WEIGHTED-K-CENTER ALGORITHM($G_i$).
1  for all $v$
2      $C(v) = 0$.
3  $\mathcal{S} = \emptyset$.
4  while $U = \{u \mid C(u) < \alpha\} \neq \emptyset$
5          let $v =$ max weight vertex in $U$.
6          create center at $v$ and and set $\mathcal{S} = \mathcal{S} \cup \{v\}$.
7          $C(v) = \alpha$.
8          $C(u) = C(u) + 1$ if $d(v,u) \cdot w(u) \leq 3w_i$.
9  end-proc
```

Theorem 3.2: *The above algorithm finds a solution to the α-neighbor K-center problem with weights with an approximation ratio of 3.*

Proof. Clearly this algorithm satisfies every vertex within a factor three of the optimal radius (because the algorithm loops until no vertex is left uncovered); we have to argue that it does not use too many centers. Assume on the contrary that 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. Note that each vertex in $V - S_{OPT}$ has at least α neighbors in S_{OPT}.

Define the j-neighborhood of a vertex $N^j(x) = \{v \in V \mid d(v,x) \cdot w(x) \leq j \cdot w_i\}$; intuitively, the set of all nodes which could cover x within radius $j \cdot w_i$. Define the neighborhood of a vertex $N(x) = N^1(x)$. Let $N_{OPT}(x) = N(x) \cap S_{OPT}$, the nodes in the optimal solution that cover x, and let $V_{OPT}(x) = \{v \in V \mid \exists w \in N_{OPT}(x)$ such that $d(w,v) \cdot w(v) \leq w_i\}$, or equivalently,

$V_{OPT}(x) = \{v \in V \mid N_{OPT}(v) \cap N_{OPT}(x) \neq \emptyset\}$, the nodes covered in the optimal solution by the nodes in $N_{OPT}(v)$. Since more than K centers were chosen by our algorithm, at least one center must have been chosen from $V - S_{OPT}$. Let v be the last such center.

Let u be the last center chosen from $N_{OPT}(v)$, after v was chosen. (If no such center exists, then set $u = v$.) We claim that in the set $v \cup N_{OPT}(v) \cup V_{OPT}(v)$ there are at most α centers placed by the algorithm. To see this observe that when u was placed, every center placed in the set is in $N^3(u)$ and thus at most $\alpha - 1$ centers were placed when we placed u, the last center in the set. Let x be a center placed in $V - S_{OPT}$, and let y be a vertex in $N_{OPT}(v) \cap N_{OPT}(x)$. (If $x \in S_{OPT}$ the proof is even easier.) To show that $x \in N^3(u)$, observe that $w(u) \cdot d(u,x) \leq w(u)(d(u,v) + d(v,y) + d(y,x)) \leq w(v) \cdot d(u,v) + w(v) \cdot d(v,y) + w(x) \cdot d(y,x)$. This is at most $w_i + w_i + w_i \leq 3w_i$. Therefore we can delete $v \cup N_{OPT}(v) \cup V_{OPT}(v)$ from G and obtain a smaller counter-example.

□

3.2. Any α with costs

We describe an algorithm that gives an approximation factor of 4 for the α-neighbor K-center problem with costs. We first run the α-neighbor K-center Algorithm, to obtain an initial set of centers $\mathcal{S}$. We then shift these centers to low cost neighbors as follows.

We create a bipartite graph $H = (\mathcal{S}, V, E')$, where an edge $(s,v) \in E'$ if $v = s$ or if the edge (s,v) is in G_i and the degree of s in G_i is at least α. We define cost of an edge $e = (s,v)$ to be $c(e)$, the cost of v. We then find a min-cost perfect matching M in H. Let $\mathcal{S}'$ be the set of nodes in V which are matched to a node in $\mathcal{S}$. Return the set $\mathcal{S}'$.

Theorem 3.3: *The above algorithm finds a solution to the α-neighbor K-center problem with costs with an approximation ratio of 4.*

Proof. We first show that the cost of $\mathcal{S}'$ is at most the cost of any solution with radius $w(e_i)$. Clearly a perfect matching between $\mathcal{S}$ and V exists, since each node in $\mathcal{S}$ can be matched to itself. We prove there exists a matching from $\mathcal{S}$ to the nodes in the optimal solution, which implies that the min-cost matching has cost at most that of the optimal solution. Let S_{OPT} be the set of K vertices in graph G that have centers placed on them by the optimal solution.

