

Survivable Network Design Problems in Wireless Networks

Debmalya Panigrahi*

Abstract

Survivable network design is an important suite of algorithmic problems where the goal is to select a minimum cost network subject to the constraint that some desired connectivity property has to be satisfied by the network. Traditionally, these problems have been studied in a model where individual edges (and sometimes nodes) have an associated cost. This model does not faithfully represent wireless networks, where the activation of an edge is dependent on the selection of parameter values at its endpoints, and the cost incurred is a function of these values. We present a realistic optimization model for the design of survivable wireless networks that generalizes various connectivity problems studied in the theory literature, e.g. node-weighted steiner network, power optimization, minimum connected dominating set, and in the networking literature, e.g. installation cost optimization, minimum broadcast tree. We obtain the following algorithmic results for our general model:

1. For $k = 1$ and 2 , we give $O(\log n)$ -approximation algorithms for both the vertex and edge connectivity versions of the k -connectivity problem. These results are tight (up to constants); we show that even for $k = 1$, it is NP-hard to obtain an approximation factor of $o(\log n)$.
2. For the minimum steiner network problem, we give a tight (up to constants) $O(\log n)$ -approximation algorithm.
3. We give a reduction from the k -edge connectivity problem to a more tractable degree-constrained problem. This involves proving new connectivity theorems that might be of independent interest. We apply this result to obtain new approximation algorithms in the power optimization and installation cost optimization applications.

1 Introduction

Survivable Network Design (SND) problems involve designing minimum cost networks subject to connectivity requirements. Traditionally, we have assumed a cost model where each edge has a fixed cost and can be *bought* independently; the cost of the network is the sum of costs of the edges bought. In some cases (called the *node-weighted* version), fixed node costs have been considered in addition to edge costs; often, this version of the problem turns out to be significantly harder than the corresponding edge-weighted version. While being reasonably accurate for wired networks, these models

are unsuitable for wireless networks. In wireless networks, one typically needs to choose the value of a parameter x_v from a domain of possible values D at each node $v \in V$ (V being the set of nodes); and the *activation* of an edge (u, v) depends on the parameters chosen at its endpoints, i.e. there is an activation function $f_{uv} : D \times D \rightarrow \{0, 1\}$ that takes as input the values of x_u and x_v and returns 1 iff the edge (u, v) is activated for the chosen values of x_u and x_v . For example, one might need to set power levels at the nodes, and depending on the power levels of the endpoints, an edge is either active or inactive; or, one may need to fix the heights of towers for mounting antennas at nodes, and whether an edge is active depends on whether there is line-of-sight between the antennas at its endpoints. The objective is to minimize the total cost $\sum_{v \in V} x_v$, while ensuring that the activated set of edges satisfies some connectivity requirement \mathcal{C} . We call such networks *activation networks*, and the generic SND problem for activation networks, therefore, has the following form:

$$\begin{array}{l} \text{minimize } \sum_{v \in V} x_v \text{ where } x_v \in D \forall v \in V \\ \text{s.t. the set of edges } (u, v) \text{ with } f_{uv}(x_u, x_v) = 1 \\ \text{satisfies connectivity requirement } \mathcal{C} \end{array}$$

Throughout this paper, we will only consider undirected edges, and assume that $f_{uv} = f_{vu}$. This is the most likely scenario in practical applications since edges need to be activated in both directions (for acks, etc even if traffic flows in only one direction).

Before proceeding further, let us give some examples of activation functions and the corresponding suite of problems defined by them:

- **Minimum Node-weighted Steiner Network** (see e.g. [22]) In the minimum node-weighted steiner network problem, vertices (nodes) and edges have weights and the goal is to construct a minimum cost forest connecting terminals in each of k *terminal sets* of vertices. (A well-known special case of this problem is the *minimum node-weighted steiner tree* problem, where there is a single terminal set.) To model this problem in our framework, we add a vertex on each edge whose cost equals that of the edge, and then have the following activation function for any edge (u, v) (u (resp., v) can either

*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. Email: debmalya@mit.edu. Part of this work was done when the author was an intern at Microsoft Research, Redmond, WA 98052.

be an original vertex or a new one, and has cost c_u (resp., c_v):

$$f_{uv}(x_u, x_v) = \begin{cases} 1, & \text{if } x_u \geq c_u \text{ and } x_v \geq c_v \\ 0, & \text{otherwise.} \end{cases}$$

Klein and Ravi [22] gave an $O(\log n)$ -approximation algorithm for this problem. (Throughout this paper, n denotes the number of vertices in V .) They also showed that this approximation factor is tight up to constants. We will match their approximation factor (up to constants) for our more general framework.

- **Installation Cost Optimization** (see e.g. [11, 32, 31]) The installation cost of a wireless network is dominated by the cost of building towers at the nodes for mounting antennas, which in turn is proportional to the height of the towers. An edge (u, v) is activated if the towers at its endpoints u and v are tall enough to overcome obstructions in the middle and establish line-of-sight between the antennas mounted on the towers; this is modeled as each edge (u, v) having a *threshold height requirement* τ_{uv} , and edge (u, v) is activated if the scaled height of the towers at its endpoints sum to at least τ_{uv} . Thus,

$$f_{uv}(x_u, x_v) = \begin{cases} 1, & \text{if } \alpha_{u,uv}x_u + \alpha_{v,uv}x_v \geq \tau_{uv} \\ 0, & \text{otherwise.} \end{cases}$$

This is the application which inspired this work; we will give several algorithms for the general problem, which are also tight for this specific application.

- **Minimum Broadcast Tree** (see e.g. [3, 21, 20, 26, 18, 10, 33] for related work) The *broadcast tree* of a network is the routing tree over which data is routed from a source s to all other nodes. In wireless networks, each broadcast at a node u successfully reaches another node v with some probability p_{uv} (which is symmetric since successful transmission involves successful reception of the ack as well); therefore the expected number of broadcasts at u until the message reaches v is $\rho_{uv} = 1/p_{uv}$. The *minimum broadcast tree* problem in such unreliable networks aims to find a tree rooted at the source that minimizes $\sum_{v \in I(T)} \max_{u \in N_v^{(T)}} \rho_{uv}$, where $I^{(T)}$ is the set of non-leaf vertices of T and $N_v^{(T)}$ are the set of children of v in T .¹ We can show that this problem is equivalent, up to a factor of two in the

approximation ratio, to the installation cost optimization problem where the connectivity requirement is that the activated edges must contain a spanning tree on V .

- **Power Optimization** (see e.g. [1, 4, 5, 6, 7, 8, 15, 16, 25, 27, 23, 24, 28, 30]) In this problem, each edge (u, v) has a *threshold power requirement* θ_{uv} and edge (u, v) is activated if the power at each of its endpoints is at least this threshold. Thus,

$$f_{uv}(x_u, x_v) = \begin{cases} 1, & \text{if } \min(x_u, x_v) \geq \theta_{uv} \\ 0, & \text{otherwise.} \end{cases}$$

Typically, this variant admits better approximation than the general problem (i.e. the lower bounds break down). However, we will give a new result in this setting for the minimum cost k -edge connected subgraph problem.

Finite domain. In practice, the set of possible parameter values is often a small, discrete finite set. Therefore, we consider finite, discrete domains D , and allow our algorithms to run in time polynomial in $|D|$ (so e.g., activation functions are lookup tables). For the problems described above, this is without loss of generality. In node-weighted steiner network, we can restrict D to the costs on the vertices and edges without changing the optimum. In power optimization, restricting D to the thresholds on the edges does not change the optimal solution. In installation cost optimization, it can be shown that the same restriction increases the optimal cost by at most a factor of two; as we show later, this problem is NP-hard to approximate to a factor of $o(\log n)$ and hence this transformation is without significant loss in the approximation factor.

Monotonicity. We will assume throughout that the activation functions are *monotonic*, i.e. $f_{uv}(a_1, b_1) = 1$ implies $f_{uv}(a_2, b_2) = 1$ for any $a_2 \geq a_1, b_2 \geq b_1$. This is indeed the case in all the above applications, and in any other realistic scenario.

