Survivable Network Design with Degree or Order Constraints

[Extended Abstract]

Lap Chi Lau [*]
CSE Dept.
The Chinese Univ. of Hong Kong
Shatin, Hong Kong
chi@cse.cuhk.edu.hk

Joseph (Seffi) Naor [†]
Microsoft Research
One Microsoft Way
Redmond, WA 98052
naor@cs.technion.ac.il

Mohammad R. Salavatipour [‡]
Dept. of Computing Science
University of Alberta
Edmonton, Alberta Canada
mreza@cs.ualberta.ca

Mohit Singh [§]
Tepper School of Business
Carnegie Mellon University
mohits@andrew.cmu.edu

ABSTRACT

We present algorithmic and hardness results for network design problems with degree or order constraints. We first consider the SURVIVABLE NETWORK DESIGN problem with degree constraints on vertices: the objective is to find a minimum cost subgraph satisfying certain connectivity requirements as well as degree upper bounds on the vertices. A well known special case is the MINIMUM BOUNDED DEGREE SPANNING TREE problem which has attracted much attention recently. Denote by B_v the degree constraint of vertex v. We present a $(2, 2B_v + 3)$-approximation algorithm for the element-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints on terminals, i.e., the cost of the solution is at most twice the optimum solution (satisfying the degree bounds), and the degree of each terminal vertex v is at most $2B_v + 3$. This extends the most general network design model which admits a 2-approximation algorithm (with no degree constraints), and implies the first constant factor (bicriteria) approximation algorithms for many network design problems with degree constraints, including the MINIMUM BOUNDED DEGREE STEINER TREE problem. In the edge connectivity SURVIVABLE NETWORK DESIGN problem, the algorithm has an interesting feature that the *average* degree of the returned solution is only violated by an additive constant of 2. Our results also extend to directed graphs and we provide the first constant factor (bicriteria) approximation algorithms for, e.g., the MINIMUM BOUNDED DEGREE ARBORESCENCE problem and the MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH problem. A striking aspect of our method is its simplicity. It is based on a natural extension of Jain's iterative rounding method. This provides an elegant and unifying algorithmic framework for a broad range of network design problems with degree constraints. In contrast, we show that the vertex-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints is very hard to approximate, even if the costs of all edges are zero.

We also study the problem of finding a minimum cost λ-edge-connected subgraph with at least k vertices, which we call the (k, λ)-subgraph problem. This generalizes some well-studied classical problems such as the k-MST and the minimum cost λ-edge-connected subgraph problems. We give a poly-logarithmic approximation for the $(k, 2)$-subgraph problem. However, by relating it to the DENSEST k-SUBGRAPH problem, we give evidence that the (k, λ)-subgraph problem might be hard to approximate for arbitrary λ.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non Numerical Algorithms and Problems—*Computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network Problems.*

General Terms

Algorithms, Performance.

Keywords

Approximation algorithms for NP-hard problems, Network design, Bounded degree, Iterative rounding.

1. INTRODUCTION

Network design is a central topic in combinatorial optimization, approximation algorithms, and operations research. The basic set-

[*]This work was partly done during a visit to Egerváry Research Group on Combinatorial Optimization (EGRES) in Budapest. Supported by European MCRTN Adonet, Contract Grant No. 504438.

[†]On leave from the Computer Science Dept., Technion, Haifa, Israel.

[‡]Supported by NSERC and a start-up grant from the University of Alberta.

[§]Supported by NSF ITR grant CCR-0122581 (The ALADDIN project) and NSF Grant CCF- 0430751.

STOC'07, June 11–13, 2007, San Diego, California, USA.

ting of network design problems is to find a minimum cost subgraph satisfying connectivity requirements between vertices. This captures a wide variety of classical problems such as MINIMUM COST FLOW, MINIMUM STEINER TREE, HAMILTONIAN CYCLE, etc. Furthermore, research results in this area provide algorithmic tools and insights (e.g., hardness results) for the design of practical networks such as telecommunication networks. Notable successes along this line of research are Jain's 2-approximation algorithm for the edge-connectivity SURVIVABLE NETWORK DESIGN problem [18], and its generalization to element-connectivity [7, 5].

A recent research trend is to study a more general class of network design problems where there are natural budget constraints. This is motivated by the need for more sophisticated and realistic models for the design of practical networks. The first type of constraints we study is *degree constraints* on vertices. The objective is to find a minimum cost subgraph satisfying connectivity requirements as well as degree bounds (e.g. workloads) on the vertices. A well-known example is the MINIMUM BOUNDED DEGREE SPANNING TREE problem, which includes the TRAVELING SALESMAN problem as a special case. Very recently, Goemans [14] obtained an approximation algorithm for this problem, with only an additive error of two on the degrees, following a long line of research. We note that the basis underlying the breakthrough results of Goemans [14] and Jain [18] is the *uncrossing technique* in combinatorial optimization.

We study a common generalization of the above two problems. Our main result is a generalization of the 2-approximation algorithm for the element-connectivity SURVIVABLE NETWORK DESIGN problem (which is the most general network design problem admitting a 2-approximation algorithm), providing near-optimal bounds on the degrees. This yields the first constant factor (bicriteria) approximation algorithms for many network design problems with degree constraints, including MINIMUM BOUNDED DEGREE STEINER NETWORK, MINIMUM BOUNDED DEGREE STEINER TREE, etc. Our results extend to directed graphs and we provide the first constant factor (bicriteria) approximation algorithms for MINIMUM BOUNDED DEGREE ARBORESCENCE, MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH, etc. A striking aspect of our method is its *simplicity*. Our approach is based on a natural extension of Jain's *iterative rounding* method. This provides an elegant and unifying algorithmic framework for a broad range of network design problems with degree constraints. In fact, very recently, the techniques used in this paper have been extended to give an $(1, B_v + 1)$-approximation algorithm for the MINIMUM BOUNDED DEGREE SPANNING TREE problem [34], settling a 15-year-old conjecture affirmatively. In contrast, we present hardness results for the vertex-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints, even if all edges have zero cost.

The second type of constraints we study is *order constraints*. Specifically, we study the problem of finding a minimum cost λ-edge-connected subgraph with at least k vertices, which we call the (k, λ)-subgraph problem. This generalizes some classical and well-studied problems such as the k-MST problem (which is the $(k, 1)$-subgraph problem) and the minimum cost λ-edge-connected subgraph problem (which is the (n, λ)-subgraph problem with n being the number of vertices). We give a poly-logarithmic approximation algorithm for the $(k, 2)$-subgraph problem. However, by relating it to the DENSEST k-SUBGRAPH problem, we give evidence that the (k, λ)-subgraph problem might be hard to approximate for arbitrary λ.

1.1 Previous Work

Network design problems have a very rich literature. For classical network design problems, we shall just highlight a few results and refer the reader to [24] for a survey. In the SURVIVABLE NETWORK DESIGN problem, we are given a connectivity requirement r_{uv} for each pair of vertices, and the goal is to find a minimum cost subgraph satisfying the connectivity requirements. This is a very general problem which captures many interesting problems as special cases (e.g., minimum Steiner tree, minimum Steiner forest, minimum k-edge-connected subgraph) and has many applications. Jain [18] gave a 2-approximation algorithm for the edge-connectivity SURVIVABLE NETWORK DESIGN problem by using an elegant iterative rounding approach. Later, Fleischer, Jain, and Williamson [7] have generalized this result to element-connectivity SURVIVABLE NETWORK DESIGN problem (another solution was proposed in [5]). On the other hand, the vertex-connectivity SURVIVABLE NETWORK DESIGN problem is shown to be very hard to approximate [23].

Network design problems with degree constraints have been studied extensively in the last 15 years . A simpler setting is minimizing the maximum degree subgraph (without considering the cost) satisfying certain connectivity requirements. A well-known example is the MINIMUM DEGREE SPANNING TREE (MDST) problem, where the objective is to find a spanning tree of smallest maximum degree. This problem is already NP-hard as it generalizes the HAMILTONIAN PATH problem. Fürer and Raghavachari [8, 9] gave an elegant approximation algorithm returning a solution with maximum degree at most one off the optimal solution. (The result holds for the Steiner version of the problem as well.) Ravi, Raghavachari, and Klein [30, 19] considered the MINIMUM DEGREE k-EDGE-CONNECTED SUBGRAPH problem, and gave an approximation algorithm with performance ratio $O(n^\delta)$ for any fixed $\delta > 0$ in polynomial time, and $O(\log n / \log \log n)$ in sub-exponential time. Recently, Feder, Motwani, and Zhu [6] obtained a polynomial time $O(k \log n)$-approximation algorithm for this problem, for any *fixed* k, thus answering an open question in [30]. Our main result implies the first constant factor approximation algorithm even for the most general edge-connectivity requirements.