Consider the nodes in $\mathcal{S}$ which are also in S_{OPT}. These nodes get matched to themselves. Notice that all nodes with degree $< \alpha$ must be in S_{OPT} and these nodes are all matched to themselves in the above algorithm. Now consider a node $s \in \mathcal{S}$ not in S_{OPT}. Remove s from H and recursively find a matching in the remaining subgraph. In the neighborhood of s there are at least α centers of S_{OPT} and at most $\alpha - 1$ nodes in $\mathcal{S}$. Therefore at least one of the nodes in S_{OPT} that are in the neighborhood of s is not matched. Match s to this node.

We now prove the approximation bound. Consider a node v. If $v \notin \mathcal{S}$, then it has α neighbors in $\mathcal{S}$ in G_i^2. These neighbors are shifted by distance at most $w(e_i)$, implying that v has α neighbors in $\mathcal{S}'$ within distance $3w(e_i)$. If $v \in \mathcal{S}$, and v is matched to itself, then v is covered by itself. Otherwise, there must be a node u in the neighborhood of v in G_i which is not in $\mathcal{S}$. This node has α centers within distance $3w(e_i)$, which implies that v has α centers within distance $4w(e_i)$.

$\square$

3.3. Any α with weights and costs

A modification of the above algorithm for weights gives an approximation algorithm for the α-neighbor K-center problem with weights and costs. We first run the α-neighbor weighted-K-center Algorithm, to obtain an initial set of centers $\mathcal{S}$. We then shift these centers to low cost neighbors as in the α-neighbor K-center problem with costs.

Let β denote the ratio of the weight of the maximum weight node in G to the weight of the minimum weight node in G. A vertex that has a center placed on it may not have a center placed on it after the shifting of centers. Two points need to be noted here: if a vertex has less than α incoming edges in the directed graph G_i then we do not need to move its center. If it has at least α incoming edges, then since all those vertices either have centers, or are covered by centers, we can argue that the algorithm provides an approximation factor of $4\beta + 1$. (The problem is that these vertices may have a low weight and thus the centers that cover them may be far away.)

4. α-Neighbor K-Suppliers Problems

We assume that $G = (U, V, E)$ is a complete bipartite graph with edges satisfying the triangle inequality. Iterate for each i from 1 to m until a solution is obtained.

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

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 multiset $\mathcal{S} \subseteq V$ of centers. We shift these to the set $\mathcal{S}' \subseteq U$ by placing a center on α neighbors in U of each node in the independent set.

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

We first prove that $|\mathcal{S}'|$ is at most P. Each node chosen as a center 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. Therefore we chose at most $\left\lfloor \frac{P}{\alpha} \right\rfloor$ nodes in V. Thus the algorithm chooses at most $\alpha \left\lfloor \frac{P}{\alpha} \right\rfloor \leq P$ 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. Consider a node $v \in V$. Node v has α centers adjacent to it in H_i. Each of these centers shifts by distance at most $w(e_i)$ to a node in U, implying that v has α centers in $\mathcal{S}'$ within distance $3w(e_i)$.

4.1. Any α with weights and costs

The above algorithm can be extended to provide an approximation algorithm to the α-neighbor K-suppliers problem with weights and costs. The basic idea is to pick heaviest unmarked vertices from V to add to $\mathcal{S}$. We then mark all nodes that are close to this node, and continue until all vertices are marked. We finally pick a set of the cheapest α centers from the supplier neighbors of the vertices in $\mathcal{S}$. This algorithm gives a 3-approximation to the α-neighbor K-suppliers problem with weights and costs. When a node v is added to $\mathcal{S}$, it marks all nodes u whose weighted distance $d(v,u) \cdot w(u)$ from v is at most $2w_i$.

```
α-NEIGHBOR WEIGHTED-K-SUPPLIER WITH COSTS ALGORITHM($G, w_i$).
1  $\mathcal{S} = \emptyset$.
2  **while** there is an unmarked node in $V$ **do**
3          let $v =$ max weight unmarked vertex in $V$.