In this paper, we consider three fundamental connectivity requirements \mathcal{C} :

- **Minimum Edge-connected Activation Network (MEAN)** Each pair of vertices must have k edge-disjoint paths in the activated subgraph, for some given $k > 0$.
- **Minimum Vertex-connected Activation Network (MVAN)** Each pair of vertices must have k vertex-disjoint paths in the activated subgraph, for some given $k > 0$.

¹This definition makes the assumption that due to interference, no two nodes can broadcast simultaneously.

- **Minimum Steiner Activation Network (MSAN)** Each pair of vertices in each of k terminal sets $R_1, R_2, \dots, R_k \subseteq V$ must be connected by a path in the activated subgraph.

Besides being theoretically important, these connectivity requirements are also practically relevant since they ensure robustness of the designed network against edge and node failures. We will also consider some other connectivity requirements:

- **Minimum Spanning Activation Tree (MSpAT).** The activated set of edges must contain a spanning tree on the vertices. This is a special case of all the three connectivity requirements described above ($k = 1$ for MEAN and MVAN, and only one terminal set $R_1 = V$ for MSAN).
- **Minimum Degree-constrained Activation Network (MDAN).** For every vertex subset U in a given partition² of vertices \mathcal{P} , there must be at least r_U activated edges with exactly one endpoint in U , where r_U is the (given) *requirement* of U .
- **Minimum Activation Flow (MAF).** The activated set of edges must contain k edge-disjoint paths between two specified vertices s, t . For the special case of $k = 1$, we call this the **Minimum Activation Path (MAP)** problem. Note that the MAP problem is also a special case of the MSAN problem with only one terminal set $R_1 = \{s, t\}$.

1.1 Our Results. We first outline the main results presented in this paper:

- We show that it is NP-hard to approximate the MSpAT problem to a factor of $o(\log n)$, even for the installation cost setting. Since this is a special case of the MVAN, MEAN and MSAN problems, it is also NP-hard to approximate these problems to a factor of $o(\log n)$.
- We give an $O(\log n)$ -approximation algorithm for the MSpAT problem. This generalizes an $O(\log n)$ -approximation algorithm for the installation cost setting [31]; for power optimization, the problem is much more tractable, and a $5/3$ -approximation algorithm is known [1].

²A partition of a set S is a collection of subsets \mathcal{T} such that (a) $T_1 \cap T_2 = \emptyset$ for any pair of subsets $T_1, T_2 \in \mathcal{T}$, and (b) $\cup_{T \in \mathcal{T}} T = S$.

- We give $O(\log n)$ -approximation algorithms for the MEAN and MVAN problems for $k = 2$. Previously, no result was known for installation cost optimization, but the problems have 4-approximation [5] and $11/3$ -approximation [24] algorithms respectively in the power optimization setting.
- We show that an α -approximation algorithm for the MDAN problem implies an $(O(\alpha \log n), 2)$ -approximation algorithm for the MEAN problem with arbitrary k . (The cost of the solution produced by our algorithm is at most $O(\alpha \log n)$ times that of the optimal, but the connectivity guarantee in the solution is that there are at least $\lceil k/2 \rceil$ edge-disjoint paths between every pair of vertices in the activated subgraph.) As applications of this result, we obtain $(O(\log^2 n), 2)$ -approximation algorithms for the MEAN problem for arbitrary k in the power optimization and installation cost optimization settings. We also show that we can improve the approximation factor for both problems to $(O(\log n \log k), 2)$ by using a more refined algorithm. The only relevant previous result that we are aware of is an $\min(O(\sqrt{n}), 2k)$ -approximation algorithm for power optimization [5, 16]; no previous result was known for installation cost optimization.
- We give an $O(\log n)$ -approximation algorithm for the MSAN problem. In power optimization, there is a 4-approximation algorithm for this problem [24]; no previous result is known for installation cost optimization.
- We give an exact algorithm for the MAP problem. However, we show that the MAF problem is at least as hard as the well-known ℓ -densest subgraph problem [12, 2].

1.2 Our Techniques. We now outline our techniques in obtaining the above results.

The MSpAT Problem. We show that this problem, even in the installation cost optimization setting, generalizes the Minimum Connected Dominating Set (MCDS) problem. It is NP-hard to approximate the MCDS problem to an $o(\log n)$ factor [14]. To complement this lower bound, we give a simple $O(\log n)$ -approximation greedy algorithm for this problem.

MEAN and MVAN problems for $k = 2$. For the special case of $k = 2$, we give $O(\log n)$ -approximation algorithms for the MEAN and MVAN problems. (Vertex connectivity of two is also known as *biconnectivity*.) Both algorithms have the same generic two-step structure:

- In the first step, we run the $O(\log n)$ -approximation algorithm for the MSpAT problem.
- In the second step, we give an $O(\log n)$ -approximation algorithm for augmenting the edge (resp., vertex) connectivity of the activated set of edges from one to two by adding new edges to this set.

Clearly, the two steps, in combination, give an $O(\log n)$ -approximation algorithm to the MEAN (resp., MVAN) problem. The difference between the two algorithms is in the second step—the algorithm for the MEAN and MVAN problems respectively depend on the edge and vertex connectivity structure of the activated subgraph. However, our basic technique is the same for both problems: we show that we can reduce both these augmentation problems (via greedy algorithms that lose a factor of $O(\log n)$) to the **Minimum Leaf-weighted Subtree (MLS)** problem which is defined as follows: *Given a node-weighted tree T rooted at node r , choose a subtree rooted at r that contains at least k edges for a given $k > 0$, and has the minimum sum of costs of the (non-root) leaves of the subtree.* We give an algorithm to solve the MLS problem exactly, thereby leading to $O(\log n)$ -approximation algorithms for the augmentation step of the MEAN and MVAN algorithms.

MEAN problem for arbitrary k . A previously used technique (in power optimization) for the MEAN problem for arbitrary k is to use the following two-step process:

- Use an approximation algorithm for the degree-constrained problem with \mathcal{P} being the partition of singleton vertices and $r_{\{v\}} = k$ for each vertex v , to obtain a partial solution for the minimum edge-connected problem.
- Augment the solution obtained in the previous step using the minimum edge-cost version of the edge-connected problem where the cost of each edge is its power threshold, to obtain the final solution.

Recall that by Menger’s theorem (see e.g. [9]), having k -edge disjoint paths between every pair of vertices is equivalent to k -edge connectivity, i.e. having at least k edges in every cut. Thus, the first step uses a weaker constraint than k -edge connectivity and hence the optimal solution for the first step is at most as expensive as the optimum for the overall problem. Further, since the connectivity requirements are local to each vertex for this step, it is typically much easier to design an approximation algorithm. However, in the second step, one loses a factor of $\Omega(\sqrt{n})$ in the approximation ratio

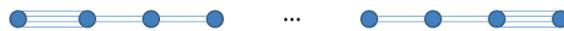


Figure 1: A graph with minimum degree 4 but $n - 2$ 4-edge-connected components

(in fact, in our more general problem, we can show that we will lose a factor of $\Omega(n)$ in the second step). To overcome this shortcoming, we show that we can use an α -approximation MDAN algorithm repeatedly in an iterative manner to obtain a MEAN algorithm.

Let us first consider the case $k = 1$. We repeatedly run the MDAN algorithm, where the partition in each iteration is the set of connected components in the subgraph activated by previous iterations, and the requirement $r_U = 1$ for each component. Since the number of components decreases by a factor of at least two in each step, we terminate after at most $\lg n$ iterations yielding an overall approximation factor of $O(\alpha \log n)$.

