Facility Location with Dynamic Distance Functions

Randeep Bhatia [1] *Sudipto Guha* [2] *Samir Khuller* [3] *Yoram J. Sussmann* [4]
Dept. of Computer Science
University of Maryland
College Park, MD 20742

[1]Also with LCC Inc., 7925 Jones Branch Drive, McLean VA 22102. Email addr: `randeep@cs.umd.edu`
[2]This work was done while this author was at the University of Maryland and his research was supported by NSF Research Initiation Award CCR-9307462. Author's current address is Stanford University. Email addr: `sudipto@cs.stanford.edu`
[3]Research supported by NSF Research Initiation Award CCR-9307462, and NSF CAREER Award CCR-9501355. Email addr: `samir@cs.umd.edu`
[4]Research supported by NSF CAREER Award CCR-9501355. Email addr: `yoram@cs.umd.edu`

Abstract

Facility location problems have always been studied with the assumption that the edge lengths in the network are *static* and do not change over time. The underlying network could be used to model a city street network for emergency facility location/hospitals, or an electronic network for locating information centers. In any case, it is clear that due to traffic congestion the traversal time on links *changes* with time. Very often, we have some estimates as to how the edge lengths change over time, and our objective is to choose a set of locations (vertices) as centers, such that at *every* time instant each vertex has a center close to it (clearly, the center close to a vertex may change over time). We also provide approximation algorithms as well as hardness results for the K-center problem under this model. This is the first comprehensive study regarding approximation algorithms for facility location for good time-invariant solutions.

1. Introduction

Previous theoretical work on facility location typically has addressed situations in which we want to locate facilities in a network and optimize an objective function. The edge lengths in the network (or the distance function) are typically assumed to be *static*. In practice however, edge lengths are not static. For example, in emergency facility location, the transit time may be a function of the traffic load at the current time. The same is true for locating information centers in networks. In this paper, we define a model of dynamic distance functions and study approximation algorithms for the K-center problem, a fundamental facility location problem, under this general model. Although in reality edge lengths may behave unpredictably, we often have estimates as to how they behave as a function of time at the macro level. In particular, the transit times in a city may oscillate periodically during the day. We will therefore assume that we have some knowledge of the behavior of the edge lengths.

Our objective in this paper is to study the problem of placing facilities in such a way that *at all possible times we meet our objective criteria effectively.* We have to choose placements for the facilities, to minimize the distance d so that each node has a facility within distance d *at all possible times.*

The dynamic edge length model is a much more realistic model for studying many fundamental network problems. In this paper we initiate this study for the facility location problem and leave open a host of other problems. For example it is natural to study the problem of finding a spanning tree of the network whose maximum weight over time is minimum. Ravi et. al. [18] studied this problem in the context of bicriterion approximation and their results imply a $1+\epsilon$ approximation for this problem when there are only two "time-slots". Similar bicriterion approximation results are known for the shortest path problem [6]. Some of the other fundamental problems, such as finding a matching whose maximum weight over time is minimized, are completely open. Our experience with this general model has convinced us that even the simplest network problems are quite complex when studied under this model. For example, many of the problems which can be solved in polynomial time on a static network are NP-complete in the general model even when there are two time-slots. In addition, the techniques used for solving these problems on a static network do not easily extend to the general model.

We first discuss the basic K-center problem and then discuss our specific model and results.

K-Center Problem:

The basic K-center problem is a fundamental facility location problem [20, 5, 10, 8, 9] and is defined as follows: given an edge-weighted graph $G = (V, E)$ find a subset $S \subseteq V$ of size at most K such that each vertex in V is "close" to some vertex in S. More formally, the objective function is defined as follows:

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

where d is the distance function defined by the underlying edge weights and hence staisfies the triangle inequality. 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. Some applications are mentioned in [14, 15]. The problem is known to be NP-hard [4].

1.1. Basic Notation

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 [5, 8] that achieve an approximation factor of 2 are known to be optimal [5, 10, 8, 9]. Several approximation algorithms are known for interesting generalizations of the basic K-center problem as well [3, 9, 17, 1, 13, 12, 2, 11]. The generalizations include cases when each node v has an associated cost $c(v)$ for placing a center on it, and rather than limiting the number of centers, we have a limited budget K [9, 17]. In this case we want to find a subset $S \subseteq V$ with the property $c(S) = \sum_{s \in S} c(s) \leq K$, with the objective function

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

Other generalizations include cases where each node v has a weight $w(v)$ associated with it and we want to minimize the weighted distance from a node to its closest center [3, 17]. More formally we want to find a subset $S \subseteq V$ of size at most K, where the objective function is

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

For the basic K-center problems, $w(v) = c(v) = 1$ for every node v. For the weighted (cost) problems $c(v) = 1$ ($w(v) = 1$) for every node v.

The K-suppliers problem [9] is a variant of the K-center problem in which we are given an edge-weighted bipartite graph $G = (U, V, E)$, and we are required to find a subset $S \subseteq U$ of size at most K such that each vertex in V is "close" to some vertex of S. More formally, the objective function is defined as follows:

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

Generalizations of the K-suppliers problem involving costs and weights are defined analogously to those given above, where the cost function is now defined on U and the weight function is defined on V. We discuss the other generalizations studied in this paper in Section 2.

We address the case when the distance function *changes* with time. Assume that $d^t(u, v)$ is the distance from u to v at time t. We assume that this function is specified in some convenient manner (for example, it could be specified as a piecewise linear function, or as a step function). The function $d^t(u, v)$ could also be specified simply as a set of points $(t, d^t(u, v))$. These could be obtained for example by "sampling" transit times at different times of day. Our objective function (for the basic K-center problem) is

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

such that $|S| \leq K$.