For the more general problem of finding a minimum cost subgraph with given connectivity requirements and degree bounds B_v on every vertex v, the most-studied case is the MINIMUM BOUNDED DEGREE SPANNING TREE (MBDST) problem. Let OPT be the cost of an optimal solution to the MBDST problem. We say an algorithm is an $(\alpha, f(B_v))$-approximation algorithm if the returned solution has cost at most $\alpha \cdot$ OPT (with OPT being the cost of the optimum solution satisfying the degree bounds) and the degree at each vertex v is at most $f(B_v)$. The first approximation was an $(O(\log n), O(\log n \cdot B_v))$-algorithm by [27, 29]. This was subsequently improved in a series of papers [21, 22, 3, 4, 31]. Very recently, Goemans [14] made a breakthrough on this problem by giving a $(1, B_v + 2)$-approximation algorithm. Remarkably, the proof of the result of Goemans is considerably simpler than that of the previous results. Very little is known for more general connectivity requirements. For the MINIMUM BOUNDED DEGREE STEINER TREE problem, there is an $(O(\log n), O(B_v \log n))$-approximation algorithm [29]. This bound was improved to $(O(1), O(B_v + \log n))$-approximation by [20], but the algorithm runs in quasi polynomial time. Our main result implies the first polynomial time $(2, 2B_v + 3)$-approximation even for the most general model of edge-connectivity requirements.

For network design problem with order constraints, the most well-studied problem is the k-MST problem, where the objective is to find a minimum cost tree spanning at least k vertices. The approximation factor for this problem was improved from $\sqrt{k}$ and

$O(\log^2 k)$ in [28, 1] down to constant in [2, 12] and very recently to 2 [13]. For the case of metric costs on the edges, the k-TSP problem, which asks to find a minimum cost TSP tour visiting at least k vertices, can also be approximated within factor 2 [13].

1.2 New results

Suppose that we are given an undirected graph with connectivity requirements r_{uv} on pairs of vertices u and v, and degree bounds B_v on each vertex v. The edge-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints asks for a minimum cost subgraph such that there are at least r_{uv} edge disjoint paths between vertices u and v and the degree of each vertex is at most B_v. We obtain the following result.

THEOREM 1.1. *There is a polynomial time $(2, 2B_v + 3)$- approximation algorithm for the edge-connectivity* SURVIVABLE NETWORK DESIGN *problem. Moreover, on average, the degree bounds are violated by at most 2.*

This gives the first constant factor bicriteria approximation algorithms for a broad range of network design problems with degree constraints such as the MINIMUM STEINER TREE problem, the MINIMUM STEINER FOREST problem, the MINIMUM k-EDGE-CONNECTED SUBGRAPH problem, etc. It also implies the first constant factor approximation algorithm for minimizing the maximum degree version of many problems (by setting $B_v = B$ for all v).

We then consider the more general element-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints. Given an undirected graph G, the set of vertices is partitioned into terminals and non-terminals. The edges and the non-terminals are called *elements*. Suppose that we are given a connectivity requirement r_{uv} for each pair of terminal vertices u and v, and a degree bound B_v on each terminal vertex v. The objective is to find a minimum cost subgraph such that there are at least r_{uv} *element*-disjoint paths between terminal vertices u and v and the degree of each terminal vertex v is at most B_v. We obtain the following result.

THEOREM 1.2. *There is a polynomial time $(2, 2B_v + 3)$- approximation algorithm for the element-connectivity* SURVIVABLE NETWORK DESIGN *problem with degree constraints on terminals.*

We remark that both Theorems 1.1 and 1.2 hold for (1) connectivity requirements that are *weakly supermodular* (technical definition is deferred to later); and (2) the case where there are both lower and upper degree bounds. In fact, the lower bounds will never be violated. For directed graphs, we study the problem of finding a minimum cost subgraph which satisfies connectivity requirements that are *intersecting supermodular* or *crossing supermodular* (technical definition is deferred to later) and indegree and outdegree constraints. This includes the MINIMUM BOUNDED DEGREE ARBORESCENCE problem, MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH problem, etc. We obtain the following result.

THEOREM 1.3. *There is a polynomial time $(4, 4B_v^{in} + 6, 4B_v^{out} + 6)$-approximation algorithm to find a minimum cost subgraph satisfying intersecting supermodular connectivity requirements, together with indegree and outdegree constraints in directed graphs. For crossing supermodular connectivity requirements, there is a polynomial time $(8, 8B_v^{in} + 12, 8B_v^{out} + 12)$-approximation algorithm.*

We have not tried to optimize the constants in Theorem 1.3; in fact we believe they can be improved (we leave the details for the journal version). We state it here just to illustrate the generality of the technique and the scope it can be applied to. In contrast to the above theorems, we present a hardness result for the vertex-connectivity version of the SURVIVABLE NETWORK DESIGN problem with degree constraints, even if the cost of the subgraph is not considered.

THEOREM 1.4. *For any $\epsilon > 0$, there is no polynomial time $(\infty, 2^{\log^{1-\epsilon} n} B_v)$-approximation algorithm for the vertex connectivity* SURVIVABLE NETWORK DESIGN *problem unless $NP \subseteq DTIME(n^{polylog(n)})$.*

Next, we turn our attention to network design problems with order constraints. We study the (k, λ)-subgraph problem, i.e. the problem of finding a minimum cost λ-edge-connected subgraph with at least k vertices. This problem generalizes the classical k-MST problem to higher connectivity requirements. However, it seems that this line of generalization might be difficult as shown by the following result.

THEOREM 1.5. *An α-approximation algorithm for the (k, λ)-subgraph problem (even for the unweighted case) for arbitrary λ, implies an $(\alpha \log^2 k)$-approximation algorithm for the* DENSEST k-SUBGRAPH *problem.*

Notice that the best known approximation algorithm for the DENSEST k-SUBGRAPH problem has ratio $O(n^{\frac{1}{3}-\epsilon})$ for some constant $\epsilon > 0$. Finally, for the $(k, 2)$-subgraph problem, we are able to obtain the following.

THEOREM 1.6. *There is an $O(\log^3 k)$-approximation algorithm for the $(k, 2)$-subgraph problem.*

1.3 Techniques and Overview

Iterative rounding for the edge connectivity SURVIVABLE NETWORK DESIGN problem (without degree constraints) works as follows. Formulate the SURVIVABLE NETWORK DESIGN problem as an integer program, and then solve the linear programming relaxation of the problem to find a basic optimum solution x. Pick an edge e^* with highest value (i.e. $x_{e^*} \geq x_e$ for all $e \in E$) and add it to the solution subgraph H (initially H is empty). Then consider the residual problem, where the edges in H are pre-selected, and repeat the above procedure (find a basic optimum solution, add an edge with highest value to H, and construct the residual problem) until all the connectivity requirements are satisfied. Jain [18] proved that the edge picked in each iteration has value at least 1/2 (i.e. $x_{e^*} \geq 1/2$), implying a 2-approximation algorithm for the problem.

We return to our problem. The starting point is that degree constraints are defined only on single vertices, and so the uncrossing technique as in [18, 14] can be applied to show that a basic optimal solution is characterized by a laminar family of tight sets. This immediately implies that, in the first iteration, there exists an edge having value at least 1/2. Now comes the key difference. Since degree constraints are *packing* constraints, we must allow for *non-integral* degree constraints in the residual problem, otherwise the residual problem may be infeasible, or its cost may significantly increase. By doing so, however, it is not necessarily true anymore that the picked edges in later iterations have value at least 1/2. We are indeed going to decrease the degree constraints by fractional values $\geq 1/2$. However, to overcome the latter difficulty the "problematic" degree constraints are identified, deleted from the residual problem, and a basic solution is computed again. This incurs an extra additive constant 3 in the approximation ratio. Once "problematic" degree constraints are deleted, we can show that the picked edges in the residual problems always have value at least 1/2 (even though there can be non-integral degree constraints). This implies a $(2, 2B_v + 3)$-approximation algorithm for the problem.