For the same approach to work for arbitrary values of k , we need to quantify the progress we make towards satisfying the k -edge connectivity constraint in each iteration of the MDAN algorithm. For this purpose, we need the following definition:

DEFINITION 1.1. *A k -edge connected component is a maximal subset of vertices such that every pair of vertices in the subset have k -edge-disjoint paths between them.*

Edge connectivity is transitive, i.e. if vertex pairs u, v and v, w are k -edge connected, then so too is u, w . This implies that the k -edge connected components form a partition of the vertex set. We can then view our goal as activating a set of edges so that this partition, which has n singleton components before any edge is activated, coalesces into a single k -edge connected component spanning all the vertices. Now, suppose any solution to the MDAN problem were to produce a set of at most cn k -edge connected components, where $c < 1$ is a constant. Let us define these components as the subsets of our partition \mathcal{P} in the next iteration of MDAN. Then, after the next iteration, we get at most c^2n k -edge connected components. Overall, this iterative algorithm where in each step, we call MDAN with the subsets of the partition defined as the k -edge connected components, will terminate in $O(\log n)$ iterations since the number of k -edge connected components decreases by a constant factor in each iteration. The overall approximation ratio of the algorithm would then be $O(\alpha \log n)$ for the MEAN problem. Unfortunately, as shown by an example in Figure 1, one can construct graphs where each vertex has degree at least k but there

are $n-2$ k -edge connected components. (So, the number of components does not decrease by a constant factor in each iteration.) However, we show, in the following theorem, that if we focus on $\lceil k/2 \rceil$ -edge connected components, we do have the desired property.

THEOREM 1.1. *The number of $\lceil k/2 \rceil$ -edge-connected components in a graph with minimum degree k is at most $3n/4$.*

This theorem establishes a relationship between local degree constraints and global edge connectivity properties—such a relationship is useful because often (as in our problem), it is much easier to satisfy a polynomial number of degree constraints rather than an exponential number of cut constraints. We hope that this theorem will find other applications.

The above theorem allows us to iteratively use the greedy MDAN algorithm, where the partition in an iteration is defined by the $k/2$ -edge³ connected components in the subgraph activated in previous iterations. This gives an approximation factor of $O(\alpha \log n, 2)$ overall for the MEAN problem for arbitrary k .

We exploit this connection between the MEAN and MDAN problems for the power optimization and installation cost optimization settings. For power optimization, we use the $O(\log n)$ -approximation algorithm for the degree-constrained version due to Kortsarz et al [23] to obtain an $(O(\log^2 n), 2)$ -approximation algorithm for the k -edge connected problem. For installation cost optimization, we design a new $O(\log n)$ -approximation algorithm for the degree-constrained version, which leads to an $(O(\log^2 n), 2)$ -approximation algorithm for the k -edge connected problem. We refine both these algorithms to improve their approximation ratios to $(O(\log n \log k), 2)$.

The MAP problem. We give a simple dynamic program that refines Bellman-Ford’s single-source shortest path algorithm [9] for solving this problem exactly.

The MAF problem. We show that this problem generalizes the **Minimum Node-weighted k -Flow (MNF)** problem which asks for the cheapest k edge-disjoint paths between two terminals s, t in a node-weighted undirected graph. Nutov [29] observed that the MNF problem is at least as hard as the ℓ -densest subgraph problem [12, 2].

The MSAN problem. We give an approximation-preserving reduction of the MSAN problem to the **Minimum Node-weighted Steiner Network (MNSN)**

problem which is defined as follows: *Given k sets of terminal nodes and a fixed cost for each non-terminal (i.e. steiner) node, find the minimum cost forest that connects all vertices in each terminal set.* This allows us to use the *spider decomposition* algorithm for MNSN due to Klein and Ravi [22] to obtain an $O(\log n)$ -approximation algorithm for the MSAN problem.

1.3 Roadmap. The proof of the lower bound for the MSpAT problem appears in section 2, while the $O(\log n)$ -approximation algorithm is presented in section 3.1. In section 3.2, we present our algorithm for the MVAN problem for $k = 2$; the corresponding algorithm for the MEAN problem appears in section 3.3. We establish the connection between the MDAN and MEAN problems, and use it to obtain algorithms for the MEAN problem with arbitrary k for the power optimization and installation cost optimization settings in section 3.4. We present the results on the MAP and MAF problems in sections 4.1 and 4.2 respectively. Finally, the reduction of the MSAN problem to the MNSN problem is presented in section 4.3.

2 Hardness of Approximation

In this section, we will show that it is NP-hard to approximate the MSpAT problem to a factor of $o(\log n)$ by giving a reduction of the minimum connected dominating set problem to the MSpAT problem for the installation cost setting.

Minimum Connected Dominating Set (MCDS). Given an undirected unweighted graph $G = (V, E)$, find a minimum-sized subset of vertices S such that (1) S is connected and (2) S is a *dominating set* (i.e. every vertex of V is either in S or has an edge connecting it to a vertex in S).

It is known that it is NP-hard to obtain an $o(\log n)$ -approximation algorithm for the MCDS problem [14].

Given an instance $G = (V, E)$ of the MCDS problem, we define an instance of the MSpAT problem on V as follows: *The domain $D = \{0, 1\}$ and for any pair of vertices u, v , if $(u, v) \in E$ then $f_{uv}(a, b) = 1$ iff $a+b \geq 1$, while if $(u, v) \notin E$, then $f_{uv}(a, b)$ is identically 0.* The following theorem establishes the validity of this reduction.

THEOREM 2.1. *If there is a connected dominating set S in G that contains c vertices, then there is a solution of cost c to the instance of the MSpAT problem. Conversely, if there is a solution to an instance of the MSpAT problem of cost c , then there is a connected dominating set S in G that contains at most $2c$ vertices.*

Proof. For the forward direction, observe that setting

³From now on, we will always use $k/2$ instead of $\lceil k/2 \rceil$ for brevity, with the understanding that our proofs hold for $\lceil k/2 \rceil$ as well.

$x_v = 1$ for all vertices in S activates a spanning subgraph of G . For the converse direction, note that we have chosen the activation function in a way that the cost of activating a set of edges is at least its *minimum vertex cover*.⁴ We can decompose a tree into set of node disjoint paths where each path contains at most one leaf vertex. Then, a vertex cover of the tree must include (disjoint) vertex covers of each of these paths, and a vertex cover of any of these paths must contain at least half the number of non-leaf vertices in the path. Thus, the number of non-leaf vertices in the optimal spanning tree in the MSpAT solution is at most $2c$, and these vertices form a connected dominating set of the graph.

3 Global Connectivity Problems

In this section, we will focus on problems where the connectivity requirement is *global*, i.e. for all vertices.

3.1 Minimum Spanning Activation Tree. First, we give an $O(\log n)$ -approximation algorithm for the MSpAT problem. To describe this algorithm, we need to define a *star*.

DEFINITION 3.1. *A star is a graph where at most one vertex has degree greater than one; this vertex is called the center while the other vertices are called peripheral vertices. (A single edge is also a star, but either vertex can be called its center.)*

Our algorithm runs in rounds, where in each round, we select a star with the minimum cost-benefit ratio to add to the set of activated edges. The *cost* of a star is the minimum sum of values of x_v at the center and peripheral vertices required to activate the edges of the star, while the *benefit* of a star is the decrease in the number of connected components in the activated subgraph when we activate the edges of the star.

We first show that each round runs in polynomial time. Since there are n vertices and $|D|$ possible values of x_v for any vertex v , it is sufficient to give an algorithm to find the optimal star (i.e. the one minimizing the cost-benefit ratio) among stars centered at a particular vertex v and having $x_v = a$ for some fixed $a \in D$. Define $x_u^{(v,a)}$ as the minimum value of x_u for which the edge (u, v) will be activated when $x_v = a$. For each vertex u , we can determine the value of $x_u^{(v,a)}$ in $O(\log |D|)$ time. Let $C(u)$ be the connected component containing vertex u . Then, for each component $C \neq C(v)$, we define a unique $m_C = \arg \min_{w \in C} (x_w^{(v,a)})$ breaking ties arbitrarily if required. Now, we define the reduced cost

(denoted by $c_{uv}^{(v,a)}$) of an edge (u, v) as the following:

$$c_{uv}^{(v,a)} = \begin{cases} x_u^{(v,a)} & \text{if } C(u) \neq C(v) \text{ and } u = m_{C(u)} \\ \infty & \text{otherwise} \end{cases}$$

We now order the edges incident on v in increasing order of reduced cost; let e_1, e_2, \dots be this order. The following lemma shows that the optimal star is a prefix of this order; therefore, our algorithm takes polynomial time.