1.2. Our Results

We study the basic K-center problem and the K-suppliers problem, as well as several variations of these problems under this model. We provide constant-factor approximation algorithms for all these problems when there are two time-slots (this models the situation of "rush hour" and "non-rush hour" travel times). For example, we could declare 7am to 9am and 4pm to 6pm as "rush hour" and all other times as "non-rush hour". (The rush hour could happen many times during the day. What is important is that there are only two distinct graphs to consider, one for rush hour and one for non-rush hour.) We provide best-possible approximation algorithms for variants of the K-suppliers problem under this model.

We also provide approximation algorithms for several variations of the K-center problem and the K-suppliers problem for arbitrarily many time-slots, including weights and costs, with factors that are close to the best possible unless $P = NP$. These results are summarized in the table below. The only known lower bound for any of the K-center problems is 2, the lower bound for the basic K-center problem. A lower bound of 3 due to Karloff for the K-suppliers problem appears in [9]. It can be shown that the K-center problem with weights and costs is a generalization of the K-suppliers problem, which implies a lower bound of 3 for this problem. For two time-slots with costs, all factors achieved in this paper match the best known factor for static distance functions.

	2 time-slots		T time-slots	
Problem	factor	lower bound	factor	lower bound
Basic + weights	3	2	$1+\beta$	$\max(2,\beta)$
Basic + weights + costs	3	3	$1+2\beta$	$\max(3,\beta)$
K-suppliers + weights + costs	3	3	$1+2\beta$	$\max(3,\beta)$
α-K-suppliers + weights + costs	3	3	$1+2\beta$	$\max(3,\beta)$
α-all-neighbor + weights + costs	3	3	$1+2\beta$	$\max(3,\beta)$
α-neighbor + costs	4	2		
Capacitated + costs	13	2		

* β is the maximum ratio of an edge's greatest length to its shortest length

What is elegant about our approach for two time-slots is that by using matchings, we can solve all the above problems in a unified manner. The algorithms for arbitrary time-slots are based on a clever extension of the Hochbaum-Shmoys method [8, 9].

We also apply the matching approach to obtain a bicriteria approximation for the asymmetric K-center problem, in which G is a directed graph. For this problem $d^t(u,v) = d^t(v,u)$ may not hold. We give an algorithm that, for any nonegative integer ν, uses at most $K\left(1+\frac{3}{\nu+1}\right)$ centers and covers all nodes within distance $c\log^* n+\nu$ times the optimal distance, where c is the constant from the set cover phase of the algorithm in [22].

Recently, we have learned that Hochbaum and Pathria [7] motivated by [19] have obtained a factor 3 approximation for the basic K-center problem as well (for 2 time-slots).

2. Preliminaries

The α-all-neighbor K-center problem with weights and costs [11, 2] is defined as follows: Given an edge-weighted graph $G = (V,E)$ with weight and cost functions defined on the vertices, find a subset $S \subseteq V$ of total cost at most K so that every vertex is "close" to at least α vertices in S. More formally,

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

where

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

The α-neighbor K-suppliers problem with weights and costs [11, 2] is defined as follows: Given an edge-weighted bipartite graph $G = (U, V, E)$ with a weight function defined on V and cost function defined on U, find a subset $S \subseteq U$ of total cost at most K such that each vertex in V is "close" to at least α vertices in S. More formally,

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

The (unweighted) α-neighbor K-center problem with costs [13, 11, 2] is defined as follows: Given an edge-weighted graph $G = (V, E)$ with a cost function defined on the vertices, find a subset $S \subseteq V$ of total cost at most K so that every vertex in $V - S$ is "close" to at least α vertices in S. More formally,

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

where

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

The (unweighted) capacitated K-center problem with costs [1, 12] requires us to place centers of total cost at most K and assign at most L vertices to each center so that (1) all nodes are assigned to a center, and (2) every vertex is "close" to its *assigned* center (not its closest center). More formally,

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

such that

$$|\{u \mid \phi(u) = v\}| \leq L \quad \forall v \in S$$

where

$$\phi : V \to S.$$

A time-slot is defined to be those instants of time over which all edge lengths are constant. We assume that time can be partitioned into T time-slots. Note that each time-slot t can be associated with a static distance function d^t, which is assumed to be distinct for each time-slot. Let β be the smallest value such that for any edge e_i, $\frac{\max_t d^t(e_i)}{\min_t d^t(e_i)} \leq \beta$. β is called the *traffic-load factor.*

A *dominating set* in a graph (digraph) is a subset S of vertices with the property that every vertex is either in S or is adjacent (has an edge) to some vertex in S. A *matching* M in a graph is a subset of edges that do not share a vertex in common. An *edge cover* S in a graph is a subset of edges such that every vertex is incident to at least one edge in S. A minimum-cost edge cover can be found in polynomial time [21, pages 580–583].

3. Hardness of Approximation for 3 time-slots

Lemma 3.1: *With three time-slots and no restriction on how the distance function can change with time, no approximation ratio for the K-center problem is possible unless $P = NP$.*

Proof. For contradiction let us assume that a polynomial time ρ-approximation algorithm exists, for some constant ρ. We show that we can use this algorithm to solve the 3-dimensional matching problem [4] in polynomial time.

