Minimizing the Diameter of a Network using Shortcut Edges

Erik D. Demaine and Morteza Zadimoghaddam

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Cambridge, MA 02139, USA
{edemaine,morteza}@mit.edu

Abstract. We study the problem of minimizing the diameter of a graph by adding k shortcut edges, for speeding up communication in an existing network design. We develop constant-factor approximation algorithms for different variations of this problem. We also show how to improve the approximation ratios using resource augmentation to allow more than k shortcut edges. We observe a close relation between the single-source version of the problem, where we want to minimize the largest distance from a given source vertex, and the well-known *k-median* problem. First we show that our constant-factor approximation algorithms for the general case solve the single-source problem within a constant factor. Then, using a linear-programming formulation for the single-source version, we find a $(1+\varepsilon)$-approximation using $O(k \log n)$ shortcut edges. To show the tightness of our result, we prove that any $(\frac{3}{2}-\varepsilon)$-approximation for the single-source version must use $\Omega(k \log n)$ shortcut edges assuming $\mathrm{P} \neq \mathrm{NP}$.

Keywords: approximation algorithms, network design, network repair

1 Introduction

Diameter is an important metric of network performance, measuring the worst-case cost of routing a point-to-point message or a broadcast. Such communication operations are ubiquitous in a variety of networks, such as information networks, data networks, telephone networks, multicore networks, and transportations networks. In information networks, search engines need to access all nodes (or sometimes just "important" nodes) in the shortest possible time; nodes might represent webpages and edges links. We can also see this problem as the information diffusion time in information networks [8, 9]. In transportation networks, passengers want short commutes. In telephone networks, we want to reduce the length of the paths between the nodes to reduce connection lag. In multicore processors, we want to build an underlying network to have short paths between different cores [2]; in many cases, the bottleneck in running time is the time spent on communication between cores.

Each of these applications has several constraints on the network design, from existing infrastructure to connectivity or fault tolerance. Minimizing diameter may be at odds with some of these constraints, yet small diameter remains important. Thus we consider the problem of augmenting an existing network design with a limited number of additional edges to reduce the diameter as much as possible.

Many variations are also of interest. We might want to reduce the shortest-path distances among the nodes in a just special subset of the nodes. In a telephone or transportation network, a company might only care about distances among their own nodes (e.g., phone numbers or airports), though doing so might require adding edges (e.g., cables or flights) anywhere in the network. In an information network, we might ignore all spam pages.

In the single-source version of the problem, a node wants to construct edges in order to minimize its distances from the other nodes. This problem has been considered in selfish agent networks [1, 4–6], where every node simultaneously tries to solve the single-source problem. Agents have high incentive to join social networks with low diameter because messages spread in short time with small delays. Because the budget of these selfish agents is limited in many applications [10], they can not add more than a few edges, and they want to minimize their distances to the other nodes.

Model. In all these applications, we can assume that we are given a weighted undirected graph $G = (V, E, \ell)$, a positive integer k, and a nonnegative real number δ. The length of edge e is represented by $\ell(e)$. Our goal is to add k shortcut edges of length δ in order to minimize the diameter of the resulting graph. Recall that the diameter of a graph is the maximum distance between two nodes, and the distance between two nodes is the length of the shortest path between them. In most applications, including [11], δ is a small constant compared to the diameter of the graph.

Related Work. Meyerson and Tagiku [11] considered the problem of minimizing the average distance between the nodes instead of the maximum distance. This is the only work that considers the problem with a hard limit on the budget (the number of edges we can add). They obtained several constant-factor approximations using the *k-median with penalties* problem. They also improved the best known approximation ratio for metric k-median with penalties, to get better approximation factors for the other problems they consider. If α denotes the best approximation known for metric k-median with penalties, they presented an α-approximation for the single-source average-shortest-path problem, and a 2α-approximation for the general average shortest-path problem.

Our Results. We start with a simple clustering algorithm, and find a lower bound on the diameter of optimum solution. In Section 2, we find a $(4 + \varepsilon)$-approximation algorithm (using at most k shortcut edges).