LEMMA 3.1. *The optimal star centered at v with $x_v = a$ is a prefix of the increasing order of reduced cost of edges incident on v .*

Proof. Suppose the lemma were false. Consider an optimal solution; let e_i be the first edge in the increasing order that is not activated while some e_j ($j > i$) is activated. Replacing e_j by e_i in the solution does not increase the cost, and retains the same benefit. We can repeatedly perform this exchange until the lemma is satisfied.

We now prove the approximation ratio of the algorithm. Let \mathcal{P} be the partition of V formed by the connected components of (V, A) , where A is the set of activated edges. Let F be a *minimal* set of edges such that $(V, A \cup F)$ is a connected graph. By minimality, F must be a forest. Root each tree of F at an arbitrary vertex, and let \mathcal{S} be the set of stars formed by each non-leaf vertex with its children in these trees. Then, the next lemma is a consequence of the previous lemma.

LEMMA 3.2. *\mathcal{S} satisfies both the following properties:*

- *Each vertex in V appears in at most two stars in \mathcal{S} .*
- *The sum of benefits of the stars in \mathcal{S} is at least the number of vertex subsets in \mathcal{P} minus one.*

Proof. Each vertex appears in at most the stars centered at itself and its parent in the tree containing it in F . Since $A \cup F$ is a connected graph, the benefit of F , and therefore the sum of benefits of the individual stars, is at least the number of vertex subsets in \mathcal{P} minus one.

The next theorem follows using standard techniques from the above lemma.

THEOREM 3.1. *The greedy algorithm for the MSpAT problem described above has an approximation factor of $O(\log n)$.*

Proof. Let the marginal cost of activating an edge be the ratio of the total cost of activating a star containing

⁴A *vertex cover* is a set of vertices such that every edge is incident on at least one vertex.

the edge to the number of edges in the star. We now sequence the edges in chronological order of addition to the activated subgraph, ordering edges from the same star arbitrarily. The above lemma ensures that the marginal cost of the k th activated edge is at most $\frac{\text{OPT}}{n-k}$, where OPT is the cost of an optimal solution. Thus, the total cost of the algorithmic solution is at most

$$\left(\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{2} + 1\right) \text{OPT} = O(\log n)\text{OPT}.$$

3.2 Minimum Biconnected Activation Network. Now, we give an algorithm for the MVAN problem for $k = 2$. Our algorithm has two phases.

- In the first phase, we run the $O(\log n)$ -approximation algorithm for the MSpAT problem given in section 3.1. This produces a connected spanning activation subgraph whose cost is at most $O(\log n)$ times that of an optimal solution of the MVAN problem.
- In the second phase, we give an $O(\log n)$ -approximation algorithm for optimally augmenting the solution from the first phase to make the activated subgraph biconnected.

As outlined in the introduction, we show that we can reduce the problem in the second phase to the Minimum Leaf-weighted Subtree (MLS) problem where the goal is to select a rooted subtree containing at least k edges that minimizes the sum of weights of the leaves. We lose a factor of $O(\log n)$ in the approximation factor in this reduction. To describe this reduction, we need to define *block-cutpoint* graphs [17].

DEFINITION 3.2. A block of an undirected graph is a maximal biconnected subset of vertices, while a cutpoint is a vertex whose removal increases the number of components in the graph. A block-cutpoint graph T of an undirected graph G is a graph whose nodes are the blocks and cutpoints of G and edges connect each block to the cutpoints contained in it.

It turns out that the block-cutpoint graph of a connected graph is a tree such that removing a cutpoint v from G splits G into the vertex subsets corresponding to the subtrees of v in T . We define the *partition number* of a vertex v (denoted by $\rho(v)$) as the number of components the graph splits into when v is removed minus one. Equivalently, if v is a cutpoint, $\rho(v)$ is the degree of v in the block-cutpoint tree T minus one; if v is not a cutpoint, $\rho(v) = 0$. Overloading our notation, the partition number of graph G (denoted by $\rho(G)$) is the sum of the partition numbers of its vertices. We show the following property of $\rho(G)$ for connected graphs.

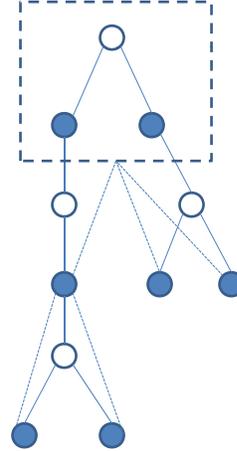


Figure 2: The rooted tree formed from the block-cutpoint tree. The bold circles represent blocks and the empty ones represent cutpoints. The edges of the block-cutpoint tree are shown in bold lines. The dashed box represents the block containing the root cutpoint v , and the dotted lines represent edges in the rooted tree.

LEMMA 3.3. If G is a connected graph, then its partition number $\rho(G)$ is at most $2n - 4$.

Proof. The partition number of any spanning tree of G containing ℓ leaves is $2(n - 1) - \ell$, which is at least $2n - 4$ since any spanning tree has at least two leaves. To complete the proof, note that adding edges does not increase the partition number of a graph.

Observe that the partition number of a biconnected graph is 0. So, the process of augmenting a connected graph into a biconnected one decreases the partition number from at most $2n - 4$ to 0. This motivates us to describe a greedy algorithm where, in each round, the goal is to obtain maximum decrease in $\rho(A)$ of the activated subgraph A while minimizing cost. To this end, in each round, we add a star s that minimizes the ratio c_s/b_s , where the cost c_s is the sum of x_v of the vertices in s required to activate all the edges in s , and the benefit b_s is the decrease in the partition number of the activated subgraph as a result of adding s to it.

We will reduce each round of this greedy algorithm to the MLS problem, and then give a polynomial time exact algorithm for the MLS problem. As earlier, we focus on finding the optimal star centered at a vertex v with $x_v = a$ for some $a \in D$. Since all edges in T are between a block and a cutpoint, and all the leaves are blocks, the tree T can be decomposed into a set of maximal stars, each of which is centered at a cutpoint

and has blocks as its peripheral vertices; the cutpoint at the center is contained in the peripheral blocks in G . We call each such star a *full component*. For the purpose of this reduction, if v is a cutpoint, we will consider the full component containing v as a single block that appears in all the full components that any block containing v appeared in. With this assumption, we root tree T at the unique block containing v . Each full component C now has a root block r_C . We replace each full component C in T with edges between r_C and the other blocks in C . Let S be the resulting rooted tree defined on the blocks (see Figure 2). Now, for each vertex $u \in V$ ($u \neq v$), define b_u as the unique block containing u if u is not a cutpoint, and as the root block of the full component that was centered at u in T if u is a cutpoint. Then, for every block b (other than the one containing v), we define $c_b = \min_{b_u=b} x_u^{(v,a)}$ and a unique $e_b = \arg \min_{b_u=b} x_u^{(v,a)}$ breaking ties arbitrarily if required. The following property ensures that we do not need to consider edges not in $E_b = \{e_b : b \text{ is a block not containing } v\}$ when finding the optimal star.

LEMMA 3.4. *There is an optimal star s centered at v and having $x_v = a$ that only contains edges from E_b .*

Proof. In a star, we can replace any edge not in E_b with the edge in E_b having its endpoint other than v in the same block as the previous edge; this does not increase the cost or change the benefit of the star.