Let the instance of the 3-dimensional matching problem ($3DM$) be the three sets A, B, C and a set of m ordered triples over $A \times B \times C$, where $|A| = |B| = |C| = K$. For every triple u let $u(i)$ denote its ith component. Note that $u(1) \in A, u(2) \in B$ and $u(3) \in C$. We create a graph G with m vertices, one for each ordered triple. G has an edge between every pair of vertices. We set $T = 3$. In the following we use the term vertices and triples interchangeably, as there is a one to one correspondence between them. We set:

$$d^i((u,v)) = \begin{cases} 1 & \text{if } u(i) = v(i) \\ \rho + \epsilon & \text{otherwise} \end{cases}$$

First note that any solution to the $3DM$ instance corresponds to K centers which cover the vertices of G at distance at most one, in every time-slot. Hence if the ρ-approximation algorithm returns a solution of cost more than ρ then the $3DM$ instance has no solution. Let S be the solution of cost at most ρ returned by the ρ-approximation algorithm. We show that S is also a solution to the $3DM$ instance. Let $a \in A$. Let u be a triple for which $u(1) = a$ (if such a triple does not exist then we already know that the $3DM$ instance has no solution). Since u must be covered at distance at most ρ at time-slot 1, and since all the edges at any time-slot are of length 1 or $\rho + \epsilon$, there exists a triple $v \in S$ such that $v(1) = a$. Similarly for any $b \in B$ ($c \in C$) there exists a triple $u \in S$ such that $u(2) = b$ ($u(3) = c$). Also since $|S| \leq K$, S is a solution to the $3DM$ instance. □

From the proof it is easy to conclude the following.

Corollary 3.2: *Unless $P = NP$ no polynomial time algorithm achieves a factor of $\beta - \epsilon$, for any $\epsilon > 0$, where β is the traffic-load factor.*

4. Approximation for 2 time-slots

In this section we present algorithms that find approximate solutions for several generalizations of the K-center problem with two time-slots.

We are given a graph $G = (V, E)$, and functions d^1, d^2 from the edge set E into $\mathcal{R}$, which denote the distances in the first and the second time-slots respectively.

High-Level Description:

We will use the standard approach of constructing a threshold graph as pioneered by Hochbaum and Shmoys [8, 9]. We describe the algorithm for the most general case, when each node has a weight and an associated cost to construct centers.

The algorithm operates as follows. Consider the following set of weighted edge lengths:

$$\left\{ w(u_i)d^t(u_i,u_j), w(u_j)d^t(u_j,u_i) \mid 1 \leq t \leq 2, 1 \leq \ i < j \leq |V| \right\}$$

Let $\ell_1, \ell_2, \ldots, \ell_p$ be the sorted list of these weighted edge lengths in increasing order.

The algorithm fixes a distance threshold $\delta = \ell_i$ for increasing values of i, and then considers the directed graphs G^t_δ one for each time slot $t = 1, 2$. The threshold graph G^t_δ includes only those edges whose weighted length in time slot t is at most the threshold δ. More formally $G^t_\delta = (V, E^t_\delta)$, where E^t_δ is the set of edges (u, v) with the property that $w(u)d^t(u, v) \leq \delta$. Note that if there is a solution to the original problem with distance δ, then there is a subset of vertices S, with cost at most K, such that S is a solution for each graph G_δ^t under the distance function d^t. The algorithm independently finds two solutions one for each threshold graph. Note that since the edge weights of the threshold graphs do not change with time, the algorithm can invoke existing approximation algorithms for solving the static version of the problem, for each threshold graph. The algorithm now uses the two solutions found above to reduce the problem to a minimum cost edge cover problem in an auxiliary graph to find a solution with cost K. The last step increases the approximation factor.

Note that the algorithm will search for the smallest threshold value for which it finds a solution with cost at most K. This can be done by linear or binary search on the list of edge lengths.

We illustrate this general technique in the following sections.

4.1. Basic K-centers

We first find potential center locations in each graph G_δ^t for $t = 1, 2$, using the Hochbaum-Shmoys method[1]. Assume that all vertices in G_δ^t are unmarked initially and $M_t = \emptyset$. We repeatedly pick an unmarked vertex v with maximum weight and place a center on it, and also add it to M_t. We mark all vertices within weighted distance 2δ from v. Specifically, we mark all vertices x such that $w(x)d^t(x, v) \leq 2\delta$.

We now construct a new auxiliary graph G' on the vertex sets M_1 and M_2. A vertex in each set contributes to a distinct node in G'.

We define *neighborhood* as follows: Vertex z is in the neighborhood of $x \in M_t$ in time-slot t if $w(x)d^t(x, z) \leq \delta$.

Note that by construction a vertex can be in the neighborhood of only one vertex in M_t in a time-slot. If it was in the neighborhood of two nodes in a time-slot, say of x and y, then the first node to be added to M_t would mark the other node. (If $w(x) \geq w(y)$ then x is chosen before y and since $w(y)d^t(x, y) \leq w(y)d^t(y, z) + w(x)d^t(x, z) \leq 2\delta$, y will get marked when x is chosen.)

If a vertex z belongs to the neighborhood of $x \in M_1$ and $y \in M_2$, create an edge $e_z = (x, y)$ in G'. If a vertex z belongs to the neighborhood of $x \in M_1$ (M_2) but of no vertex in M_2 (M_1), create a self loop $e_z = (x, x)$ in G'. The cost of edge e_z is $c(z)$.

Find a minimum cost edge cover C of this graph G', and let V_C be the vertices corresponding to the labels on the edges in C. If $c(V_C) \leq K$, return V_C as the solution. Otherwise repeat the above procedure for a higher threshold value.

Theorem 4.1: *The above algorithm yields a solution of distance at most 3 times the distance for the optimal placement of centers.*

Proof. Assume there is a placement of centers with total cost at most K, such that every vertex has weighted distance at most δ from its nearest center in either time-slot. Let the set of centers be OPT. The placement of the centers has to be such that each center in OPT appears in the neighborhood in G_δ^1 (G_δ^2) of only one node in M_1 (M_2).