The above technique is also adapted to prove the claimed guarantees for the element connectivity SURVIVABLE NETWORK DESIGN problem (Theorem 1.2) and for the directed graph result (Theorem 1.3). In fact, the technique developed is so general and powerful that it can be extended to settle the conjecture on the MINIMUM BOUNDED DEGREE SPANNING TREE problem [34] affirmatively, i.e. to give a $(1, B_v + 1)$-approximation algorithm for the MBDST problem.

2. SURVIVABLE NETWORK DESIGN WITH DEGREE CONSTRAINTS

For ease of exposition, we start with Theorem 1.1. Theorem 1.2 is deferred to the full version of this paper.

2.1 Edge-Connectivity SNDP with Degree Constraints

This section addresses a generalization of the EC-SNDP with non-uniform upper and lower bounds on vertex degrees, and presents a bicriteria approximation algorithm which will imply Theorem 1.1. More specifically, assume we are given a complete graph $G = (V, E)$ and nonnegative costs $c : E \to \mathbb{R}^+$ for the edges, an integer valued connectivity requirements function $r_{i,j}$ on pairs of vertices i, j, and a degree upper bound B_i and lower bound L_i for each vertex i. We also have an upper-bound $U_e \geq 1$ on the multiplicity of edge e in the solution (which would be 1 if each edge can be picked only once). The goal is to find a subset of edges F of minimum cost such that the subgraph $H = (V, F)$ satisfies the connectivity requirements and the degree bounds, that is, H has $r_{i,j}$ edge disjoint paths between vertices i, j, for each pair i, j, and each vertex i has $L_i \leq \deg_H(i) \leq B_i$ and each edge e appears at most U_e times in H.

Our method is a simple extension of Jain's iterative rounding method for solving EC-SNDP (above problem without degree constraints). Jain's method consists of solving an LP (linear programming) relaxation, finding an optimal basic solution x^*, rounding an edge e with $x^*_e \geq \frac{1}{2}$ to 1, and then repeating with a new LP relaxation for the residual problem (the problem obtained from the original one by adding edge e to the solution subgraph). The key point in Jain's method and proof is that every basic solution has an edge with value $\geq \frac{1}{2}$. We show that with some extra care, the same property holds in our more general setting. Thus our method finds a subgraph that satisfies all of the edge connectivity requirements, such that the degree of every vertex i is at least L_i and at most $2B_i + 3$ and has cost within a factor of 2 of the LP optimal cost[1].

An integer function on sets of vertices $f : 2^V \to \mathbb{Z}^+$ that has $f(V) = 0$ is called *weakly supermodular* if one of the two inequalities: $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$ or $f(A) + f(B) \leq f(A - B) + f(B - A)$, holds for every pair of sets $A, B \subseteq V$ with $A \cap B \neq \emptyset$. An important point in Jain's method is that the connectivity requirements are specified via a weakly supermodular function f on the sets of vertices; it is well known that a connectivity requirement function $r_{i,j}$ (i.e., $r : V \times V \to \mathbb{Z}^+$) is captured by a weakly supermodular function f. When we update the LP from one iteration to the next one (to account for one or more edges being added to the solution subgraph), it is easy to update the function f and to verify that the new function stays weakly supermodular. We denote $x(U) := \sum_{e \in U} x_e$, and $\delta(S)$ for a set $S \subset V$ denotes the set of edges with exactly one end-point in S. Here is the LP relaxation for EC-SNDP with degree bounds:

$$
\begin{array}{llrcll}
\text{(LP)} & \text{minimize} & z_{LP} & = & \sum_{e \in E} c_e\, x_e & \\
& \text{subject to} & x(\delta(S)) & \geq & f(S), & \forall\, S \subseteq V \\
& & x(\delta(i)) & \geq & L_v, & \forall\, v \in V \\
& & x(\delta(i)) & \leq & B_v, & \forall\, v \in V \\
& & 0 \;\leq\; x_e & \leq & U_e & \forall\, e \in E
\end{array}
$$

Clearly the first part of Theorem 1.1 follows from the following.

THEOREM 2.1. *If the LP has an optimal solution of cost z_{LP}, then there exists an integral solution $\hat{x}$ of cost $\leq 2z_{LP}$ that satisfies all of the constraints on f and $L_i \leq \sum_{e \in \delta(i)} \hat{x}_e \leq 2B_i + 3, \forall i \in V$. Moreover, $\hat{x}$ can be computed in polynomial time.*

The proof of this theorem follows from an extension of Jain's method [18] explained below. First note that any degree lower bound constraint can be considered as a connectivity constraint (with $f(\{v\}) = L_v$ for the cut $S = \{v\}$). By doing so, the new function f obtained from the connectivity constraints and degree lower bounds remains weakly supermodular. Therefore, if we satisfy all the connectivity constraints then we have satisfied the degree lower bounds too. So from now on, we assume that aside from connectivity constraints we only have degree upper bounds. We simply refer to them as degree constraints. Since the algorithm may change the degree bounds B_i's to fractional values, we may assume that B_i's are all fractional.

We say that a pair of sets A, B *intersect properly* if all of the sets $A \cap B, A - B, B - A$ are nonempty, and we say that a family of sets $\mathcal{L} = \{A_1, A_2, \ldots, A_\ell\}$ is *laminar* if no two of its sets are properly intersecting. For any set $A \subseteq V$, let χ_A denote the incidence vector of the set of edges $\delta(A)$; note that in the constraints matrix of the LP, χ_A is the row for the set A (the constraint for A may be either a connectivity constraint or a degree bound). Consider any solution x of the LP. We call a set of vertices A *tight* (w.r.t. x) if either

1. Initialization $F \leftarrow \emptyset$, $f' \leftarrow f$, and $\forall i \in V$: $B'_i = B_i$;
2. While $f' \neq \emptyset$ do
   (a) Find a basic feasible solution x with cut requirement f' and remove every edge e with $x_e = 0$.
   (b) If there are any B'_i's where vertex i has at most 4 nonzero incident edges delete those constraints and goto the next iteration.
   (c) For each edge $e = (u, v)$ with $x_e \geq 1/2$ add $\lceil x_e \rceil$ copies of x_e to F and decrease the bounds for B'_u and B'_v by x_e.
   (d) For every $S \subseteq V$: $f'(S) \leftarrow f(S) - |\delta_F(S)|$.
3. return $H = (V, F)$.

Figure 1: Bounded Degree EC-SND Algorithm

[1]Here and elsewhere, we assume that the LP relaxation has an optimal solution, that is, it has a solution of minimum cost that satisfies all of the constraints; of course, if the LP has no feasible solution, then the problem has no feasible solution, and so the issue of approximation is pointless.

$A = \{v\}$ and x satisfies the degree constraint for v with equality, $x(\delta(v)) = B_v$, or x satisfies the connectivity constraint for A with equality, $x(\delta(A)) = f(A)$ (in the latter case, A may be a singleton or not). The following lemma characterizes the tight constraints (i.e. constraints satisfied as equalities) of a basic feasible solution.

LEMMA 2.2. *Let the requirement function f of (LP) be weakly supermodular, and let x be a basic solution of (LP) such that $0 < x_e < 1$ for all edges $e \in E$. Then, there exists a laminar family $\mathcal{L}$ of tight sets such that $\mathcal{L}$ partitions into a set of singletons $\mathcal{L}'$ for the degree constraints, and the remaining sets $\mathcal{L}'' = \mathcal{L} - \mathcal{L}'$ for the connectivity constraints, such that:*
(i) Every set $A = \{v\} \in \mathcal{L}'$ has $B_v > 0$ and every set $A \in \mathcal{L}''$ has $f(A) \geq 1$.
(ii) $|\mathcal{L}| = |E|$.
(iii) The vectors χ_A, $A \in \mathcal{L}$, are linearly independent.
(iv) x is the unique solution to:
$\{x(\delta(v)) = B_v, \forall \{v\} \in \mathcal{L}'\} \bigcup \{x(\delta(A)) = f(A), \forall A \in \mathcal{L}''\}$.