For any subset of blocks B that does not include the block containing v , the cost of activating the subset of edges $E_B = \{e_b : b \in B\}$ with $x_v = a$ is $a + \sum_{b \in B} c_b$ and its benefit is the number of edges in the minimal rooted subtree in S containing all blocks in B . This yields the following algorithm for finding the optimal star centered at v with $x_v = a$ (here, β is the number of blocks in the transformed block-cutpoint tree): *For each $k = 1, 2, \dots, |S| - 1$, solve the MLS problem to obtain the minimum $\sum_{b \in B} c_b$ given that the benefit has to be at least k . Now, compare the solutions and take the one that has the best c_s/b_s ratio.* This completes the reduction.

Minimum Leaf-weighted Subtree (MLS). We now give an algorithm for exactly solving the MLS problem for parameter k on an arbitrary tree S with root r and cost c_v for vertex v . For every vertex, order its children arbitrarily. Then, consider the MLS subproblem for the subtree subtended at a vertex v , but only including the subtrees subtended at the first i children of v , with the constraint that the size of the selected subtree needs to be exactly j . We denote this subproblem by $\text{MLS}(v, i, j)$. We solve the $\text{MLS}(v, i, j)$ problem for all

$v \in V$ (in post-order), for all $1 \leq i \leq n_v$ where v has n_v children (in increasing order of i) and for all $1 \leq j \leq k$. Clearly, the overall MLS problem is identical to $\text{MLS}(r, n_r, k)$. Our dynamic program is given in Figure 3. To describe this dynamic program, let S_i denote the subtree subtended at u_i plus the edge (v, u_i) . In the non-trivial third case in Figure 3, we compare the following possibilities:

- The entire optimal subtree is contained S_1, S_2, \dots, S_{i-1} .
- The optimal subtree contains only the (v, u_i) edge from S_i ; the remaining edges are from S_1, S_2, \dots, S_{i-1} .
- The entire optimal subtree is contained in S_i .
- Exactly $1 \leq \ell \leq j - 2$ of the edges of the optimal subtree are from S_1, S_2, \dots, S_{i-1} while $j - \ell$ edges are from S_i .

Our base case are the leaf vertices v , for which $\text{MLS}(v, i, j) = \infty$ for all i, j . This dynamic program runs in polynomial time and solves the MLS problem exactly. This completes the description of each round of the greedy algorithm.

We now prove that the greedy algorithm described above has an approximation ratio of $O(\log n)$ for augmenting vertex connectivity from one to two in the activated subgraph. The following lemma is crucial.

LEMMA 3.5. *Let $G = (V, E)$ be any connected graph and F_1, F_2 be two sets of edges on V . Let b_1, b_2 and $b_{1,2}$ be the decrease in the partition number of G due to the addition of F_1, F_2 and $F_1 \cup F_2$ respectively. Then, $b_{1,2} \leq b_1 + b_2$.*

Proof. For any cutpoint v , let the vertex subsets that graph G decomposes into on removing v be called the *components* of v . Define a star graph G_v where v is the central vertex and each component of v is contracted into a single peripheral vertex. Then, the decrease in the partition number of v due to the addition of a set of edges F is equal to the number of edges in any spanning forest of F on graph G_v . Clearly, the sum of the number of edges in spanning forests of F_1 and F_2 on G_v is at least as much as the number of edges in a spanning forest for $F_1 \cup F_2$. Thus, the decrease in $\rho(v)$ due to F_1 and F_2 separately is at least as much as the decrease in $\rho(v)$ due to $F_1 \cup F_2$. To complete the proof, note that the decrease in $\rho(G)$ is the sum of decreases in $\rho(v)$ for the cutpoints v .

Let (V, A) be a connected graph, and let F be any *minimal* set of edges such that $(V, F \cup A)$ biconnected.

$$\text{MLS}(v, i, j) = \begin{cases} \min_{\ell=1}^i (c_{u_\ell}) & \text{if } j = 1 \\ \text{MLS}(u_1, n_{u_1}, j - 1) & \text{if } i = 1 \text{ and } j > 1 \\ \min(\text{MLS}(v, i - 1, j), \text{MLS}(v, i - 1, j - 1) + c_{u_i}, \text{MLS}(u_i, n_{u_i}, j)), & \\ \min_{\ell=1}^{j-2} (\text{MLS}(v, i - 1, \ell) + \text{MLS}(u_i, n_{u_i}, j - \ell - 1)) & \text{if } i > 1 \text{ and } j > 1 \end{cases}$$

Figure 3: Dynamic Program for the $\text{MLS}(v, i, j)$ problem for $1 \leq i \leq n_v$ where v has n_v children and $1 \leq j \leq k$. Here, u_1, u_2, \dots are the children of v in an arbitrary order.

Because of minimality, F must be a forest. We root each tree of this forest arbitrarily and decompose each tree into stars \mathcal{S} centered at each non-leaf vertex and containing its children as peripheral vertices. Then, the previous lemma ensures that \mathcal{S} has the following properties.

LEMMA 3.6. \mathcal{S} satisfies both the following properties:

- Each vertex in V appears in at most two stars in \mathcal{S} .
- The sum of benefits of the stars in \mathcal{S} is at least the partition number of (V, A) .

Proof. Each vertex appears in at most the stars centered at itself and its parent in the tree containing it in F . Since $A \cup F$ is a biconnected graph, the benefit of F , and therefore the sum of benefits of the individual stars by Lemma 3.5, is at least the partition number of A .

The next theorem follows from the above lemma using standard techniques (similar to the proof of Theorem 3.1).

THEOREM 3.2. *The algorithm presented above achieves an approximation guarantee of $O(\log n)$ for the second step of the MVAN algorithm.*

Combining all the pieces, we get the following theorem.

THEOREM 3.3. *The two-stage algorithm for the MVAN problem for $k = 2$ has an approximation factor of $O(\log n)$.*

3.3 Minimum 2-edge-connected Activation Network. We now give an algorithm for the MEAN problem for $k = 2$. As with the corresponding MVAN algorithm, this algorithm has two phases.

- In the first phase, we run the $O(\log n)$ -approximation algorithm for the MSpAT problem given in section 3.1. This produces a connected spanning activation graph whose cost is at most $O(\log n)$ times that of the optimal solution for the MEAN problem.

- In the second phase, we give an $O(\log n)$ -approximation algorithm for optimally augmenting the solution from the first phase to make the activated subgraph 2-edge-connected.

As with the MVAN algorithm, we show that we can reduce the problem in the second phase to the Minimum Leaf-weighted Subtree (MLS) problem where the goal is to select a rooted subtree containing at least k edges that minimizes the sum of weights of the leaves. We lose a factor of $O(\log n)$ in the approximation factor in this reduction. To describe this reduction, we need to use the following property of connected graphs.

LEMMA 3.7. *A connected graph induces a spanning tree on its 2-edge-connected components.*

Proof. Clearly, the induced graph on the 2-edge-connected components is a connected graph; if it contains cycles, that the maximality of the 2-edge-connected components is violated.

Observe that this tree is vacuous, i.e. contains no edges, for a 2-edge-connected graph. So, the process of augmenting a connected graph into a 2-edge-connected one decreases the number of edges in this tree from at most $n - 1$ to 0. Let us denote the number of edges in this tree for a connected graph G as $\eta(G)$. This motivates us to describe a greedy algorithm where, in any round, the goal is to obtain maximum decrease in $\eta(A)$ of the activated subgraph A while minimizing cost. To this end, in each round, we add a star s that minimizes the ratio c_s/b_s , where the cost c_s is the sum of x_v of the vertices in s required to activate all the edges in s , and the benefit b_s is the decrease in $\eta(A)$ as a result of adding s to A .

We will reduce each round of this greedy algorithm to the MLS problem. As earlier, we focus on finding the optimal star centered at a vertex v with $x_v = a$ for some $a \in D$. For any vertex u , let the 2-edge-connected component containing u be $C(u)$. Then, the benefit of a star s centered at v is the number of edges in a minimal subtree of the spanning tree described above that contains all the components $\{C : C = C(u), (u, v) \in s\}$, where the tree is rooted at

$C(v)$. For each 2-edge-connected component $C \neq C(v)$, we define $w_C = \min_{C(u)=C} x_u^{(v,a)}$ and a unique $e_C = \arg \min_{C(u)=C} x_u^{(v,a)}$ breaking ties arbitrarily if required. The following property is crucial.