Next we study approximation algorithms with *resource augmentation*: by allowing the algorithm to add more than k edges, but still comparing to the optimal solution with just k edges, we can decrease the approximation ratio. To do so, we study the structure of the optimum solutions in more detail to get a better lower bound. In Section 3, we obtain a $(2+\varepsilon)$-approximation using at most $2k$ shortcut edges for small values of δ. Previous work assumes that δ is zero for simplicity [11], and in most of the applications, it is negligible comparing to the diameter of the graph.

In Section 4, we study the single-source version of the problem in which we want to minimize the maximum distance of all nodes from a specified node, called the *source*, by adding k shortcut edges. We prove that any α-approximation algorithm for the original problem (minimizing diameter of the whole graph) can be seen as a 2α-approximation for the single-source version of the problem. We present linear-programming approaches to get better approximations for the single-source version with resource augmentation. We obtain a $(1+\varepsilon)$-approximation for the single-source problem using $O(k \log n)$ shortcut edges. Our linear program is similar to that for the k-median problem studied in [3]. To show the optimality of our algorithm, we prove that any $(\frac{3}{2}-\varepsilon)$-approximation for the single-source problem uses at least $\Omega(k \log n)$ shortcut edges assuming P $\neq$ NP.

In Section 5, we consider the multicast version of our problems in which only a given subset of nodes is important, and we want to reduce the maximum distance between the nodes in the given subset. This problem also has a single-source variant in which we want to minimize the maximum distance of the nodes in the subset from a given source node. We show that all of our results apply just as well to these multicast variations.

2 $(4+\varepsilon)$-Approximation using k Shortcut Edges

Let D be the diameter of the current graph G (without any shortcut edges). The diameter of the optimum solution is a value $D' \leq D$. In fact, there exists a set of at most k edges S such that the diameter of graph $G = (V, E \cup S)$ is D'. We want to find and add k shortcut edges, such that the new graph after adding our edges has diameter at most a constant times D'.

At first, we estimate the value of D' with an iterative process as follows. We need to find a lower bound and an upper bound for D'. We know that D' is not more than D. So D can be seen as an upper bound. Let a be the minimum length of the edges in G. We know that D' can not be less than $\min\{a, \delta\}$ because every path between any pair of vertices should use at least either one of the current edges in G or one of the shortcut edges. We can also assume that a is not zero otherwise we can contract the zero edges, and solve the problem for the graph after these contractions. If δ is also nonzero, we can use $\min\{a, \delta\}$ as a nonzero lower bound for D'. Otherwise if k is at least $n-1$, we can add $n-1$ shortcut zero edges to build a spanning tree with only zero edges. This way, the diameter of graph would be zero. So the problem is solved. Otherwise we can not reduce the diameter to $\delta = 0$, and the distance between some pairs of vertices would be at least a after adding $k < n-1$ shortcut edges. So in all cases, we can find a positive lower bound for D' which is either $\min\{a, \delta\}$ or a. Define L to be this lower bound. So we can assume that $0 < L \leq D' \leq D$.

Choose an arbitrary small $\varepsilon > 0$. There exists an $0 \le i \le \log_{1+\varepsilon}(D/L)$ such that $D/(1+\varepsilon)^{(i+1)} < D' \le D/(1+\varepsilon)^i$.

Clustering algorithm. This algorithm receives an input parameter $x \ge 0$. We partition the vertices of our graph into clusters of diameter at most $2x$ as follows. At first we pick a subset of vertices S as the centers of our clusters. This set should satisfy the following properties. The distance between any pair of vertices of set S in graph G should be greater than $2x - \delta$. For every vertex $u \notin S$, there should be a vertex v in S whose distance to v is at most $2x - \delta$. We find a set S with above properties as follows. Choose an arbitrary vertex from G like v and put it in S. While there exists a vertex like u outside S whose distance to every vertex in S is greater than $2x - \delta$, we add u to S. Clearly this iterative process finishes in at most n iterations because there are n vertices in G. For every vertex u outside S, there exists a vertex v in S such that $\text{dist}(u, v)$ is at most $2x - \delta$ where $\text{dist}(u, v)$ is the distance between u and v. Otherwise we would add u to S. Let k' be $|S|$, and $v_1, v_2, \ldots, v_{k'}$ be the vertices in S.