For each center $z \in OPT$, choose the edge e_z in G'. This set of edges covers all the nodes in G', guaranteeing the existence of an edge cover of cost at most K.

[1]Strictly speaking, their method does not address the case when nodes have weights, but can be easily extended to handle node weights.

The distance bound follows from the fact that in each time-slot t and for every vertex $v \in V$, there is a vertex u in M_t within weighted distance at most 2δ from v (by the maximality of M_t). Since that node of M_t is covered by some edge e_z in C, it is at most weighted distance δ from the node in V_C that corresponds to that edge. Thus $w(v)d^i(v,z) \leq w(v)d^i(v,u)+w(u)d^i(u,z) \leq 2\delta+\delta = 3\delta$. Hence all the vertices in G are at most distance 3δ from some node in V_C in each time-slot. ∎

Remark. We showed above how to obtain a solution to the K-center problem with two time-slots whose distance is at most 3 times the optimal. However, it is possible that one time-slot will have many short edges, in which case the value of δ that we use could be much larger than necessary *for that time-slot.* For example, if one time-slot represents rush hour and the other represents non-rush hour, then distances will be much shorter in the second time-slot. We cannot avoid long distances during rush hour, but we would like everyone to have a center nearby (in terms of travel time) during non-rush hour, since there is no reason not to.

We can guarantee this by using two different values δ_1 and δ_2, one for each time-slot, and solving each time-slot separately as before. To combine the solutions, we make the definition of *neighborhood* depend on the time-slot. That is, we say that vertex z is in the neighborhood of $x \in M_t$ in time-slot t if $w(x)d^t(x,z) \leq \delta_t$. Thus in each time slot t the distance is at most $3\delta_t$.

4.2. K-suppliers

The above algorithm can easily be generalized to obtain a factor 3 algorithm for the K-suppliers problem with weights and costs, using the approach of Hochbaum-Shmoys. We first pick vertex sets M_t in each graph G^t_δ *only among nodes in* V using the same method as before. We now construct G' as follows: if a vertex $u \in U$ belongs to the neighborhood of $x \in M_1$ and $y \in M_2$, create edge $e_u = (x,y)$ in G'. If u belongs to the neighborhood of $x \in M_1$ (M_2) but of no vertex in M_2 (M_1), create a self loop $e_u = (x,x)$ in G'. The cost of edge e_u is $c(u)$. We now find a minimum-cost edge cover as before to obtain our approximate solution.

Theorem 4.2: *A solution to the K-suppliers problem with weights and costs with distance at most 3 times the distance for the optimal placement of centers, can be found in polynomial time.*

4.3. Fault tolerant K-Centers

4.4. α-neighbor K-centers

In this section we show how to use the high-level approach given above to obtain an algorithm for the α-neighbor K-center problem with costs for two time-slots.

In this case all node weights are equal to 1 and so the graphs G^δ_t are undirected. *The main difficulty is that the edge-cover approach used in the previous section does not work since a vertex may be adjacent to more than one center in the set S_i. (We can easily reduce this to a set-cover instance, but that problem cannot be solved optimally in polynomial time.) We are finally able to achieve this by a reduction to min cost perfect matching in an auxiliary graph.*

First construct approximate solutions S_1 and S_2 independently for the graphs G^1_δ and G^2_δ using the algorithm for the α-neighbor K-center problem given in [11]. (Each vertex either has a center placed on it, or at least α centers within distance two.)

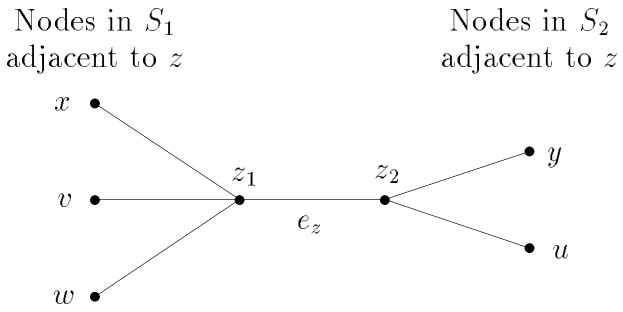

Figure 1: Gadget to replace edges

We now construct an auxiliary graph G' using the vertex sets S_1 and S_2. The vertex set for G' consists of all the vertices in S_1 and S_2 as well as two copies of V, which we will call V_1 and V_2. A vertex appearing in multiple sets will have multiple instantiations in G'. There is also a vertex set C with $|S_1| + |S_2|$ vertices.

We introduce the following gadget for each vertex in V. For each $v \in V$, let its two copies be $v_1 \in V_1$ and $v_2 \in V_2$. Connect v_1 and v_2 by an edge e_v of cost 0 in G'.

We redefine *neighborhood* as follows: Vertex z is in the neighborhood of $x \in S_t$ in time-slot t if $x = z$ or if $d^t(z, x) \leq \delta$ and the degree of x in G^t_δ is at least α. If a vertex $z \in V$ belongs to the neighborhood of $x \in S_1$, create edge $e_{xz} = (x, z_1)$, and similarly if z belongs to the neighborhood of $y \in S_2$, create edge $e_{yz} = (y, z_2)$ in G'. Assign all such edges cost $c(z)$ (see Fig. 1).

Finally, let C be a clique with edges of cost 0, and connect each vertex $x \in V_1 \cup V_2$ to all vertices in C with edges of cost $c(x)$. This clique guarantees the existence of a perfect matching. Find a min-cost perfect matching in G'. The solution returned by the algorithm is the set of nodes in V_1 (equivalently V_2) that are not matched to their counterparts in V_2 (V_1).