LEMMA 3.8. *There is an optimal star s centered at v and having $x_v = a$ that only contains edges from $E_b = \{e_b : b \text{ is a block not containing } v\}$.*

Proof. For any star s , first discard all edges with both endpoints in $C(v)$. Then, for any 2-edge-connected component $C \in \{C : \exists(u,v) \in s \text{ s.t. } C(u) = C\}$, we replace the set of edges with the endpoint other than v in C by the edge e_C . The new star formed after the transformation has the same benefit and at most as much cost as s .

The cost of activating a star $s \subseteq \{e_C : C \neq C(v)\}$ is $a + \sum_{C(u):(u,v) \in s} w_{C(u)}$. This leads to the following algorithm for finding the optimal star centered at v with $x_v = a$ (here, t is the number of edges in the spanning tree): *For each $k = 1, 2, \dots, t$, solve the MLS problem to obtain the minimum $\sum_C w_C$ given that the benefit has to be at least k . Now, compare the solutions and take the one that has the best c_s/b_s ratio.* This completes the reduction.

Let us now prove that the greedy algorithm described above has an approximation factor of $O(\log n)$ for augmenting edge connectivity from one to two in the activated subgraph. The following lemma is crucial.

LEMMA 3.9. *Let $G = (V, E)$ be any connected graph and F_1, F_2 be two sets of edges on V . Let b_1, b_2 and $b_{1,2}$ be the decrease in the number of edges in the spanning tree of 2-edge-connected components of G due to the addition of F_1, F_2 and $F_1 \cup F_2$ respectively. Then, $b_{1,2} \leq b_1 + b_2$.*

Proof. The lemma follows from the observation that for any particular edge of the spanning tree, it appears in the benefit if it is in the fundamental cycle of any added edge.

Let (V, A) be a connected graph, and let Y be any minimal set of edges such that $(V, F \cup A)$ is 2-edge-connected. Because of minimality, Y must be a forest. We root each tree of this forest arbitrarily and decompose each tree into stars \mathcal{S} centered at each non-leaf vertex and containing its children as peripheral vertices. Then, the previous lemma ensures that \mathcal{S} has the following properties.

LEMMA 3.10. *\mathcal{S} satisfies both the following properties:*

- *Each vertex in V appears in at most two stars in \mathcal{S} .*

- *The sum of benefits of the stars in \mathcal{S} is at least the number of edges in the spanning tree on 2-edge-connected components of (V, A) .*

Proof. Each vertex appears in at most two stars—one where it is the center and another where its parent in F is the center. The second assertion follows from applying the above lemma multiple times.

Using standard techniques, we can now show the following theorem (proof similar to that of Theorem 3.1).

THEOREM 3.4. *The algorithm presented above achieves an approximation guarantee of $O(\log n)$ for the second step of the MEAN algorithm.*

Combining all the pieces, we get the following theorem.

THEOREM 3.5. *The two-stage algorithm for the MEAN problem for $k = 2$ has an approximation factor of $O(\log n)$.*

3.4 Minimum k -Edge-connected Activation Network. Our first goal in this section is to prove the following theorem that connects the MEAN and MDAN problems.

THEOREM 3.6. *An α -approximation algorithm for the MDAN problem implies an $O(\alpha \log n, 2)$ -approximation algorithm for the MEAN problem for arbitrary k .*

Then, we apply the above theorem to obtain $(O(\log^2 n), 2)$ -approximation algorithms for the MEAN problem with arbitrary k in the power optimization and installation cost optimization settings. We then refine both these algorithms to improve the approximation factor to $O(\log n \log k, 2)$.

As outlined in the introduction, our main tool in proving Theorem 3.6 is Theorem 1.1 that we stated in the introduction. In fact, we prove the following stronger theorem.

THEOREM 3.7. *For any k , if $(c + 1/2)n$ vertices in an undirected graph $G = (V, E)$ have degree at least k for some $0 < c \leq 1/2$, then the number of $\lceil k/2 \rceil$ -edge-connected components in G is at most $(1 - c/2)n$.*

Clearly, Theorem 1.1 follows from this theorem by setting $c = 1/2$. Also, in the remainder of this proof, we will replace $\lceil k/2 \rceil$ by $k/2$ for brevity; the proof holds for $\lceil k/2 \rceil$ as well.

To prove this theorem, we need to introduce a data structure called *Gomory-Hu trees* [13]. (We only need the version for unweighted graphs, and that is what we define.)

DEFINITION 3.3. A Gomory-Hu tree is a weighted tree T defined on the vertices of an undirected unweighted graph $G = (V, E)$ satisfying the following properties:

- For any pair of vertices $u, v \in V$, the number of edge-disjoint paths between u and v in G is equal to the minimum weight of an edge in the unique path between u and v in T .
- For any edge $e \in T$ with weight $w(e)$, the cut in G corresponding to the vertex partition produced by removing e from T has $w(e)$ edges in it.

Proof. (of Theorem 3.7) Let vertices that have degree at least k in G , but do not have k edge-disjoint paths to any other vertex in G , be called *dangerous* vertices. We will show that every dangerous vertex must have at least three neighbors in any Gomory-Hu tree T of G . Since any tree has at most $n/2$ vertices with degree three or more, it follows that there are at most $n/2$ dangerous vertices in G . But, note that every vertex that is not dangerous must be in a $k/2$ -edge-connected component that has at least two vertices. The theorem follows.

We now show that a dangerous vertex v has at least three neighbors in a Gomory-Hu tree T . By the second property of Gomory-Hu trees, every edge incident on v in G adds a weight of one to exactly one of the edges incident on v in T . Therefore, the sum of weights on edges incident on v in T is at least the total degree of v in G , which is at least k since v is a dangerous vertex. Thus, if there are less than three neighbors of v in T , there is at least one edge (u, v) incident on v in T with weight at least $k/2$. But property 1 then ensures that T has at least $k/2$ edge-disjoint paths in G between u and v , contradicting that v is dangerous.

Consider an iterative algorithm for MEAN that runs the α -approximation algorithm for MDAN in every round with the partition defined by the $k/2$ -edge connected components of the subgraph activated in previous iterations. By Theorem 1.1, this algorithm terminates in $O(\log n)$ iterations, thereby proving Theorem 3.6.

We now apply Theorem 3.6 to the power optimization and installation cost optimization settings. For power optimization, we can extend an algorithm due to Kortsarz et al [23] (which only considered singleton partitions \mathcal{P}) to obtain the following theorem.

THEOREM 3.8. *There is an $O(\log n)$ -approximation algorithm for the MDAN problem for power optimization.*

We give a new $O(\log n)$ -approximation algorithm for the MDAN problem for installation cost optimization. Recall that in this problem, every edge (u, v) has a threshold τ_{uv} and an edge is activated iff $\alpha_{u,uv}x_u +$

$\alpha_{v,uv}x_v \geq \tau_{uv}$. As in the MSpAT problem, our algorithm runs in rounds, where in each round, it greedily select a star with the minimum cost-benefit ratio to add to the set of activated edges. In this problem, we define the *cost* of a star s centered at v as $c_s = \max_{(u,v) \in s} \frac{\tau_{uv}}{\alpha_{v,uv}}$. Observe that we can activate all the edges of s by increasing the value of x_v to c_s and keeping the value of x_u for all other vertices u unchanged. We define the *benefit* of s as the overall decrease in the degree requirements of the subsets in partition \mathcal{P} when we activate the edges of s .

We first show that each round runs in polynomial time. This is simple now because we only need to consider n possible values of x_v for any vertex v (corresponding to $\frac{\tau_{uv}}{\alpha_{v,uv}}$), and calculate the benefit for each of these choices (keeping the values of all other vertices unchanged). To prove the approximation ratio, we use the following decomposition lemma.