If we add $k' - 1$ shortcut edges from v_1 to all other center vertices in set S $(v_2, v_3, \ldots, v_{k'})$, the diameter of the new graph is at most $2[2x - \delta] + 2\delta = 4x$. Consider two vertices u and w in the new graph. There are two vertices in v_i and v_j in S such that $\text{dist}(v_i, u) \le 2x - \delta$ and $\text{dist}(v_j, w) \le 2x - \delta$. We also know that the distance between v_i and v_j is at most 2δ in the new graph because they are both connected to v_1 using two shortcut edges. We conclude that we can reduce the diameter of G to a value at most $4x$ using $k' - 1$ edges. Following we show how to use this clustering algorithm to solve our problem. Note that D' is the diameter of the optimum solution. We show that without using at least $k' - 1$ edges, the diameter of graph can not be reduced to x or less. So the number of edges used in the optimum solution, which is k, should be at least $k' - 1$ if D' is at most x.

Lemma 1. *If D' is at most x, the number of edges used in the optimum solution is at least $k' - 1$, and therefore k is at least $k' - 1$.*

Proof. Assume that there are less than $k' - 1$ edges used in the optimum solution. Let G' be the new graph after addition of shortcut edges of the optimum solution. Consider a shortest path tree T from vertex v_1 in G'. The shortest path tree T is a tree that contains a shortest path from v_1 to every other vertex of graph G', i.e., the result of the Dijkstra's algorithm in graph G' from source v_1. Note that the distance between v_1 and v_i in G is greater than $2x - \delta$ because they are both in S. Their distance in G' is at most D'. Note that $x \ge D'$, and δ can not be greater than D' otherwise addition of some edges with length δ does not reduce the diameter of the graph to some value less than D'. So $2x - \delta$ is at least D', and therefore the distance between v_i and v_1 is reduced during addition of the new edges. So there exists at least one shortcut edge in the new shortest path from v_i to v_1. Let e_i be the first shortcut edge in the shortest path from v_i to v_1 in T'. For each $2 \le i \le k'$, we have a shortcut edge e_i. But there are less than $k' - 1$ shortcut edges in the whole graph. So some of these edges must be the same.

Suppose e_i and e_j are the same edge (u', w'). Let P_i and P_j be the shortest paths from v_i and v_j to v_1 that both use the edge (u', w'). Without loss of generality, assume that the distance from u' to v_1 is less than the distance from w' to v_1 in graph G'. So both paths P_i and P_j should use this edge in direction u' to w'. Let Q_i and Q_j be the first parts of the paths P_i and P_j that connects v_i and v_j to u' respectively. So Q_i and Q_j are two paths that do not use any shortcut edge (note that we picked the first shortcut edge in each path). Because the length of P_i and P_j are both at most D', and they both use at least one shortcut edge, the lengths of paths Q_i and Q_j are at most $D' - \delta$. Because Q_i and Q_j are two paths from v_i and v_j to the same destination u', and they do not use any shortcut edge, the length of the shortest path between v_i and v_j in graph G is at most the sum of the lengths of Q_i and Q_j which is $2(D' - \delta) \leq 2x - 2\delta$. This is a contradiction because $\mathrm{dist}(v_i, v_j)$ should be greater than $2x - \delta$. This contradiction shows that the number of shortcut edges in the optimum solution is at least $k' - 1$. □

Now we can use the clustering algorithm iteratively to find a $(4+\varepsilon)$-approximation algorithm. Recall that an instance of the problem consists of a graph G and two parameters k and δ. We should use k shortcut edges of length δ to minimize the diameter of the graph.

Theorem 1. *For any $\varepsilon' > 0$, there exists a polynomial-time $(4 + \varepsilon')$-approximation algorithm that uses at most k shortcut edges.*

Proof. We choose an arbitrary small $\varepsilon > 0$. There exists an $0 \leq i \leq \log_{1+\varepsilon}(D/L)$ such that $D/(1+\varepsilon)^{(i+1)} < D' \leq D/(1+\varepsilon)^i$. We can run the clustering algorithm with $x = D/(1+\varepsilon)^j$ for different values of $0 \leq j \leq \log_{1+\varepsilon}(D/L)$, so for one of these values, the above inequality holds, and we can estimate D' with multiplicative error ε. If the number of clusters k' is at most $k+1$, we can find a solution with diameter $4x$ by adding $k' - 1 \leq k$ edges. If the number of clusters is more than $k+1$, using Lemma 1, we know that $D' > x$.