PROOF. The proof follows from the uncrossing method, see Lemmas 4.1–4.3 of [18], or Chapter 52.4 of [33]. The main point is that if two tight sets A, B are properly intersecting then neither can be a singleton set, so the connectivity constraints for A, B must be holding with equality (the degree constraints are irrelevant); then either $A \cap B, A \cup B$ are tight and $\chi_A + \chi_B = \chi_{A\cap B} + \chi_{A\cup B}$ or $A - B, B - A$ are tight and $\chi_A + \chi_B = \chi_{A-B} + \chi_{B-A}$. □

The algorithm is now given in Figure 1. The following lemma is similar to the key lemma in [18].

LEMMA 2.3. *Let $\mathcal{L}$ be a laminar family of tight sets satisfying conditions (i)–(iv) in Lemma 2.2. Moreover, suppose that: 1) each $f(A)$ is an integer, for $A \subseteq V$, and 2) each vertex i that has a degree constraint has at least five incident edges with non-zero values. Then, in the unique solution x to the system, there is an edge e^* such that $x_{e^*} \geq \frac{1}{2}$.*

Now we give the sketch of the proof of Lemma 2.3. This is similar to the proof of key lemma in [18] as described in [35](Theorem 23.6). Let $\mathcal{L}$ be the laminar family of tight sets obtained in Lemma 2.2 when applied to the basic solution just before we execute line 2c. The number of sets in $\mathcal{L}$ is equal to the number of edges in G. We can view $\mathcal{L}$ as a forest of rooted trees where each vertex in the tree corresponds to a set in $\mathcal{L}$ and a root is a set not contained in any other set. Set T is the *parent* of S if it is the smallest set containing S. Following terminology of [35], S is said to *own* end-point v of edge $e = (u, v)$ if S is the smallest set containing v. A subtree owns e if one of the sets of $\mathcal{L}$ corresponding to the vertices in that subtree owns e. Note that there are a total of $2m$ end-points in G. The proof is established by showing that if every edge e has $x_e < 1/2$ then we can assign end-points to the sets in such a way that for every set S, S gets at least 3 end-points and each of its descendants gets 2 endpoints. We get a contradiction of having more than $2m$ end-points, once this argument is applied to the roots of the trees in the forest of laminar family. We need one more definition from [35]. For every set $S \in \mathcal{L}$ we define the *corequirement of* S as $\mathrm{coreq}(S) = \frac{1}{2}|\delta(S)| - f(S)$. The counting argument leading to a contradiction is done through the following lemma which is essentially Lemma 23.21 of [35].

LEMMA 2.4. *Let T be a subtree rooted at S and assume that $\forall e, x_e < 1/2$. The endpoints owned by T can be redistributed in such a way that S gets at least 3 endpoints and each of its descendants gets 2. Furthermore, if $\mathrm{coreq}(S) \neq 1/2$, then S gets at least 4 endpoints and if S is a degree constraint then it gets at least 5 endpoints.*

PROOF. First, note that the fractional-value tight sets are singletons coming from degree constraints. Each degree constraint is a leaf in the forest and each owns at least 5 endpoints by the assumption on x (line 2b). The same argument as in [35] shows that every other leaf (which is not a degree constraint) satisfies the requirements of the lemma. We say a set S has a surplus of p if $p + 2$ endpoints have been assigned to it. Consider a non-leaf set S.

(1) If S has two or more children, one of which is a degree constraint, then it can collect three endpoints from the surplus of its degree constraint child, and one endpoint from the surplus of one if its other children, for a total of at least 4.

(2) If S has only one child, say S', and it is a degree constraint, then since $\delta_G(S) \neq \delta_G(S')$ (by linear independence), S owns at least one endpoint. It can also collect 3 endpoints from the surplus of S', for a total of at least 4.

(3) If none of the children of S are degree constraints, then the same analysis as in [35] shows that S satisfies the requirements of the lemma. □

We are now ready to prove Theorem 2.1.

Proof of Theorem 2.1: The results of Jain [18] and Grötschel et al., [16, Theorem 6.4.9] show that a basic optimal solution x^* of the initial LP (if it exists) can be found in polynomial time. It can be seen that the updated function f' stays weakly supermodular (since we are subtracting a symmetric submodular function). It is clear that the final edge set F (at termination) satisfies all of the connectivity requirements.

We need to prove that the cost of F is $\leq 2z_{LP}$, where z_{LP} is the cost of x^* (the optimal solution of the initial LP) and that $\deg_F(i) \leq 2B_i + 3$ for all vertices i. We prove the former by induction on the number of iterations in which line 2c is executed. The base case is clear since we round up edges e with $x_e \geq 1/2$. The induction hypothesis is that the algorithm finds an edge set F' of cost $\leq 2z'_{LP}$ that satisfies the connectivity constraints where z'_{LP} is the solution to the current LP. Now, $z_{LP} \geq z'_{LP} + \frac{c_e}{2}$, because x^* restricted to the edges in $E - F$ satisfies all the constraints of current LP (this needs detailed verification for the connectivity constraints and for the degree bounds) and the cost of x^* restricted to the edges in $E - F$ is $\leq z_{LP} - \frac{c_e}{2}$, hence, $z'_{LP} \leq z_{LP} - \frac{c_e}{2}$. Hence, the cost of F is $c(F) = c(F') + c_e \leq 2z'_{LP} + c_e \leq 2z_{LP}$.

Finally we prove that for every vertex $i \in V$: $\deg_F(i) \leq 2B_i + 3$. Consider a degree constraint, say for vertex i, and focus on the last iteration in which B'_i changes or the constraint gets deleted in line 2b (so $B'_i > 0$ and we will not add any edge incident to i in future iterations). Suppose that we have added α edges incident to vertex i before this iteration. Since each edge added had value $\geq 1/2$: $B_i \geq \frac{\alpha}{2} + B'_i$. If in this iteration we add $\beta \geq 1$ more edges incident to i then the final degree of i will be $\alpha + \beta$ (by the assumption that this is the last such iteration); and we must have had $B'_i \geq \beta/2$ (because $x_e \geq 1/2$, we have $x_e \geq \lceil x_e \rceil/2$); therefore we have: $\alpha + \beta \leq 2B_i$. If we delete this constraint at this iteration (in line 2b) and solve the problem for this relaxed version, in the worst case, the final solution contains all the (at most) 4 remaining edges incident with vertex i. So the degree of i will be $\alpha + 4$ whereas B_i (the initial degree bound for i) is at least $\frac{\alpha}{2} + B'_i$, which implies that $\alpha + 4 \leq 2B_i + 4 - 2B'_i$, and since $\alpha + 4$ is an integer and $B'_i > 0$: $\alpha + 4 \leq 2B_i + 3$. □

Remark 1: One may wonder whether the bicriteria approximation guarantee of this theorem is best possible. The following example shows that the integrality gap of the LP is at least the minimum between $(2, 2B_v + 1)$ and $(2, B_v + 2)$. That is, if the LP is feasible and has an optimal solution with cost z_{LP}, then in any integral solution the cost is at least $2z_{LP}$, and each vertex v has degree at least $2B_v + 1$ or $B_v + 2$ (note that it is well-known that if we do not have degree bounds then the integrality gap is at least 2). Take a 3-regular 3-edge connected graph G with no Hamiltonian path.[2] Let $r_{ij} = 1$ for every pair of vertices in G and for all $i \in V$, let $B_i = 1$. Assigning $x_e = 1/3$ to every edge gives a feasible solution with cost $|V(G)|/2$ and degree bounds satisfied. It's not hard to see that this is also an optimal solution. On the other hand, any feasible integer solution with degree bounds at most 2 (which is $2B_i = B_i + 1$) needs to be a Hamiltonian path in G.

Remark 2: Our iterative rounding method applies also to the setting of minimizing the maximum degree subject to edge-connectivity constraints. We start with the above LP and introduce a new variable Δ, and replacing the degree constraints $x(\delta(i)) \le B_i, \forall i \in V$ by $x(\delta(i)) \le \Delta, \forall i \in V$. The objective function is to minimize Δ. Let (LP-Δ) denote this linear program. The following theorem follows immediately from Theorem 2.1, which implies the first constant factor approximation algorithm for many smallest maximum degree subgraph (satisfying connectivity requirements) problems.