LEMMA 3.11. *Suppose x_v for $v \in V$ activates a set of edges A in the installation cost optimization problem. Then, A can be decomposed into a set of center-disjoint stars such that all edges in a star centered at v can be activated by setting $2x_v$ at v and 0 at the peripheral vertices.*

Proof. For each edge $e \in A$, associate edge e with a unique endpoint v such that $x_v \geq \frac{\tau_e}{2\alpha_{v,uv}}$, breaking ties arbitrarily if required. Now, decompose A into each vertex and its associated set of edges; these stars satisfy the lemma.

The next theorem now follows from the above lemma using standard techniques (similar to the proof of Theorem 3.1).

THEOREM 3.9. *The greedy algorithm for the MDAN problem for installation cost optimization described above has an approximation ratio of $O(\log n)$.*

Using Theorem 3.6, we obtain an $(O(\log^2 n), 2)$ -approximation algorithm for the MEAN problem for installation cost optimization with arbitrary k . We now show that we can improve the approximation ratio for this problem to $O(\log n \log k, 2)$ by using a slightly more refined algorithm.

Improved Algorithm. The next result is for installation cost optimization; a similar result can be obtained in power optimization.

THEOREM 3.10. *For any $\epsilon > 0$, the approximation ratio of the MEAN algorithm for arbitrary k in the installation cost optimization setting can be improved to $(O(\log n \log(4 + 8/\epsilon)), 2 + \epsilon)$.*

If we choose $\epsilon < 4/(k-2)$, then integrality of edge connectivity implies that we obtain an $(O(\log n \log k), 2)$ -approximation algorithm.

The core idea in the above theorem is to exploit the greater generality of Theorem 3.7 compared to Theorem 1.1. In particular, if we are interested in a $k/(2+\epsilon)$ -edge-connected activation subgraph, then the following lemma shows that we can terminate the greedy algorithm for MDAN early.

LEMMA 3.12. *If the total degree requirement satisfied by a partial solution to the MDAN problem is at least $kn \left(\frac{8+3\epsilon}{8+4\epsilon}\right)$, then there are at least $3n/4$ vertices with degree at least $\left(\frac{2}{2+\epsilon}\right)k$.*

Proof. Let us order the vertices v_1, v_2, \dots, v_n by decreasing degree in the activated subgraph. Then, the degree of $v_{3n/4}$ is at least

$$\frac{kn \left(\frac{8+3\epsilon}{8+4\epsilon}\right) - \frac{3kn}{4}}{\frac{n}{4}} = k \left(\frac{2}{2+\epsilon}\right).$$

We terminate the MDAN algorithm once it satisfies $\frac{8+3\epsilon}{8+4\epsilon}$ fraction of the total degree requirement. The following theorem shows that this modified MDAN algorithm has a better approximation ratio; Theorem 3.10 follows by setting $\delta = \frac{8+3\epsilon}{8+4\epsilon}$.

THEOREM 3.11. *If the MDAN algorithm is terminated when it has satisfied δ fraction of the total degree requirement, then the cost of the partial MDAN solution is at most $O(\log \frac{1}{1-\delta})$ times that of the optimal.*

Proof. Due to the early termination, the total cost of the partial MDAN solution is at most

$$\begin{aligned} & \left(\frac{1}{nk} + \frac{1}{nk-1} + \dots + \frac{1}{\delta nk}\right) \text{OPT} \\ &= O\left(\log \frac{1}{1-\delta}\right) \text{OPT}, \end{aligned}$$

where OPT is the cost of an optimal MDAN solution.

4 Steiner Connectivity Problems

In this section, we will focus on problems where the connectivity requirement is specific to a subset of vertices (called the *terminals*).

4.1 Minimum Activation Path. We first give an exact algorithm for the MAP problem with terminals s, t . The algorithm mimics Bellman-Ford's single-source shortest path algorithm (see e.g. [9]) with source vertex s , except that the dynamic program has to be additionally parameterized by the value $x_v \in D$ chosen at the

destination v . Let $d(v, a)$ and $\pi(v, a)$ be variables respectively representing the length and the predecessor of v in the shortest path from s to v discovered thus far, where $x_v = a \in D$. Initially, $d(v, a) = \infty$ and $\pi(v, a) = \text{NULL}$ for all vertices $v \neq s$, and for all $a \in D$; also, $d(s, a) = a$ and $\pi(s, a) = \text{NULL}$ for all $a \in D$. The algorithm runs in $n-1$ rounds, where in each round it *relaxes* each edge (u, v) by making the following updates for each $a, b \in D$ such that $f_{uv}(a, b) = 1$:

- if $d(v, b) > d(u, a) + b$, update $d(v, b)$ to $d(u, a) + b$ and $\pi(v, b)$ to (u, a) .
- if $d(u, a) > d(v, b) + a$, update $d(u, a)$ to $d(v, b) + a$ and $\pi(u, a)$ to (v, b) .

The final output is $\min_{a \in D} d(t, a)$ and the corresponding path given by the predecessors.

The following lemma is crucial to proving correctness of the above algorithm.

LEMMA 4.1. *Suppose u is the immediate neighbor of v on a shortest path between s and v with $x_v = a$. Also, let $x_u = b$ on this path. Then, the prefix of this path between s and u is a shortest path between s and u where $x_u = b$.*

Proof. If not, we can replace the s to u segment of the s to v shortest path with the shorter alternative path. Since x_u and x_v remain unchanged, the edge (u, v) remains activated.

We now use the above lemma to prove the following lemma; setting $i = n-1$ in the lemma proves correctness of the algorithm.

LEMMA 4.2. *If a shortest path from s to v with $x_v = a$ contains i edges, then $d(v, a)$ and $\pi(v, a)$ are correctly set after i rounds of the algorithm.*

Proof. We prove by induction on i . The base case, for $i = 0$, is immediate. For the inductive case, let u be the neighbor of v on the shortest path from s to v with $x_v = a$; let $x_u = b$ on this path. By Lemma 4.1 and the inductive hypothesis, $d(u, b)$ and $\pi(u, b)$ are correctly set at the end of round $i-1$. The proof follows since edge (u, v) is relaxed with values $x_u = b, x_v = a$ in round i .

4.2 Minimum Activation Flow. We now turn our attention to the MAF problem for arbitrary flow requirement k between the pair of terminals s and t . We show that this problem generalizes the following problem.

Minimum Node-weighted k -Flow (MNF). Given a node-weighted undirected graph and two terminals s

and t , and a flow requirement k , find the minimum cost set of k edge-disjoint paths between s and t .

Nutov [29] observed that the MNF problem is at least as hard as the well-known ℓ -densest subgraph problem [12, 2]. Since our problem generalizes MNF, it is at least as hard as well.

To encode the MNF problem as a special case of the MAF problem, let D be the set of node weights in the graph and the activation function f_{uv} for edge (u, v) be defined as (c_u (resp., c_v) is the cost of vertex u (resp., v)):

$$f_{uv}(x_u, x_v) = \begin{cases} 1, & \text{if } x_u \geq c_u \text{ and } x_v \geq c_v \\ 0, & \text{otherwise.} \end{cases}$$

4.3 Minimum Steiner Activation Network. We now give an $O(\log n)$ -approximation algorithm for the MSAN problem. We give an approximation-preserving reduction to the following problem:

Minimum Node-weighted Steiner Network (MNSN). Given a graph with k sets of *terminals* R_1, R_2, \dots, R_k and weights on the *steiner* (i.e. non-terminal) nodes, find a minimum-weight forest spanning the terminals such that any two terminals in the same set R_i are connected in the forest.

Klein and Ravi [22] gave an $O(\log n)$ -algorithm for this problem based on a *spider decomposition* technique. We can use this algorithm, and the reduction that we describe, to obtain an $O(\log n)$ -approximation algorithm for our problem.