Let $x = D/(1+\varepsilon)^{j'}$ be the smallest value of x for which the number of clusters is at most x. We can find a solution with diameter $4x = 4D/(1+\varepsilon)^{j'}$ in this case. On the other hand, we know that the number of clusters for $x = D/(1+\varepsilon)^{j'+1}$ is more than $k+1$, so $D/(1+\varepsilon)^{j'+1}$ is less than D'. We conclude that the diameter of our solution is at most $4D/(1+\varepsilon)^{j'} \leq 4(1+\varepsilon)D'$. By choosing $\varepsilon = \varepsilon'/4$, we find a $(4+\varepsilon')$-approximation. □

3 Improving the Approximation Ratio using $2k$ Edges

In this section, we show how to use the clustering algorithm to find a solution with at most $2k$ additional edges having diameter at most $(2+\varepsilon)D' + 2\delta$, where D' is the diameter of the optimum solution using k additional edges.

We change our clustering algorithm slightly as follows. We pick the vertices of S such that their distance is greater than $2x$ instead of $2x - \delta$. Like before, we iteratively add a vertex u to S if its distance to all vertices in S is more than

$2x$. We stop when we can not insert anything to S. Again we run the clustering algorithm with $x = D/(1+\varepsilon)^j$ for different values of $0 \le j \le \log_{1+\varepsilon}(D/L)$. We prove that this time the number of clusters is not more than $2k$ when x is at least $D'/2$ as follows.

Lemma 2. *If x is at least $D'/2$, the number of clusters in our algorithm is not more than $2k+1$.*

Proof. Here we show why our algorithm acts like a clustering algorithm. Let $v_1, v_2, \ldots, v_{k'}$ be the k' centers we pick in our algorithm. Partition the vertices of graph G into k' clusters as follows. Put $v_1, v_2, \ldots, v_{k'}$ in clusters $C_1, C_2, \ldots, C_{k'}$ respectively. For every other vertex u, find the center v_i $(1 \le i \le k')$ with minimum distance $\mathrm{dist}(v_i, u)$, and put u in cluster C_i (remember $\mathrm{dist}(v_i, u)$ is the distance between v_i and u in graph G). This distance is at most $2x$ for every vertex u by definition of our algorithm (otherwise we could add u to S in the clustering algorithm). The distance between each pair of the k' centers is greater than $2x$. We conclude that all vertices in graph G whose distance to v_i is at most x are in cluster C_i for each $1 \le i \le k'$. We can prove this by contradiction. Assume vertex u whose distance to v_i is at most x in in another cluster C_j. It means that center v_j is the closest center to u so the distance between u and v_j is also at most x. Therefore vertex u has distance at most x from both centers v_i and v_j. So the distance between two centers v_i and v_j is at most $x + x = 2x$ which is a contradiction. We conclude that each cluster contains the vertices around its own center with radius x (and probably some other vertices as well).

Now consider the optimum solution that uses at most k edges. If the number of clusters k' is at most $2k+1$, the claim is proved. Otherwise there exists at least two clusters like C_i and C_j such that the additional edges are not incident to the vertices of $C_i \cup C_j$. Because every additional edge has two endpoints, therefore it can be incident to at most two clusters. So the number of clusters that are incident to some additional edges is not more than twice the number of additional edges. Because we have at least $2k+2$ clusters, there exists two clusters like C_i and C_j whose vertices are not incident to any additional edge.

Let G' be the new graph after adding additional edges in the optimum solution. The distance between v_i and v_j should be at most D' in G'. Their distance is greater than $2x \ge D'$ in graph G. This means that the shortest path between v_i and v_j in G' should use at least one of the additional edges. Let P be this shortest path between v_i and v_j in G'. Suppose $v_i = u_1, u_2, u_3, \ldots, u_l = v_j$ are the vertices of this path. Assume that edge (u_a, u_{a+1}) is the first additional edge in this path, and edge (u_b, u_{b+1}) is the last additional edge in this path where $a \le b$. Let P_1 be the first part of the path P before the first additional edge, i.e., $P_1 = (v_i = u_1, u_2, \ldots, u_a)$. And let P_2 be the last part of the path P after the last additional edge, i.e., $P_2 = (u_{b+1}, u_{b+2}, \ldots, u_l = v_j)$.