THEOREM 2.5. *If (LP-Δ) has an optimal solution with objective value Δ^*, then there exists an integral solution $\hat{x}$ of maximum degree $\le 2\lceil\Delta^*\rceil + 3$ that satisfies all of the constraints on f. Moreover, $\hat{x}$ can be computed in polynomial time.*

Remark 3: For the average degree claim, notice that the number of edges in the support is at most $2n - 1$, since a basic feasible solution is characterized by a laminar family which has at most $2n - 1$ members. A naive argument (even if we take all the edges in the support) shows that the average degree bound is violated by an additive constant 4; this can be improved to 2 by a more careful argument, as follows.

Let's assume that $\tilde{B}$ is the average degree upper bound (i.e. $\tilde{B} = \frac{1}{n}\sum_{i\in V} B_i$). Then the arguments in the proof of Theorem 2.1 can also be used to show that in the final solution, the average degree of the vertices is at most $\tilde{B} + 2$; in other words, the degree of each vertex v in the final solution, on average, is at most $B_v + 2$ (i.e. the second part of Theorem 1.1). To prove this, we modify each iteration of the algorithm by adding the following line after line 2a and before line 2b:

(a') If there are any edges $e = (u, v)$ with $x_e \ge 1$ then add a copy of e to F; decrease U_e and the bounds for B'_u and B'_v by 1 and go to Step 2d.

It is easy to check that the same analysis shows that with this reformulation the cost of the solution is still at most $2z_{LP}$.

Consider the first iteration in which we have a totally fractional solution, i.e. $x_e < 1$ for all edges e. For each vertex v let $\alpha_v \ge 0$ be the number of edges incident with vertex v selected so far; thus $B'_v = B_v - \alpha_v$ because all the edges e selected so far had $x_e \ge 1$, and the degree bounds were decremented by 1.

CLAIM 2.6. *From now until the end of the algorithm, we select a total of at most $2n - 1$ other edges.*

[2] Such graphs exist. The following construction was brought to our attention by Jim Geelen and Jacques Verstraete. Let P denote the Petersen graph and $P - v$ be the graph obtained from it by deleting one vertex v, and let us call the neighbors of v in P as w_1, w_2, and w_3 (so these 3 vertices have degree 2 in $P - v$). Now, take 3 copies of $P - v$, and 3 new vertices v_1, v_2, and v_3, and attach them as follows: add edges from v_j to each of the 3 copies of w_j ($1 \le j \le 3$). It's not hard to argue that this graph on 30 vertices does not have any Hamiltonian path.

PROOF. By Lemma 2.2, the number of sets in $\mathcal{L}$ is equal to the number of edges (remaining) in the graph. Also, since the ground set has n vertices, an easy induction shows that the number of sets in $\mathcal{L}$ is at most $2n - 1$. Therefore the number of edges in G (with non-zero values) is at most $2n - 1$. □

Each time we select an edge e with $x_e \ge 1/2$, we increase the total degree by two (as it has two end-points), whereas the LP would increase the total degree by at least 1 (at least $1/2$ for each end-point). So we increase the total degrees by an extra (at most) 1 in every iteration with respect to the LP solution. Since there are at most $2n - 1$ iterations left (by the above claim), we increase the total degrees by an extra amount of at most $2n - 1$ (compared to LP), which is an average of at most 2 per vertex.

2.2 Directed Graphs

Our iterated rounding technique extends to directed graphs with some restricted types of connectivity requirements and degree bounds, via the results of Melkonian and Tardos [26].

For a set of vertices S, $\delta^{out}(S)$ denotes the set of arcs $\{ij \in E \mid i \in S, j \notin S\}$, and $\delta^{in}(S)$ denotes the set of arcs $\{ij \in E \mid i \notin S, j \in S\}$. An integer function on sets of vertices $f : 2^V \to \mathbb{Z}^+$ is called *crossing supermodular* if the inequality

$$f(A) + f(B) \le f(A \cap B) + f(A \cup B)$$

holds for every pair of sets $A, B \subseteq V$ such that $A \cap B \ne \emptyset$ and $A \cup B \ne V$. The connectivity requirement of "k-edge-connected spanning subgraph" can be formulated via the crossing supermodular function $f(S) = k,\ \forall \emptyset \ne S \subsetneq V$, but the connectivity requirement of "directed Steiner tree" cannot be so formulated. An integer function on sets of vertices $f : 2^V \to \mathbb{Z}^+$ is called *intersecting supermodular* if the inequality

$$f(A) + f(B) \le f(A \cap B) + f(A \cup B)$$

holds for every pair of sets $A, B \subseteq V$ such that $A \cap B \ne \emptyset$. This is a stronger requirement than crossing supermodularity; for example the connectivity requirements of a strongly k-edge-connected subgraph cannot be formulated as an intersecting connectivity requirement function. An example of an intersecting supermodular function is the connectivity requirement of an arborescence. Given a directed graph $G = (V, E)$ and a specific vertex $r \in V$ called the root, a subgraph $H = (V, E')$ of G is called an r-arborescence if there is exactly one directed path from r to every other vertex. In other words, it is a spanning tree rooted at r. Another example of an intersecting supermodular connectivity requirement function comes from the rooted k-edge-connected subgraph problem.

First we address the problem of finding a minimum cost subgraph satisfying intersecting supermodular connectivity requirements and non-uniform degree requirements (both in-degrees and out-degrees). In the following the connectivity requirements are specified by an intersecting supermodular function f. Furthermore, for simplicity, we assume that the connectivity requirement function comes from the rooted k-edge-connected subgraph problem. Figure **??** is the LP-relaxation for our problem (as before U_e is the upper bound on the multiplicity of edge e):

THEOREM 2.7. *If the above LP (for directed graphs) has an optimal solution of cost z_{LP}, then there exists an integral solution $\hat{x}$ of cost $\le 4z_{LP}$ that satisfies all of the constraints on f if f is intersecting supermodular and $x(\delta^{out}(i)) \le 4B_i^{out} + 6$ and $x(\delta^{in}(i)) \le 4B_i^{in} + 6$ for all $i \in V$. Moreover, $\hat{x}$ can be computed in polynomial time.*

$$
\begin{aligned}
\text{(DLP) minimize} \quad z_{DLP} &= \sum_{e\in E} c_e x_e \\
\text{subject to} \quad \sum_{e\in\delta^{in}(S)} x_e &\geq f(S), && \forall S\subseteq V, r\notin S \\
\sum_{e\in\delta^{out}(i)} x_e &\leq B_i^{out}, && \forall i\in V \\
\sum_{e\in\delta^{in}(i)} x_e &\leq B_i^{in}, && \forall i\in V, i\neq r \\
0 \leq x_e &\leq U_e, && \forall e\in E
\end{aligned}
$$

Figure 2: LP for directed case

The proof of this theorem follows from an extension of the methods of Jain and of Melkonian & Tardos [18, 26], similar to our proof of Theorem 2.1

We say that a pair of sets A, B are *crossing* if all of the sets $A\cap B, A-B, B-A, V-(A\cup B)$ are nonempty, and we say that a family of sets $\mathcal{L} = \{A_1, A_2, \ldots, A_\ell\}$ is *cross-free* if no two of its sets are crossing. For any set $A\subseteq V$, let χ_A denote the incidence vector of the set of arcs $\delta^{out}(A)$; note that in the constraints matrix of the LP, we rewrite the out-degree constraints $x(\delta^{out}(i)) \leq B_i^{out}$ as $x(\delta^{in}(V-\{i\})) \leq B_i^{out}$, and that χ_A is the row for the set A (the constraint for A may be either a connectivity constraint or a degree bound). Consider any solution x of the LP. We call a set of vertices A *tight* (w.r.t. x) if either $A=\{i\}$ or $A=V-\{i\}$ and x satisfies the degree constraint for i with equality, $x(\delta^{out}(i)) = B_i^{out}$ or $x(\delta^{in}(i)) = B_i^{in}$, or x satisfies the connectivity constraint for A with equality, $x(\delta^{out}(A)) = f(A)$ (in the latter case, A may be a singleton or not). The following lemma follows from arguments similar to those of proof of Lemma 3 in [26].