4          create center at $v$ and and set $\mathcal{S} = \mathcal{S} \cup \{v\}$.
5          mark $v$.
6          mark $u$ if $d(v,u) \cdot w(u) \leq 2w_i$.
7  **end-proc**
```

After selecting the set $\mathcal{S}$, each node in $\mathcal{S}$ places a center on each of its α cheapest neighbors in U, where a neighbor of v is a node u such that the weighted distance $d(u,v) \cdot w(v)$ from u to v is at most w_i. These nodes comprise $\mathcal{S}'$, which is the final set of centers.

For any node $v \in V$, let $N_U(v)$ be the set of nodes $u \in U$ whose weighted distance $d(u,v) \cdot w(v)$ to v is at most w_i.

Lemma 4.1: *For any $s_1, s_2 \in \mathcal{S}$, $N_U(s_1) \cap N_U(s_2) = \emptyset$.*

Proof. Consider nodes $s_1, s_2 \in \mathcal{S}$. Suppose s_1 was chosen before s_2, so that $w(s_1) \geq w(s_2)$. Assume there exists some $u \in N_U(s_1) \cap N_U(s_2)$. The distance from s_1 to s_2 can then be bounded by $d(s_1,s_2) \cdot w(s_2) \leq (d(s_1,u) + d(u,s_2)) \cdot w(s_2) \leq d(u,s_1)) \cdot w(s_1) + d(u,s_2) \cdot w(s_2) \leq 2 \cdot w_i$. Thus, s_2 would have been marked when s_1 was chosen as a center, contradicting the fact that s_2 was chosen as a center. □

Let $P = \sum_{s \in S_{OPT}} c(s)$, the cost of the optimal solution.

Lemma 4.2: *The cost of the solution returned is at most P.*

Proof. Since each node must be covered by at least α nodes in S_{OPT}, the nodes in S_{OPT} must incur a cost at least as much as that of the α lowest-cost neighbors of each node in S. Moreover (by the previous lemma), these vertices do not share any common neighbors. Thus $\sum_{s \in \mathcal{S}'} c(s) \leq P$. □

Theorem 4.3: *The above algorithm achieves an approximation ratio of 3.*

Proof. Consider a vertex $v \in \mathcal{S}$. Let u be a vertex that was marked by v. We need to argue that the weighted distance from the set of centers chosen by v to u is at most $3w_i$, where w_i is the cost of the optimal solution. Let y be a center chosen by v. We argue this as follows: $d(y,u) \cdot w(u) \leq (d(y,v) + d(v,u)) \cdot w(u) \leq d(y,v) \cdot w(v) + 2w_i \leq 3w_i$. (We used the fact that $w(v) \geq w(u)$.) $\square$

References

[1] M. Dyer and A. M. Frieze, "A simple heuristic for the p-center problem", *Operations Research Letters*, Vol 3:285–288, (1985).

[2] J. Edmonds and D. R. Fulkerson, "Bottleneck extrema", *Journal of Combinatorial Theory*, Vol 8:299-306, (1970).

[3] T. Gonzalez, "Clustering to minimize the maximum inter-cluster distance", *Theoretical Computer Science*, Vol 38:293–306, (1985).

[4] M. R. Garey and D. S. Johnson, "Computers and Intractability: A guide to the theory of NP-completeness", *Freeman, San Francisco* (1978).

[5] D. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem", *Mathematics of Operations Research*, Vol 10:180–184, (1985).

[6] D. Hochbaum and D. B. Shmoys, "A unified approach to approximation algorithms for bottleneck problems", *Journal of the ACM*, Vol 33:533–550, (1986).

[7] W. L. Hsu and G. L. Nemhauser, "Easy and hard bottleneck location problems", *Discrete Applied Mathematics*, Vol 1:209-216, (1979).

[8] S. O. Krumke, "On a generalization of the p-center problem", *Information Processing Letters*, Vol 56:67–71, (1995).

[9] J. Plesnik, "A heuristic for the p-center problem in graphs", *Discrete Applied Mathematics*, Vol 17:263–268, (1987)

[10] L. Smith, "Volunteers' rescue response rates worsen in Pr. William", *The Washington Post*, April (1996).