The paths P_1 and P_2 are two shortest paths in graph G' and G between pairs of vertices (v_i, u_a) and (u_{b+1}, v_j). Vertices u_a and u_{b+1} are outside clusters C_i and C_j because these two clusters are not incident to any additional edge, but vertices u_a and u_{b+1} are both incident to some additional edges. So the distance between u_a and v_i is greater than x otherwise u_a would be in cluster C_i (as proved above). So the length of path P_1 is more than x. With the same proof, the length of path P_2 is also more than x. So the length of P is greater than $x + x + \delta \geq 2x \geq D'$. This is a contradiction because the distance between v_i and v_j should be at most D' in the new graph G'. This shows that the number of clusters k' is at most $2k + 1$. $\square$

Now if we add $k' - 1$ from v_1 to all other centers, the diameter of the new graph would be at most $2x+2\delta$. If we choose x such that $D'/2 \leq x \leq D'(1+\varepsilon)/2$, the number of additional edges in our solution would not be more than $2k$, and the diameter of our graph would be at most $(2+\varepsilon)$ times the optimum solution.

Theorem 2. *There is a polynomial-time algorithm which finds a solution with diameter at most $(2 + \varepsilon)D' + 2\delta$ using at most $2k$ edges.*

Proof. Similar to our previous approach, we know that there exists an $0 \leq i \leq \log_{1+\varepsilon}(D/L)$ such that $D/(1 + \varepsilon)^{(i+1)} < D' \leq D/(1 + \varepsilon)^i$. We can run our clustering algorithm with $x = D/(1 + \varepsilon)^j$ for different values of $0 \leq j \leq \log_{1+\varepsilon}(D/L)$. This way we can find x such that $D'/2 \leq x \leq D'(1 + \varepsilon)/2$ in one of our runs. In that run, we find the desired solution. $\square$

4 Single-Source Version

In this section, we study the problem of adding k shortcut edges in order to minimize the maximum distance of all vertices from a given source s. Let $e(s)$ denote the maximum distance of vertex s from all other vertices, known as the *eccentricity* of vertex s.

At first we show that constant-factor approximations for the diameter-minimization problem can be converted to constant-factor approximations for the single-source version.

Lemma 3. *If there exists an α-approximation for diameter minimization problem, then there also exists a 2α-approximation for the single-source version.*

Proof. Consider a graph G. Let D' be the minimum possible diameter of G after adding k shortcut edges. Let r_s be the minimum possible $e(s)$ after adding k shortcut edges. We show that $D'/2 \leq r_s \leq D'$. If we add the k edges to reduce the diameter of G to D', the eccentricity of s would be also at most D'. So r_s can not be greater than D'. On the other hand, if we add the k edges such that the distance of every vertex to s is at most r_s, the distance between any pair of vertices can not be greater than $r_s + r_s = 2r_s$ using the triangle inequality. So the diameter of this graph is at most $2r_s$. We conclude that $D' \leq 2r_s$.

So if we use the α-approximation to minimize diameter, we get a graph with diameter at most $\alpha D'$. The eccentricity of s is also at most $\alpha D'$ which is at most $2\alpha r_s$. So using the same algorithm we get a 2α-approximation for the single-source version of the problem. $\square$

In the remainder of this section, we show how to obtain a $(1+\varepsilon)$-approximation algorithms using more additional edges. We use a linear-programming formulation similar to the linear-programming formulations of facility location and k-median problems. Then we show how the single-source version of the problem can solve the set-cover problem. We conclude that any $(\frac{3}{2}-\varepsilon)$-approximation for the single-source version should use at least $\Omega(k \log n)$ edges. This shows the optimality of our linear-programming algorithm.

We need the following lemma in our algorithm.

Lemma 4. *There exists an optimal solution for the single-source version in which all k shortcut edges are incident to the given source s.*

Proof. The proof is similar to the proof of Lemma 1 in [11]. □

First we solve a decision problem using linear programming. The decision problem is the following: can we add k shortcut edges to reduce the eccentricity of vertex s to some value at most x where x is a given value in the input graph? In the other words, is r_s at most x? If r_s is at most x, then the following linear program has a feasible integral solution.