Theorem 4.3: *The above algorithm yields a solution with distance at most 4 times the distance for the optimal placement of centers.*

Proof. A proof by induction shows the existence of a matching from S_1 and S_2 to the set of centers in the optimal solution in V_1 and V_2. By construction of G', the cost of the perfect matching is exactly twice that of the set of centers selected by the matching. Observe that any node in S_1 (S_2) with degree in G^1_δ (G^2_δ) less than α *must* be in the optimal solution, and these nodes are matched to their counterparts by the algorithm. Thus the cost of the solution returned is at most the cost of the optimal solution.

To prove the distance bound, first note that the algorithm in [11] returns a solution to the α-neighbor K-center problem with an approximation factor of 2. The solution returned by the above matching procedure selects new centers in the neighborhoods of the original centers. Nodes which were not chosen as centers in the first phase now have at least α centers at distance 3δ rather than 2δ. A node v which was chosen as a center in the first phase but did not match to its counterpart must have had at least one node which was not a center in its neighborhood. This node has at least α centers within distance 3δ, implying that v has at least α centers within distance 4δ. □

4.4.1. α-all-neighbor K-centers

We can easily generalize the algorithm of Section 4.1 for the α-all-neighbor K-center problem with weights and costs. In this case the existence of a solution guarantees the existence of an α-edge cover (where each vertex has degree $\geq \alpha$) in the graph G'. Such a cover will provide a solution in which every vertex has at least α centers within a distance 3 times that of optimal. Since the α-edge cover problem can be solved in polynomial time even with costs [21, pages 580–583], we can find a minimum cost α-edge cover in G' and obtain an approximate solution to the α-all-neighbor K-center problem with weights and costs.

Theorem 4.4: *A solution to the α-all-neighbor K-center problem with weights and costs with distance at most 3 times the distance for the optimal placement of centers can be found in polynomial time.*

4.4.2. α-neighbor K-suppliers

By combining the techniques of Sections 4.2 and 4.4.1, we can obtain a 3-approximation algorithm for the α-neighbor K-suppliers problem with weights and costs.

Theorem 4.5: *A solution to the K-suppliers problem with weights and costs with distance at most 3 times the distance for the optimal placement of centers can be found in polynomial time.*

4.5. Asymmetric K-centers

The asymmetric K-center problem requires us to place at most K centers on a directed graph so that all vertices are "close" to some center, where distance is measured by the length of the edge *from* the center *to* the vertex. Since we can try all possible threshold values (there are at most $2n^2$ distinct distances), we assume that we construct the threshold digraphs G_1 and G_2 for time-slots 1 and 2 respectively with the optimal radius.

We use a variant of the algorithm described in [22] to solve the asymmetric K center problem for each time-slot that guarantees a factor of $O(\log^* n)$. The algorithm developed by Panigrahy [16] (also described in [22]) uses a set cover approach to obtain a solution with $2K$ centers with an approximation factor of $O(\log^* n)$. We refer to the vertices chosen to be centers for the two time-slots as S_1 and S_2 respectively. We are left with at most $4K$ "candidate" centers.

Suppose each S_t satisfies the following property: for any pair of vertices $x, y \in S_t$, there is no vertex z $s.t.(z, x), (z, y) \in G_\delta^t$ (recall that G_δ^t has self-loops as well). Then there is an easy way of solving the problem. The existence of K centers guarantees that there are K nodes from which these vertices are reachable in one step. Since a vertex has an edge to at most one candidate for each time-slot, it can "cover" at most two candidates (one from each S_t). In fact, in this case clearly $|S_t| \leq K$.

We construct a graph G' on the vertices $S_1 \cup S_2$. For each vertex v which can "cover" two candidates, we join the candidates by an edge labeled e_v. If some vertex v' "covers" only one candidate, we introduce a self loop labeled $e_{v'}$. We now compute a minimum edge cover in G'. The vertices corresponding to the labels of the edges chosen form the solution and all the nodes in each time-slot are serviced within a distance of $O(\log^* n) + 1$ times the threshold.

Hence the first part of the algorithm will be to modify the candidate sets so that the condition is satisfied.

Stage 1. Fix a threshold δ and construct the threshold graphs G_δ^1 and G_δ^2. Let ν be a nonnegative integer.

Stage 2. Use a repeated set cover algorithm to find $2K$ centers for the digraph G_δ^1. Call this set of vertices S_1. Similarly for G_δ^2 find a set S_2 of $2K$ centers.

Stage 3.

for each $t = 1, 2$ **do**
 Set $S'_t = S_t$.
 for all nodes $v \in S'_t$, set $level(v) = 0$.
 while there are two nodes $x, y \in S'_t$ such that for some z, $(z, x) \in G_\delta^t$ and $(z, y) \in G_\delta^t$
  $S'_t \leftarrow S'_t - \{x, y\}$.
  $level(z) = \max(level(x), level(y)) + 1$.
  if $level(z) = \nu$ **then** $T_t \leftarrow T_t \cup \{z\}$ **else** $S'_t \leftarrow S'_t \cup \{z\}$

Stage 4. Construct a graph on the vertices $S'_1 \cup S'_2$. We say that a vertex v *covers* a candidate vertex $x \in S_t$ iff v has an edge to x in graph G_δ^t. For each vertex v which can cover two candidates, we join the candidates by an edge labelled e_v. If some vertex v' covers only one candidate, we introduce a self loop labeled $e_{v'}$. We now compute a minimum edge cover in this graph. Let S denote the vertices corresponding to the edges in the edge cover of $S'_1 \cup S'_2$.

Stage 5. Our final solution is $S \cup T_1 \cup T_2$.

Lemma 4.6: *At the end of Stage 3 we have* $(\nu + 1) \cdot |T_t| + |S'_t| \leq |S_t|$.