Given an instance of the MSAN problem on a set of vertices V with terminal sets $R_1, R_2, \dots, R_k \subseteq V$ and activation functions f_{uv} , we construct an instance of the MNSN problem as follows: *For each vertex $v \in V$, construct a star with $|D|+1$ nodes having v_0 as its center and $\{v_a : a \in D\}$ as the peripheral vertices; v_0 has weight 0 while v_a has weight a . Now, connect u_a to v_b with an edge iff $f_{uv}(a, b) = 1$. The terminal sets in the constructed graph are $\{v_0 : v \in R_i\}$.* The following theorem establishes the validity of the reduction.

THEOREM 4.1. *For any solution to the instance of the MSAN problem, there is a solution to the instance of the MNSN problem constructed by the reduction with at most as much cost, and vice-versa.*

Proof. Suppose that in the MSAN solution, $x_v = a_v$ for vertices $v \in V$ and let the edges in the steiner forest be T . Then, the edges $\{(v_0, v_{a_v}) : v \in T\}$ and edges $\{(u_{a_u}, v_{a_v}) : (u, v) \in T\}$ connect terminals in the same set and have total node cost equal to $\sum_{v \in V} a_v$. Conversely, for any solution to the MSAN instance, if there are multiple v_a nodes in the steiner forest,

then we only retain the node with the maximum value of a . Since the activation functions are monotonic, this retained node has edges to all nodes that were connected via the non-retained nodes. Therefore, after the transformation, we obtain a new steiner forest with at most as much weight as the original solution. Now, in the MSAN instance, set $x_v = a$ if v_a is selected in the transformed solution to the MNSN instance. Clearly, all terminals in each set can be connected using these values of x_v since the solution to the MNSN instance contained a steiner forest connecting terminals in the same set.

5 Future Work

The activation network model described in this paper opens up a new set of practically relevant network design problems. One important direction of future research is to obtain similar results in directed networks. Observe that our framework extends naturally to directed networks by removing the restriction that $f_{uv} = f_{vu}$ on activation functions. However, the algorithmic techniques used here are tailored to undirected graphs, and do not, in general, extend to directed graphs. In fact, some of the results here also seem specific to undirected graphs; e.g., finding a near-optimal *directed steiner tree* is a major open question even in the traditional edge-weighted network model. On the other hand, other problems such as finding a near-optimal *arborescence* or *strongly connected subgraph* might be more tractable.

Another interesting direction of research is to restrict the activation functions further to obtain better approximation ratios. For example, the minimum steiner tree problem (or even the much richer generalized steiner network problem [19]) admits constant factor approximation algorithms in edge-weighted graphs. These problems are special cases of our general framework; so is this a manifestation of some special structural property in their activation functions? Can we identify and exploit these structural properties to obtain good approximation algorithms for wider subclasses of activation functions? We leave these questions for future investigation.

Acknowledgements

The author would like to thank colleagues at Bell Labs Research, Bangalore for introducing him to the problem of installation cost optimization in wireless networks. The author is also grateful to Yossi Azar, Kamal Jain, David Karger and Sudeepa Roy for helpful discussions, and to the anonymous reviewers for useful suggestions.

References

- [1] Ernst Althaus, Gruia Calinescu, Ion I. Mandoiu, Sushil K. Prasad, N. Tchernovski, and Alexander Zelikovsky. Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. *Wireless Networks*, 12(3):287–299, 2006.
- [2] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $(1/4)$ approximation for densest -subgraph. In *STOC*, pages 201–210, 2010.
- [3] Sanjit Biswas and Robert Morris. Exor: opportunistic multi-hop routing for wireless networks. In *SIGCOMM*, pages 133–144, 2005.
- [4] Gruia Calinescu, Sanjiv Kapoor, Alexander Olshevsky, and Alexander Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In *ESA*, pages 114–126, 2003.
- [5] Gruia Calinescu and Peng-Jun Wan. Range assignment for biconnectivity and k -edge connectivity in wireless ad hoc networks. *MONET*, 11(2):121–128, 2006.
- [6] Andrea E. F. Clementi, Paolo Penna, and Riccardo Silvestri. Hardness results for the power range assignment problem in packet radio networks. In *RANDOM-APPROX*, pages 197–208, 1999.
- [7] Andrea E. F. Clementi, Paolo Penna, and Riccardo Silvestri. The power range assignment problem in radio networks on the plane. In *STACS*, pages 651–660, 2000.
- [8] Andrea E. F. Clementi, Paolo Penna, and Riccardo Silvestri. On the power assignment problem in radio networks. *MONET*, 9(2):125–140, 2004.
- [9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
- [10] Arindam Kumar Das. Minimum power broadcast trees for wireless networks: Integer programming formulations. In *INFOCOM*, 2003.
- [11] Partha Dutta, Sharad Jaiswal, Debmalya Panigrahi, K. V. M. Naidu, Rajeev Rastogi, and Ajay Kumar Todimala. Villagenet: A low-cost, 802.11-based mesh network for rural regions. In *COMSWARE*, 2007.
- [12] Uriel Feige, David Peleg, and Guy Kortsarz. The dense -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.
- [13] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 9(4):551–570, 1961.
- [14] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.
- [15] Mohammad Taghi Hajiaghayi, Nicole Immorlica, and Vahab S. Mirrokni. Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. *IEEE/ACM Trans. Netw.*, 15(6):1345–1358, 2007.
- [16] Mohammad Taghi Hajiaghayi, Guy Kortsarz, Vahab S. Mirrokni, and Zeev Nutov. Power optimization for connectivity problems. *Math. Program.*, 110(1):195–208, 2007.
- [17] Frank Harary. *Graph Theory*. Addison-Wesley, 1969.
- [18] Robert J. Marks II, Arindam Kumar Das, Mohamed A. El-Sharkawi, Payman Arabshahi, and Andrew Gray. Minimum power broadcast trees for wireless networks: optimizing using the viability lemma. In *ISCAS (1)*, pages 273–276, 2002.
- [19] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
- [20] Intae Kang and Radha Poovendran. Iterated local optimization for minimum energy broadcast. In *WiOpt*, pages 332–341, 2005.
- [21] Intae Kang and Radha Poovendran. Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks. *MONET*, 10(6):879–896, 2005.
- [22] Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.
- [23] Guy Kortsarz, Vahab S. Mirrokni, Zeev Nutov, and Elena Tsanko. Approximating minimum-power degree and connectivity problems. In *LATIN*, pages 423–435, 2008.
- [24] Guy Kortsarz and Zeev Nutov. Approximating minimum-power edge-covers and 2, 3-connectivity. *Discrete Applied Mathematics*, 157(8):1840–1847, 2009.
- [25] Yuval Lando and Zeev Nutov. On minimum power connectivity problems. *J. Discrete Algorithms*, 8(2):164–173, 2010.
- [26] Weifa Liang. Constructing minimum-energy broadcast trees in wireless ad hoc networks. In *MobiHoc*, pages 112–122, 2002.
- [27] Zeev Nutov. Approximating minimum power covers of intersecting families and directed connectivity problems. In *APPROX-RANDOM*, pages 236–247, 2006.
- [28] Zeev Nutov. Approximating minimum-power k -connectivity. In *ADHOC-NOW*, pages 86–93, 2008.
- [29] Zeev Nutov. Approximating steiner networks with node weights. In *LATIN*, pages 411–422, 2008.
- [30] Zeev Nutov. Approximating minimum-power k -connectivity. *Ad Hoc & Sensor Wireless Networks*, 9(1-2):129–137, 2010.
- [31] Debmalya Panigrahi, Partha Dutta, Sharad Jaiswal, K. V. M. Naidu, and Rajeev Rastogi. Minimum cost topology construction for rural wireless mesh networks. In *INFOCOM*, pages 771–779, 2008.
- [32] Sayandeep Sen and Bhaskaran Raman. Long distance wireless mesh network planning: problem formulation and solution. In *WWW*, pages 893–902, 2007.
- [33] Liansheng Tan, Xiaoli Zhan, Jie Li, and Fuzhe Zhao. A novel tree-based broadcast algorithm for wireless ad hoc networks. *IJWMC*, 1(2):156–162, 2006.