Let $v_1, v_2, \ldots, v_n$ be the vertices of the input graph G. Without loss of generality assume that s is v_1. For every vertex v_i put a variable y_i in the linear program where $2 \leq i \leq n$. For every pair of vertices v_i and v_j put a variable $x_{i,j}$ where $2 \leq i, j \leq n$. All these variables are between 0 and 1. If we assume the integer programming version of this linear program. The variable y_i is equal to 1 if there is a shortcut edge from v_i to $s = v_1$, and it is zero when there is no such an edge. We add the constraint $\sum_{i=2}^{n} y_i \leq k$ because we know that the number of shortcut edges is at most k in the optimum solution. Variable $x_{i,j}$ is equal to 1 if the shortest path from vertex v_j to s in the optimum solution uses the shortcut edge v_i to v_1. If $x_{i,j}$ is 1, there should be an edge from v_i to s. So we add the constraint $x_{i,j} \geq y_i$ for any $2 \leq i, j \leq n$. On the other hand, vertex v_j can not use shortcut edge (v_i, s) if the distance between v_j and v_i is more than $x - \delta$. So we define the variable $x_{i,j}$ only for a pair of vertices (v_i, v_j) such that $\mathrm{dist}(v_i, v_j)$ is at most $x - \delta$. If the distance between s and v_j is at most x, the vertex v_j does not need to use any shortcut edge to reach s. But if $\mathrm{dist}(v_j, s)$ is greater than x, it has to use one of this edges, so we have

$$\mathrm{dist}(v_j, s) > x : \sum_{i:\mathrm{dist}(v_i,v_j) \leq x-\delta} x_{i,j} = 1 \qquad \text{for } 1 \leq j \leq n.$$

Here is the whole formulation of our linear program, or more precisely, our linear feasibility problem, as we do not want to minimize or maximize anything.

$$\begin{aligned} \sum_{i=2}^{n} y_i &\leq k, & \\ x_{i,j} &\leq y_i & \text{for } 2 \leq i, j \leq n, \\ \sum_{i:\mathrm{dist}(v_i,v_j) \leq x-\delta} x_{i,j} &= 1 & \text{for } 1 \leq j \leq n \text{ with } \mathrm{dist}(v_j, s) > x. \end{aligned}$$

As described above, if x is at least the optimum solution r_s, then the above linear program has a feasible solution. Next we show how to solve our problem using this linear program, and then show how to find the best x.

Lemma 5. *If there exists a feasible solution to the above linear program, then there exists a polynomial-time algorithm that finds $O(k \log n)$ shortcut edges whose addition reduces the eccentricity of vertex s to some value at most x.*

Proof. The algorithm proceeds as follows. Solve the linear program and find a feasible solution. While there exists some vertex in graph whose distance to s is greater than x, do the following. Add a shortcut edge (v_i, s) to the graph with probability y_i for each $2 \le i \le n$. After adding these edges, if there still exist a set of vertices whose distance to s is greater than x, do the same. Iteratively add edges to the graph until the distances of all vertices to s are at most x.

When this algorithm stops, the eccentricity of s is at most x. Now we show that the algorithm does not add a lot of edges compared to the optimum solution. The expected number of edges we add in each phase is $\sum_{i=2}^{n} y_i$, which is at most k. So we do not add more than k edges in each iteration in expectation. We now prove that the number of iterations is at most $O(\log n)$ with high probability (probability $1 - 1/n^c$ for arbitrary constant c).

Consider a vertex v_j whose distance to s is greater than x. Let $\{v_{a_1}, v_{a_2}, \ldots, v_{a_l}\}$ be the set of vertices whose distance to v_j is at most $x - \delta$. We know that $\sum_{i=1}^{l} x_{a_i,j}$ is equal to 1. On the other hand, we have that y_{a_i} is at least $x_{a_i,j}$. So $\sum_{i=1}^{l} y_{a_i}$ is at least 1. If we add a shortcut edge from one of these l vertices to s, the distance of v_j to s reduces to at most x. So if its distance to s is still greater than x after one iteration, it means that we did not add any of these edges in this iteration.