LEMMA 2.8. *Let the requirement function f of (DLP) be intersecting supermodular, and let x be a basic solution of (LP) such that $0 < x_e < 1$ for all edges $e\in E$. Then there exists a cross-free family $\mathcal{Q}$ of tight sets such that $\mathcal{Q}$ partitions into a set of singletons or complements of singletons $\mathcal{Q}'$ for the degree constraints, and the remaining sets $\mathcal{Q}'' = \mathcal{Q}-\mathcal{Q}'$ for the connectivity constraints form a laminar family, such that:*
(i) Every set $A=\{i\}\in\mathcal{Q}'$ has $B_i^{in}\geq 1$, every set $A=V-\{i\}\in\mathcal{Q}'$ has $B_i^{out}\geq 1$, and every set $A\in\mathcal{Q}''$ has $f(A)\geq 1$.
(ii) $|\mathcal{Q}| = |E|$.
(iii) The vectors χ_A, $A\in\mathcal{Q}$, are linearly independent.
(iv) x is the unique solution to $\{x(\delta^{in}(i)) = B_i^{in}, \forall\{i\}\in\mathcal{Q}'\} \bigcup \{x(\delta^{out}(i)) = B_i^{out}, \forall V-\{i\}\in\mathcal{Q}'\} \bigcup \{x(\delta^{in}(A)) = f(A), \forall A\in\mathcal{Q}''\}$.

LEMMA 2.9. *[26] Let $\mathcal{Q}$ be a cross-free family of tight sets satisfying the conditions in Lemma 2.8. Moreover, suppose that: 1) each $f(S)\geq 1$ is an integer, for $S\in\mathcal{Q}$, and 2) each vertex that has a in-degree constraint has at least eight in-going edges with non-zero values and each vertex that has an out-degree constraint has at least eight out-going edges with non-zero values. Then in the unique solution x to the system, there is an edge e^* such that $x_{e^*}\geq\frac{1}{4}$.*

This lemma is the same as the key lemma in [26]; Our rounding algorithm is very similar to that of Theorems 2.1: The only difference is that we delete degree constraints (in-degree/out-degree) for vertices with at most seven incident edges (in going/out-going respectively). By the above lemmas, there is at least one edge e with $x_e^*\geq\frac{1}{4}$. We add one such edge e to our solution edge set and update the degree bounds and function f accordingly. An easy argument similar to that of Theorem 2.1 shows that the cost of the solution is at most $4z_{LP}$ and that for each vertex $v\in V$ the final out-degree (in-degree) of v is at most $4B_v^{out}+6$ $(4B_v^{in}+6)$. We leave the proof of Lemma 2.9 to the full version of the paper.

For the second claim of Theorem 1.3, we use a technique in [26] (which in turn is inspired by Frank [11]) to decompose a crossing supermodular connectivity requirement function into two intersecting supermodular connectivity requirement functions (see [26]). For example, the strongly k-edge-connected subgraph problem can be decomposed into two rooted k-edge-connected subgraph problems, and similarly for the bounded degree version. So, an (a, b, c)-approximation algorithm for the latter problem immediately gives a $(2a, 2b, 2c)$-approximation algorithm for the former problem. This shows the second claim of Theorem 1.3 where the connectivity requirement function comes from the strongly k-edge-connected subgraph problem. In fact, we believe that the approximation ratios in Theorem 1.3 can be improved (details to appear in the journal version).

2.3 Hardness of Low Degree Subset k-Vertex Connected Subgraph

In this subsection, we show that unlike the degree bounded EC-SNDP for which we presented a $(2, 2B_v+3)$-bicriteria approximation algorithm, the vertex-connectivity version, which we call degree bounded VC-SNDP is very hard to approximate. In the VC-SNDP we are given a weighted undirected graph $G=(V,E)$ with degree bound B_v for every $v\in V$, and a connectivity requirement $r: V\times V\rightarrow\mathbb{Z}^+$. We want to find a minimum cost subgraph G' satisfying the connectivity requirements and the degree bounds. As we will see, it is hard to get an $(\infty, 2^{\log^{1-\epsilon} n}\cdot B_v)$-approximation for this problem. In other words, even if all edge costs are zero and we just have to approximate the degree bounds it is still hard. In fact the same hardness holds for a more special case of the problem, called degree-bounded subset k-vertex connected subgraph (DkVC for short) in which $r_{uv}=k$ for every pair $u,v\in S$ for some set $S\subseteq V$ and $r_{uv}=0$ otherwise. An α-approximation for DkVC will find a solution G' in which the degree of every vertex v is at most αB_v and there are r_{uv} vertex-disjoint paths between every pair $u,v\in V$. The following theorem immediately implies Theorem 1.4.

THEOREM 2.10. *Unless NP $\subseteq$ DTIME($n^{polylog(n)}$) there is no $2^{\log^{1-\epsilon} n}$-approximation for DkVC for some $\epsilon > 0$.*

We have a similar hardness result for the Low Degree Directed Steiner Forest (LDSF) problem. In LDDSF, we are given a directed graph $G=(V,E)$, degree bounds B_v for every $v\in V$, and connectivity requirements $r: V\times V\rightarrow\{0,1\}$. The goal is to find smallest $\alpha\geq 1$ and a subgraph G' satisfying the connectivity requirements in which the degree of each vertex v is at most αB_v. The proofs of Theorems 2.10 and 2.11 follow from the construction for the hardness of vertex-connectivity version of survivable network design problem (SNDP) and subset connectivity [23]. Details appear in the full version of the paper.

THEOREM 2.11. *Unless NP $\subseteq$ DTIME($n^{polylog(n)}$) there is no $2^{\log^{1-\epsilon} n}$-approximation for LDSF for some $\epsilon > 0$.*

3. MINIMUM COST λ-CONNECTED K-SUBGRAPHS

In this section we focus on the following class of problems. Given are a (multi)graph $G(V, E)$ with edge costs $c : E \rightarrow \mathbb{R}^+$, and positive integers k and connectivity requirement $\lambda \geq 1$; the (k, λ)-subgraph problem asks to find a minimum cost λ-edge-connected subgraph of G with at least k vertices. We should point out that edge costs induce an arbitrary function. Furthermore, we are not allowed to take more copies of an edge than are present in the graph. Otherwise, a 2-approximate solution can be computed by taking a 2-approximate k-MST solution T, and then taking λ copies of T.

Note that the (k, λ)-subgraph problem contains, as special cases, several classical problems. For instance, the minimum cost λ-edge-connected spanning subgraph problem is just the minimum (n, λ)-subgraph, and the classical k-MST problem is the $(k, 1)$-edge-subgraph problem. Another related and well-studied problem is that of k-TSP (finding a minimum cost traveling salesman tour visiting at least k vertices) for the metric cost functions. Although there are approximation algorithms for each of these special cases, we are not aware of any study of the more general problem of (k, λ)-subgraph. As we will see below, it seems that this problem for arbitrary values of λ (and even unweighted graphs) is very difficult to approximate.

For this reason, we look into the approximability of the $(k, 2)$-subgraph, which is the first generalization of k-MST to higher connectivity. We show that $(k, 2)$-subgraph has an $O(\log^3 k)$-approximation. This algorithm is based on the results of [1, 17]. It works for the rooted version of the problem where a particular vertex $r \in V$ is required to be in the solution. It is easy to see that given an algorithm for the rooted version, we can try all possible vertices as the root to obtain an algorithm for the unrooted version.

THEOREM 3.1. *There is an $O(\log^3 k)$-approximation algorithm for the rooted $(k, 2)$-subgraph problem.*

As mentioned earlier, we show that for arbitrary values of λ, the (k, λ)-subgraph problem seems to be very difficult. As evidence, we show a reduction from the k-dense-subgraph problem. In the k-dense-subgraph problem we are given a graph G and integer k and have to find a subgraph with k vertices with maximum number of induced edges. Despite considerable effort, the best known approximation algorithm for the k-dense-subgraph problem has ratio $O(n^{\frac{1}{3}-\epsilon})$ for some constant $\epsilon > 0$ [10]. We can prove that:

THEOREM 3.2. *If there is an α-approximation algorithm for (k, λ)-subgraph for arbitrary λ, even for unweighted graphs, then there is an $(\alpha \cdot \log^2 k)$-approximation for the k-dense subgraph problem.*

Therefore, obtaining any poly-logarithmic approximation for the (k, λ)-subgraph problem would imply a poly-logarithmic approximation for the k-dense subgraph problem. Proof of Theorem 3.2 appears in the full version of this paper.