Proof. The proof of this lemma requires the observation that for each node at level $j (< \nu)$, the size of S'_t has decreased by at least j. This is easily proved by induction as follows. For $j = 1$ this is trivial. Assume it is true for all $j \leq m$. Then for $j = m + 1$, let x and y be the two nodes who were responsible for this node to become a member of S_t, and assume without loss of generality that $level(x) = m$. Then the size of S'_t has already decreased by m because of x, and in the current iteration it decreases by one.

However, for each node z in T_t, the size of S'_t decreases by at least $\nu + 1$, since just previously to the selection of z, the size of S'_t has decreased by at least $\nu - 1$, and when z becomes a member of T_t, the size of S'_t decreases by two. Since the initial size of S'_t was $|S_t|$, the lemma follows. ∎

We now prove the following theorem.

Theorem 4.7: *The above algorithm uses at most* $K\left(1 + \frac{3}{\nu+1}\right)$ *centers. and every node has a center within distance* $O(\log^* n) + \nu$ *times the optimal, in each time-slot.*

Proof. Using the above lemma, $(\nu + 1)(|T_1| + |T_2|) + |S'_1| + |S'_2| \leq |S_1| + |S_2| \leq 4K$, after stage 3. We can find an edge cover in the graph on $S'_1 \cup S'_2$ of size at most $|S'_1| + |S'_2|$, (where $|S'_t|$ is the size of S'_t after stage 3). Plugging this inequality in the expression above,

$$(\nu + 1)(|T_1| + |T_2|) + |S| \leq 4K$$

Since $|S| \leq K$ we have $\nu|S| \leq \nu K$. Adding the two equations and dividing by $\nu + 1$, we get $|S| + |T_1| + |T_2| \leq K\left(1 + \frac{3}{\nu+1}\right)$.

In time-slot t, every node is within distance $O(\log^* n)$ times the threshold of some candidate center x. After stage 3, x is reachable either within $\nu - 1$ steps from some node in S'_t, or within ν steps from a node in T_t. If x is reachable within $\nu - 1$ steps from some node in S'_t, then x is reachable within ν steps from some node in S. Thus x is reachable within ν steps from some node in $S \cup T_t$. Hence every node is within $O(\log^* n) + \nu$ steps from a node in $S \cup T_t$. □

5. Approximation for bounded variance on edge-lengths

In this section we consider the case when there are arbitrarily many time-slots. We restrict our attention to instances of the problem in which the ratio of an edge's maximum length to its minimum length is bounded by some constant β. This section considers the case of distinct time-slots with edge lengths given by a step function.

5.1. Basic K-centers

For each edge e_i we assume that

$$\max_t d^t(e_i) \leq \beta \cdot \min_t d^t(e_i).$$

We first give an algorithm for the weighted version (weights on the nodes), and then show how to modify it when there are costs associated with building centers as well. We note that getting an approximation factor of 2β is trivial by a direct extension of the algorithm for the K-center problem with static distance functions. We present a better algorithm with an approximation ratio of $1 + \beta$.

List all the weighted edge lengths

$$\left\{ w(u_i)d^t(u_i, u_j), w(u_j)d^t(u_j, u_i) \mid 1 \leq t \leq T, 1 \leq i < j \leq |V| \right\}$$

Let $\ell_1, \ell_2, \ldots, \ell_p$ be the sorted list of these weighted edge lengths in increasing order. We will use the standard approach of constructing a threshold graph as pioneered by Hochbaum and Shmoys [8, 9].

The main idea behind the algorithm is to fix a distance threshold $\delta = \ell_i$ for increasing values of i, and to then consider the directed graph G^t_δ. For each time-slot t, let $G^t_\delta = (V, E^t_\delta)$, where E^t_δ is the set of edges (v, u) with the property that $w(v)d^t(v, u) \leq \delta$. (Note that by this definition, the self-loop (v, v) is also included.) If there is an optimal solution with optimal weighted distance δ, then there is an optimal subset S^* of K vertices that forms a dominating set for each graph G^t_δ. (In other words, at any point of time, each vertex is either in the dominating set or has an edge to some vertex in the dominating set.)

The algorithm works as follows: consider a vertex v. Assume that v has an edge to $u \in S^*$ at time-slot t. If we place a center at v for time-slot t, we would like to "cover" all the vertices that are covered by u during any time-slot. If we pick the heaviest unmarked vertex (at some time-slot t) at each step, we can cover all unmarked vertices that can cover it. By our choice of v, all the vertices covered by u can reach v by using at most two edges: one edge from E^t_δ and one edge from $E^{t'}_\delta$ for any time-slot t'. So we "mark" v together with all vertices w such that w can reach v by using at most two such edges. We mark the nodes for the time-slots during which they are covered: the goal is for every node to be marked during every time-slot. This guarantees that we cover each vertex within a distance of $(1+\beta)\delta$.

Time-Invariant Bounded K-centers(G, δ).