The probability of the event that we do not add any edge from these k edges to s in one iteration is $\prod_{i=1}^{k}(1 - y_{a_i})$ which is at most $[1 - (\sum_{i=1}^{l} y_{a_i}/l)]^l$. Because $\sum_{i=1}^{l} y_{a_i}$ is at least 1. This probability is at most $(1 - 1/l)^l$ which is at most $1/e$. So the probability of not choosing any of these edges in p iterations is at most $1/e^p$. If we do this iterative process for $(c+1)\ln(n)$ times, the distance of v_j to s is greater than x with probability at most $1/e^{(c+1)\ln(n)} = 1/n^{(c+1)}$. Using the union bound, we can prove that there exists a vertex with distance greater than x from s with probability at most $n \cdot (1/n^{(c+1)}) = 1/n^c$. So with high probability (at least $1 - 1/n^c$) all distances from s are at most x, and the algorithm stops after $(c+1)\ln(n)$ iterations. □

Theorem 3. *For any $\varepsilon > 0$, there exists a polynomial-time algorithm that adds $O(k \log n)$ edges to reduce the eccentricity of s to at most $1+\varepsilon$ times the optimum eccentricity r_s for k shortcut edges.*

Proof. Let r be the eccentricity of the source in the input graph G. The optimum eccentricity r_s is at most r. We also know that r_s can not be less than δ. So r_s is in range $[\delta, r]$. Run the above algorithm for $x = r/(1+\varepsilon)^i$ for $0 \le i \le \log_{(1+\varepsilon)}(r/\delta)$. Consider the smallest x for which the linear program has a feasible solution, and return the result of the above algorithm in that case. There exists a j for which r_s is in range $[r/(1+\varepsilon)^{j+1}, r/(1+\varepsilon)^j]$. The linear program has a feasible solution for $x = r/(1+\varepsilon)^j$ because $r/(1+\varepsilon)^j$ is at least r_s. So we can find a solution with eccentricity x which is at most $(1+\varepsilon)r_s$ using $O(k \log n)$ shortcut edges. $\square$

Now we reduce the set-cover problem to our single-source problem in order to show that using $o(k \log n)$ shortcut edges, we can not get an approximation ratio of better than $\frac{3}{2}$.

Theorem 4. *Any polynomial-time $(\frac{3}{2} - \varepsilon)$-approximation algorithm for the single-source version of our problem needs $\Omega(k \log n)$ shortcut edges assuming* $\mathrm{P} \neq \mathrm{NP}$.

Proof. Consider an instance of set cover. There are m sets $S_1, S_2, \ldots, S_m$, and we want to find the smallest collection of these sets whose union is equal to the union of all these m sets. Suppose there are n items in the union of these m sets. Construct a graph as follows. Let $v_1, v_2, \ldots, v_n$ be the items. Put a vertex for each of these n items. For each set S_j, put a vertex u_j. Connect u_j and v_i for each $1 \leq i \leq n$, and $1 \leq j \leq m$ if item v_i is in set S_j. Add two other vertices s and s'. Vertex s is the source of our instance, and is only connected to s'. The vertex s' has also m edges to all vertices $u_1, u_2, \ldots, u_m$. Set the length of all edges, and δ to be equal to 1. Now the problem is to add k shortcut edges in order to minimize the eccentricity of source s in this graph.

The eccentricity of s is now equal to 3. We prove that there exists a set of k shortcut edges whose addition reduces the eccentricity of s to at most 2 if and only if there is a solution for set-cover instance with size at most k. Assume that the set-cover instance has a solution with size at most k. We can add k edges from s to the vertices associated with these k sets (k subsets in the solution of the set-cover instance). This way the eccentricity of s would be at most 2. We now prove that if we can reduce the eccentricity of s using only k edges, the set-cover instance has a solution of size k. Note that the only vertices whose distance to s is more than 2 are $v_1, v_2, \ldots, v_n$. Using Lemma 4, we know that the k edges are all incident to s. We prove that there is an optimal solution in which all edges are between s and the vertices $u_1, u_2, \ldots, u_m$. Assume that there is an additional edge from s to v_i. Any path from s to v_j ($j \neq i$) that uses this edge, has length at least 3. So the only usage of this edge is reducing the distance of v_i to s from 3 to 1. There exists a vertex u_l such that there is an edge from v_i to u_l (the item v_i is in some set S_l). We can add an edge from u_l to s instead of an edge from v_i to s. The eccentricity of s would be still at most 2. So we can assume that all edges are from s to vertices $u_1, u_2, \ldots, u_m$. The distances of all vertices $v_1, v_2, \ldots, v_n$ are also at most 2. So for each v_i there exists an additional edge (s, u_l) such that item v_i is in set S_l. So the k additional edges form a solution of size k for the set-cover instance.