$S = \emptyset$.
for $1 \le t \le T$
  for all v
    $marked^t(v) = \text{FALSE}$.
while $\exists v \notin S, t \in [1, T]$ with $marked^t(v) = \text{FALSE}$ **do**
  let v, t be such a pair for which $w(v)$ is maximized.
  create center at v and set $S = S \cup v$.
  for $(v, u) \in E^t_\delta$
    for $1 \le t' \le T$
      set $marked^{t'}(u) = \text{TRUE}$. (* $w(u)d^{t'}(u,v) \le w(v)\beta d^t(u,v) \le \beta\delta$. *)
      for $(w, u) \in E^{t'}_\delta$
        set $marked^t(w) = \text{TRUE}$. (* $w(w)d^t(w,v) \le w(w)d^t(w,u) + w(w)d^t(u,v) \le \beta\delta + \delta$. *)
        set $marked^{t'}(w) = \text{TRUE}$. (* $w(w)d^{t'}(w,v) \le w(w)d^{t'}(w,u) + w(w)d^{t'}(u,v) \le \delta + \beta\delta$. *)
  for $1 \le t' \le T$
    for $(v, u) \in E^{t'}_\delta$
      set $marked^t(u) = \text{TRUE}$. (* $w(u)d^t(u,v) \le w(v)\beta d^{t'}(u,v) \le \beta\delta$. *)
      set $marked^{t'}(u) = \text{TRUE}$. (* $w(u)d^{t'}(u,v) \le w(v)d^{t'}(u,v) \le \delta$. *)
      for $(w, u) \in E^t_\delta$
        set $marked^t(w) = \text{TRUE}$. (* $w(w)d^t(w,v) \le w(w)d^t(w,u) + w(w)d^t(u,v) \le \delta + \beta\delta$. *)
        set $marked^{t'}(w) = \text{TRUE}$. (* $w(w)d^{t'}(w,v) \le w(w)d^{t'}(w,u) + w(w)d^{t'}(u,v) \le \beta\delta + \delta$. *)

Theorem 5.1: *The above algorithm returns a solution to the weighted time-varying facility location problem with an approximation ratio of $1+\beta$, where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

Proof. Consider a vertex $x \notin S$. Let $1 \le t \le T$ and $marked^t(x)$ be set TRUE for the first time in the WHILE loop when a center is placed at $v \in S$ by the algorithm. Note that $w(x) \le w(v)$. We have to argue that for each vertex that is marked for time-slot t, we can guarantee that x is within a weighted distance $(1+\beta)\delta$ of $v \in S$ in time slot t. There are a few cases, and the proof for each case is shown as part of the pseudo-code in comments.

Since when the algorithm terminates all nodes not in S are marked for every time-slot, S dominates all vertices, in every time-slot, within a weighted distance of $(1+\beta)\delta$.

It remains to show that if $|S| > K$, then the optimal solution using K centers has weighted distance strictly greater than δ.

Consider a vertex $v \in S$ and a vertex u in the optimal solution that covers v at some time t (u could be v). We will show that the selection of vertex v as a center, along with the choice t for time-slot, causes any node w covered by u at *any* time t', to be marked for t'. We have edge $(v, u) \in E^t_\delta$ and edge $(w, u) \in E^{t'}_\delta$. Therefore vertex w is marked for time-slot t' (see Fig. 2). Since the selection of vertex v as a center, along with the choice t for time-slot, causes all vertices that are covered by u in any time-slot t' to be marked for time-slot t', the size of the optimal solution is at least $|S|$. ∎

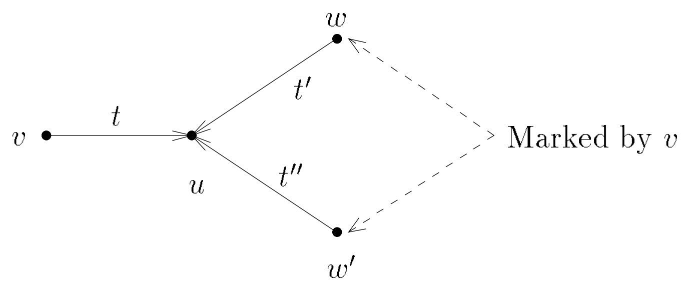

Figure 2: Nodes marked by v

5.1.1. The cost case

Here we assume that vertices have cost and we have a limited budget to spend on the centers. Our algorithm for the cost case first runs the algorithm for the weighted case to get the set S of centers, and then just shifts each center in S to a node with the least cost, among all nodes that it has edges to (including itself). Let S' be the resulting set of centers.

Theorem 5.2: *The above algorithm returns a solution to the time-varying facility location problem with costs, with an approximation ratio of* $1 + 2\beta$, *where* $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.

Proof. Note that by the proof of Thm. 5.1, in every time-slot each node is within a distance $(1+\beta)\delta + \beta\delta$ of some node in S'. Also by the proof of Theorem 5.1, for each $v \in S$ there is a distinct optimal center x_v, which is a neighbor of v in some time-slot. Hence if v is shifted to v' by the algorithm then $c(v') \leq c(x_v)$. Summing over all the nodes in S we get that the cost of the nodes in S' is a lower bound on the optimal cost. ∎

5.2. K-suppliers

The above algorithm can easily be generalized to obtain a factor $1+2\beta$ algorithm for the K-suppliers problem with weights and costs.

As before, we fix a threshold δ and try to find a solution with cost at most K. For each time-slot t, let $G^t_\delta = (U, V, E^t_\delta)$, where E^t_δ is the set of edges (v, u) with the property that $w(v)d^t(v,u) \leq \delta$. If there is an optimal solution with optimal weighted distance δ, then there is an optimal subset of K vertices chosen from U, such that for each graph G^t_δ, all nodes in V are dominated. (In other words, at any point of time, each vertex in V has an edge to some vertex in the dominating set chosen from U.)

The algorithm works as follows: consider a maximum weight unmarked vertex $v \in V$ (in some time-slot t). Assume that v has edges to nodes in U at time-slot t. We pick the lowest cost node u that v has an edge to in time-slot t and place a center there. This is a lower bound on the optimal solution's cost to cover v at time t. The optimal solution may place a center at a vertex u' to cover v in time-slot t. We would like to "cover" all the vertices w that are covered by u' during *any* time-slot. By our choice of v, such vertices w can reach u, where we placed a center, by using at most three edges: one edge (w, u') from $E^{t'}_\delta$ for any time-slot t', and two edges from E^t_δ (from u'

to u). So we "mark" all nodes w such that (w, u') and (v, u') are edges in $E_\delta^{t'}$ and E_δ^t respectively. We mark the nodes for the time-slots during which they are covered: the goal is for every node to be marked during every time-slot. This guarantees that we cover each vertex within a distance of $(1 + 2\beta)\delta$.