Assume that there exists a $(\frac{3}{2} - \varepsilon)$-approximation algorithm A for the single-source problem. For any k, if there is a solution of size at most k for the set-cover instance, the eccentricity of s in the optimum solution of the single-source instance is at most 2. So the eccentricity of s in the solution of algorithm A can not be 3 or more. Because algorithm A is a $(\frac{3}{2} - \varepsilon)$-approximation, so it has to find a solution with eccentricity at most $2 \cdot (\frac{3}{2} - \varepsilon) < 3$. The fact that the eccentricity can only take integer values shows that the result of algorithm A also has eccentricity at most 2. The result of algorithm A can be converted to a solution for the set-cover instance. We also know that there is no $o(\log n)$-approximation for set-cover problem [7], so algorithm A uses at least $O(k \log n)$ in its solution. This completes the proof. □

5 Multicast Version

In this section we show that our results and techniques are all applicable in the multicast version of the problem in which we just care about a subset of the nodes. Formally we are given an undirected weighted graph $G = (V, E, \ell)$, and a subset of vertices $V' \subseteq V$. We want to add k shortcut edges (of a fixed given length δ) in order to minimize the maximum distance between the nodes in set V'. In previous parts, we showed how to solve this problem when V' is equal to V. In the single-source version of the multicast problem, we are also given a source node s, and we want to minimize the maximum distance of nodes in V' from source s.

For our clustering algorithm, we just need to pick centers from vertex set V'. So we do not select any vertex outside V' as a center in our algorithm. We stop when we can not select any vertex of set V'. We get the same approximation ratio by this method, and all proofs and claims work similarly in this case as well.

For the linear-programming approach, we have to write the constraint $\sum_{i:\mathrm{dist}(v_i,v_j)\le x-\delta} x_{i,j} = 1$ for vertex v_j if v_j is in set V', and its distance from s is greater than x, i.e., $\mathrm{dist}(v_j, s) > x$. Because the distances of vertices outside V' from s are not important for us. Again all claims can be proved in the same way in this case as well. To make it more clear, the new linear-programming formulation is the following:

$$\begin{aligned}
\sum_{i=2}^{n} y_i &\le k, \\
x_{i,j} &\le y_i \qquad \text{for } 2 \le i, j \le n, \\
\sum_{i:\mathrm{dist}(v_i,v_j)\le x-\delta} x_{i,j} &= 1 \qquad \text{for } j \in V' \text{ with } \mathrm{dist}(v_j, s) > x.
\end{aligned}$$

References

1. Susanne Albers, Stefan Eilts, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. On Nash equilibria for a network creation game. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*. Miami, FL, pages 89–98, 2006.
2. Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70-78, 2002.

3. Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.
4. Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, and Morteza Zadimoghaddam. The price of anarchy in cooperative network creation games. *SIGecom Exchanges* 8(2), December 2009.
5. Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, and Morteza Zadimoghaddam The price of anarchy in network creation games. In *Proceedings of the 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 292–298, 2007. To appear in *ACM Transactions on Algorithms.*
6. Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*. Boston, MA, pages 347–351, 2003.
7. Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.
8. Jon Kleinberg. Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems* 14:431–438, 2001.
9. Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.
10. Nikolaos Laoutaris, Laura Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. Bounded budget connection (BBC) games or how to make friends and influence people, on a budget. In *Proceedings of the 27th ACM Symposium on Principles of Distributed Computing*, pages 165–174, 2008.
11. Adam Meyerson and Brian Tagiku. Minimizing average shortest path distances via shortcut edge addition. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, vol. 5687 of Lecture Notes in Computer Science, pages 272–285, 2009.