Theorem 5.3: *We can find in polynomial time a solution to the K-suppliers problem with weights and costs with distance at most $1 + 2\beta$ times the distance for the optimal placement of centers where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

5.3. Fault tolerant K-centers

5.3.1. α-all-neighbor K-centers

We can extend the algorithm in Section 5.1 to provide a $1 + 2\beta$ algorithm for the α-all-neighbor K-center problem with weights and costs. Select an initial set of nodes as in the algorithm for the weighted basic K-center problem. Then for each node v in the initial set, place α centers on the α cheapest neighbors of v (including itself) in *any* time-slot.

Theorem 5.4: *The above algorithm returns a solution to the α-all-neighbor K-center problem with weights and costs, with an approximation ratio of $1 + 2\beta$, where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

Proof. Note that by the proof of Thm. 5.1, in every time-slot each node is within a distance $(1 + \beta)\delta + \beta\delta$ of some node in S'. Also a simple extension of the proof of Theorem 5.1 shows that for each $v \in S$ there are α distinct optimal centers $x_v^1, \ldots, x_v^\alpha$, which are neighbors of v in some time-slot. Hence if the algorithm places centers on $v_1', \ldots, v_\alpha'$ then $\sum_i c(v_i') \leq \sum_i c(x_v^i)$. Summing over all the nodes in S we get that the cost of the nodes in S' is a lower bound on the optimal cost. □

5.3.2. α-neighbor K-suppliers

A straightforward extension of the algorithm in Section 5.2 yields the following result.

Theorem 5.5: *We can obtain an algorithm for the α-neighbor K-suppliers problem with weights and costs, with an approximation ratio of $1 + 2\beta$, where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

5.3.3. α-neighbor K-centers

By only considering the edge weights in one time-slot, and running the algorithm given in [11], we can obtain the following result.

Theorem 5.6: *We can obtain an algorithm for the α-neighbor K-center problem with weights and costs, with an approximation ratio of 4β, where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

Proof. The algorithm given in [11] gives a 4-approximation to the α-neighbor K-center problem with weights and costs for one time-slot. Each edge weight increases by at most a factor of β in any time-slot, resulting in a 4-approximation for all time-slots. □

5.4. Capacitated K-centers

Theorem 5.7: *We can obtain an algorithm for the capacitated K-center problem with all costs equal to 1, with an approximation ratio of 6β, where $\beta = \max_e \frac{max_t d^t(e)}{min_t d^t(e)}$.*

References

[1] J. Bar-Ilan, G. Kortsarz and D. Peleg, "How to allocate network centers", *J. Algorithms*, 15:385–415, (1993).

[2] S. Chaudhuri, N. Garg, and R. Ravi, "Best possible approximation algorithms for generalized k-Center problems", Technical Report MPI-I-96-1-021, Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany, (1996).

[3] M. Dyer and A. M. Frieze, "A simple heuristic for the p-center problem", *Operations Research Letters*, Vol 3:285–288, (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] T. Gonzalez, "Clustering to minimize the maximum inter-cluster distance", *Theoretical Computer Science*, Vol 38:293–306, (1985).

[6] R. Hassin, "Approximation schemes for the restricted shortest path problems", *Mathematics of Operations Research*, Vol 17:36-42, No 1. Feb. 1992.

[7] D. Hochbaum, personal communication, Oct (1996).

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

[9] D. Hochbaum and D. B. Shmoys, "A unified approach to approximation algorithms for bottleneck problems", *Journal of the ACM*, Vol 33(3):533–550, (1986).

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

[11] S. Khuller, R. Pless, and Y. J. Sussmann, "Fault tolerant K-Center problems", *Proc. of the 3^{rd} Italian Conference on Algorithms and Complexity*, LNCS 1203, pages 37–48, (1997).

[12] S. Khuller and Y. J. Sussmann, "The capacitated K-Center problem", *Proc. of the 4^{th} Annual European Symposium on Algorithms*, LNCS 1136, pages 152–166, (1996).

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

[14] H. L. Morgan and K. D. Levin, "Optimal program and data locations in computer networks", *Communications of the ACM*, Vol 20:315–322, (1977).

[15] K. Murthy and J. Kam, "An approximation algorithm to the file allocation problem in computer networks", *Proc. of the 2^{nd} ACM Symposium on Principles of Database Systems*, pages 258–266, (1983).

[16] R. Panigrahy, "An $O(\log n)$ approximation algorithm for the asymmetric p-center problem", *manuscript*, 1995.

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

[18] R. Ravi, M. X. Goemans, "The constrained minimum spanning tree problem", *SWAT 1996*, 66-75.

[19] D. Serra and V. Marianov, "The P-median problem in a changing network: The case of Barcelona", paper presented at the *International Symposium in Locational Decisions VII, (ISOLDE)*, Edmonton, Canada, (1996).

[20] C. Toregas, R. Swain, C. Revelle and L. Bergman, "The location of emergency service facilities", *Operations Research*, Vol 19:1363–1373, (1971).

[21] J. Van Leeuwen, Ed., *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, The MIT Press/Elsevier, 1990.

[22] S. Vishwanathan, "An $O(\log^* n)$ approximation algorithm for the asymmetric p-Center problem", *Proc. of the* 7^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–5, (1996).