3.1 Proof of Theorem 3.1

Recall that an instance $\mathcal{I}$ to the rooted $(k, 2)$-subgraph has a graph $G = (V, E)$, parameter k, and a root $r \in V$. Our algorithm, which is based on [1, 17], has a key procedure, which we call it Partial. This procedure tries to find a 2-edge-connected subgraph on at least $\frac{k}{4}$ vertices that contains the root and whose cost is at most $O(\log^2 k)$-factor of the optimum solution to instance $\mathcal{I}$. Then, by running this procedure at most $O(\log k)$ rounds, we find a $(k, 2)$-subgraph of cost at most $O(\log^3 k)$ of the optimum solution to $\mathcal{I}$.

Procedure Partial is a Kruskal-like algorithm. At any given time during this procedure, we have a set of 2-edge-connected components which we call *clusters*. We start with every vertex as a single cluster and at each iteration we try to connect two or more clusters. Once there is at least one cluster that contains at least $\frac{k}{4}$ vertices the procedure stops and returns that cluster. The main step of this procedure is to find the set of clusters that have to be merged. We always look for clusters that have about the same size (more specifically, factor at most two apart) to find a group of size at least two that are to be merged. To simplify the algorithm, at each iteration we contract each cluster C_i into a single vertex and use c_i to refer to that (contracted) cluster. The multiplicity of that vertex is the number of vertices contracted into that vertex.

Procedure Partial

1. Let every $v_i \in V$ be a single cluster C_i with multiplicity one.
2. $p \leftarrow 0$
3. For every C_i and every s_i satisfying $s_i \leq |C_i| \leq 2s_i$ do
   (a) Let S_i be the set of clusters (vertices) with multiplicity between s_i and $2s_i$.
   (b) Find a cluster $C_j \in S_i$ with the smallest ratio of $d_2(C_i, C_j)/s_i$; $p \leftarrow p + 1$; Let $M_p = \{C_i, C_j\}$ be a candidate set, its edge set F_p be the edges of the two disjoint paths found to compute $d_2(C_i, C_j)$, and its ratio be $d_2(C_i, C_j)/s_i$.
   (c) Find a cycle D containing (vertex) c_i with the smallest ratio of $\sum_{e \in D} c(e)/(s_i \cdot |D \cap (S_i - C_i)|)$; $p \leftarrow p + 1$; Let M_p be the set of clusters of S_i on cycle D (including C_i), F_p be the set of edges of this cycle, and r_p be its ratio.
4. Among all p candidate sets, let M_q ($1 \leq q \leq p$) be the one with the smallest ratio r_q.
5. Merge all the clusters in M_q by adding the edges in F_q. Contract the new cluster into a single vertex and assign its multiplicity as the sum of the multiplicities of all the vertices contracted.
6. Goto Step 3 unless there is a cluster of size (multiplicity) at least $\frac{k}{4}$. Return the largest cluster.

Figure 3: The main subroutine used in the algorithm for $(k, 2)$-subgraph problem

For any two clusters C_i and C_j we define $d_2(C_i, C_j)$ as the minimum cost of two edge-disjoint paths that run between these two clusters. To compute $d_2(C_i, C_j)$ we use a min-cost flow algorithm between c_i and c_j [33]. Consider a fixed cluster C_i and let S_i be the set of clusters of size (multiplicity) between s_i and $2s_i$ where $s_i \leq |C_i| \leq 2s_i$ (we consider all possible values of s_i that satisfy this inequality, and therefore the corresponding set S_i, separately). First, we consider all clusters $C_j \in S_i$ and compute the minimum ratio $r_1 = d_2(C_i, C_j)/s_i$. So far $\{C_i, C_j\}$ is one candidate for the merge (using the two disjoint paths found between them). This set has ratio r_1. Next we compute other sets of clusters as candidates for merge with C_i. To do so, we compute a minimum ratio cycle containing c_i (again, we work with contracted clusters), where the ratio of a cycle is equal to the cost of its edges divided by s_i times the number of vertices corresponding to the clusters from $S_i - C_i$ in that cycle. We later show how to compute a minimum ratio cycle. Every such cycle defines a set of clusters from S_i (including C_i) as a candidate whose ratio is the ratio of the cycle (as defined).

The Main $(k, 2)$-subgraph Algorithm

1. Guess the value of optimum solution; let it be μ.
2. Delete all the vertices v with $d_2(v, r) > \mu$ as they cannot belong to OPT (r is the root).
3. While $k > 0$ do
   (a) Run Partial with parameter k; let C_i be the largest cluster found and $s = |C_i| \geq 2$.
   (b) Connect C_i to the root using two edge-disjoint paths P_1 and P_2 with total cost at most μ. Let p be the total number of vertices in $C_i \cup P_1 \cup P_2$.
   (c) Contract C_i and these two paths into the root and set $k = k - p + 1$.
4. Uncontract all the clusters contracted in the root (and their paths) and return it.

Figure 4: The algorithm for $(k, 2)$-subgraph problem

We do this for all values of s_i (that satisfy $s_i \leq |C_i| \leq 2s_i$) and also for all clusters C_i. Among all candidate sets found, take the one with the smallest ratio and merge the clusters in that set. The full description of procedure Partial is given in Figure 2

Here are more details for step 3c. First subdivide every edge (temporarily) by adding a new vertex into it and replacing it by a path of length 2. For every vertex c_j corresponding to a cluster $C_j \in S_i - C_i$, let all edges e of that vertex have $w(e) = 1$. Every other edge e, including the ones incident to vertex c_i (for cluster C_i), have $w(e) = 0$. Then we compute a cycle D containing vertex c_i with minimum ratio $\left(\sum_{e\in D} c(e)\right) / (s_i \sum_{e\in D} w(e))$. This can be done using the min-ratio cycle algorithms [25]. We prove the following upper bound on the cost of the solution returned by Partial. Let OPT be an optimum solution and μ be its cost.

LEMMA 3.3. *The cost of the solution returned by Partial is at most $O(\mu \log^2 k)$.*

First we show how using this lemma we obtain an $O(\log^3 k)$-approximation for $(k, 2)$-subgraph. We guess the cost of the optimum solution, μ (we can do binary search for μ, or as we explain later do it in $O(\log k)$ iterations). The main algorithm is given in Figure 3. The following lemma follows from the algorithm and Lemma 3.3.

LEMMA 3.4. *The cost of the solution returned by the Main Algorithm is at most $O(\mu \log^3 k)$.*

To guess the value μ, for every vertex $v \neq r$, first we compute $d_2(r, v)$. Let L be the kth smallest value. Clearly, $L \leq \mu \leq kL$. So it is enough to start with estimate $\mu = L$ and then double the estimate of μ if the algorithm does not succeed; we have to do this at most $O(\log k)$ times. To prove Lemma 3.3 we need the following two claims. These two claims are the heart of the proof of correctness, specially Claim 3.6. We defer the proofs to the full version of the paper.

CLAIM 3.5. *If the largest ratio used by procedure Partial so far is at most r then every cluster with p vertices has cost at most $3rp \log p$.*

CLAIM 3.6. *Procedure partial never merges the clusters of a candidate set with ratio larger than $(24\mu \log k)/k$.*

PROOF. By way of contradiction, suppose that all the clusters found so far have size smaller than $\frac{k}{4}$ and the smallest ratio that the procedure finds is larger than $r = 24\mu \log k/k$. We group the clusters into at most $\log k$ buckets by placing clusters of size between $k/2^i$ and $k/2^{i+1}$ into bucket i, for $i \geq 2$. Ignore all the clusters that do not intersect the optimum solution. If we consider buckets that have exactly one cluster inside, then the number of vertices inside them is at most $\frac{k}{4} + \frac{k}{8} +, \ldots < \frac{k}{2}$. Thus, there are at least $\frac{k}{2}$ vertices of the optimum that are in the buckets with at least two clusters each. So there is at least one such bucket, say bucket j, that contains at least $\frac{k}{2 \log k}$ vertices of the optimum and all the clusters of this bucket have size between $s = k/2^{j+1}$ and $2s = k/2^j$. Thus, the optimum intersects at least $\ell = \frac{k}{4s \log k} \geq 2$ clusters in this bucket. Take the optimum solution and contract each of these clusters into a single vertex and call this set of vertices $P = \{p_1, \ldots, p_\ell\}$. Now in the optimum solution, make a parallel copy of every edge to obtain an Eulerian graph, call it G'. We call two cycles that meet in exactly one vertex a bi-cycle. Our goal is to obtain an edge-disjoint collection $\mathcal{T}$ of cycles and bi-cycles of G' with the following properties: (i) each bi-cycle in $\mathcal{T}$ is marked to exactly two and each cycle in $\mathcal{T}$ is marked to at least two vertices of P such that each $D \in \mathcal{T}$ contains the vertices of P marked to it, (ii) no vertex of P is marked by more than one element of $\mathcal{T}$, and (iii) at least $\ell/3$ vertices of P are marked by some element of $\mathcal{T}$. To construct $\mathcal{T}$, we start with $\mathcal{T} = \emptyset$ and every vertex of P is unmarked. Take an Eulerian tour of G', starting from vertex $p_1 \in P$. Each time we visit a vertex put that vertex and the edge we traversed on the stack. The first time we re-visit a vertex v, we look at all the vertices and the edges on the stack until the most recent copy of v on the stack:

1. If there are no unmarked vertices of P in this set then we pop all the vertices and edges from the stack and discard them; continue with the Euler tour.
2. If there are at least two unmarked vertices of P in this set we have found a cycle containing at least two unmarked vertices of P. We place this cycle in $\mathcal{T}$, mark those unmarked vertices of P in the cycle, and pop all the edges and vertices until the most recent copy of v on the stack; continue with the Euler tour.
3. If there is exactly one unmarked vertex from P then we have found a cycle with exactly one unmarked vertex of P. We continue with the tour until we either find another cycle containing v and exactly one other unmarked vertex of P (in which case we place this bi-cycle in $\mathcal{T}$, mark those two vertices of P with this bi-cycle, and pop all its edges and vertices from the stack) or we find a cycle with at least two unmarked vertices of P; in the latter case we ignore the first cycle found with only one unmarked vertex of P and continue.

Each time (except possibly the very last iteration) we ignore a cycle containing exactly one unmarked vertex of P (in step 3 above) in the next round we mark at least two vertices from P. So for every vertex left unmarked in P there are at least 2 vertices marked. It is easy to see that at the end there are at least two marked vertices in P if $\ell \leq 5$ and at least $2\ell/3 - 1$ marked vertices in P if $\ell \geq 6$. In either case we have at least $\ell/3$ marked vertices. Now by the assumption, the ratio of each cycle and bi-cycle in $\mathcal{T}$ is larger than r. This implies for every cycle $D \in \mathcal{T}$: $\sum_{e\in D} c(e)/(s \cdot |D \cap P|) > r$ and for every bi-cycle $B \in \mathcal{T}$: $\sum_{e\in B} c(e)/s > r$. Thus

$$\frac{\sum_{\text{cycle D}\in\mathcal{T}} \sum_{e\in D} c(e) + \sum_{\text{bi-cycle B}\in\mathcal{T}} \sum_{e\in B} c(e)}{\sum_{\text{cycle D}\in\mathcal{T}} (s \cdot |D \cap P|) + \sum_{\text{bi-cycle B}\in\mathcal{T}} s} > r.$$

By property (iii) for set $\mathcal{T}$, the sum in the denominator is at least $s\ell/3$. This implies:

$$\begin{aligned} 2\mu &\geq \sum_{\text{cycle D}\in\mathcal{T}} \sum_{e\in D} c(e) + \sum_{\text{bi-cycle B}\in\mathcal{T}} \sum_{e\in B} c(e) > sr\ell/3 \\ &= s \cdot \frac{24\mu \log k}{k} \cdot \frac{k}{12 s \log k} \\ &= 2\mu, \end{aligned}$$

which is a contradiction. $\square$

4. ACKNOWLEDGMENTS

Many results in this paper were obtained in collaboration with Joseph Cheriyan. In particular, we thank him for allowing us to put the proof of Theorem 1.2 in this submission.

5. REFERENCES

[1] B. Awerbuch, Y. Azar, A. Blum and S. Vempala, *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Computing 28(1):254-262, 1999.

[2] A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k-MST problem*, J. Comput. Syst. Sci. 58(1): 101-108, 1999.

[3] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar, *What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs*, In Proc. of APPROX 2005, pp. 26-39.

[4] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. *Push relabel and an improved approximation algorithm for the bounded-degree mst problem* , In Proc. of ICALP 2006.

[5] J.Cheriyan, Santosh Vempala and Adrian Vetta, *Network design via iterative rounding of setpair relaxations,* Combinatorica 26(3):255-275, (2006).

[6] T. Feder, R. Motwani, and A. Zhu, *k-Connected spanning subgraphs of low degree*, ECCC report 41, 2006.

[7] L. Fleischer, K. Jain and D.P. Williamson, *Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems.*, J. Comput. Syst. Sci. 72(5):838-867, 2006.

[8] M. Fürer and B. Raghavachari, *An NC approximation algorithm for the minimum degree spanning tree problem*, In Proc. of the 28th Annual Allerton Conf. on Commun., Control and Computing, pages 274–281, 1990.

[9] M. Fürer and B. Raghavachari, *Approximating the minimum-degree Steiner tree to within one of optimal*, J. of Algorithms 17(3):409-423, 1994.

[10] U. Feige, G. Kortsarz and D. Peleg, *The Dense k-subgraph problem*, Algorithmica, 29(3): 410-421, 2001. Preliminary version in the Proc. 34-th IEEE FOCS pp 692-701, 1993.

[11] A. Frank, *Kernel Systems of Directed Graphs*, Acta Sci. Math (Szeged) 41:63-76, 1979.

[12] N. Garg, *A 3-Approximation for the minim tree spanning k vertices*, In Proc. of the 37th IEEE FOCS, 302-309, 1996.

[13] N. Garg, *Saving an epsilon: a 2-approximation for the k-MST problem in graphs*, In Proc. of the 37th STOC, 396 - 402, 2005.

[14] M.X. Goemans, *Minimum Bounded-Degree Spanning Trees,* Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 273–282.

[15] M. Goemans and D. Williamson, *A general Approximation Technique for Constrained Forest Problems*, SIAM J. on Computing, 24:296-317, 1995.

[16] M.Grötschel, L.Lovász and A.Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, New York, 1988.

[17] M. Hajiaghayi, G. Kortsarz and M. Salavatipour, *Approximating Buy-at-Bulk and Shallow-light k-Steiner trees*, In proceedings of APPROX 2006.

[18] K.Jain, *A factor 2 approximation algorithm for the generalized Steiner network problem*, Combinatorica, 21:39-60, 2001.

[19] P. Klein, R. Krishan, B. Raghavachari, and R. Ravi, *Approximation algorithms for finding low-degree subgraphs*, Networks, 44(3): 203-215, (2004).

[20] J. Könemann and R. Ravi, *Quasi-polynomial time approximation algorithm for low-degree minimum-cost steiner trees.* In Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, 2003.

[21] J. Könemann and R. Ravi, *A matter of degree: Improved approximation algorithms for degree bounded minimum spanning trees*, SIAM J. on Computing, 31:1783-1793, 2002.

[22] J. Könemann and R. Ravi, *Primal-Dual meets local search: approximating MSTs with nonuniform degree bounds*, SIAM J. on Computing, 34(3):763-773, 2005.

[23] G. Kortsarz, R. Krauthgamer, and J. Lee, *Hardness of approximation for vertex-connectivity network design problems*, SIAM J. on Computing 33(3):704–720, 2003.

[24] G. Kortsarz and Z. Nutov, *Approximating min-cost connectivity problems*, Survey Chapter in Approximation Algorithms and Metaheuristics, Editor T.F. Gonzalez, 2006.

[25] N. Megiddo, *Combinatorial Optimization with Rational Objective Functions*, Mathematics of Operation Research, 4(4):414-424, 1979.

[26] V. Melkonian and E. Tardos, *Algorithms for a Network Design Problem with Crossing Supermodular Demands*, Networks, 43(4):256-265, 2004.

[27] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrants, and H.B. Hunt, *Many birds with one stone: Multi-objective approximation algorithms*, in Proc. of the 25th STOC, pp 438-447, 1993.

[28] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrants, and S.S. Ravi, *Spanning trees short or small*, SIAM Journal on Discrete Mathematics, 9(2):178-200, 1996.

[29] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrants, and H.B. Hunt, *Approximation Algorithms for degree-constrained minimum-cost network-design problems*, Algorithmica 31:58-78, 2001.

[30] R. Ravi, B. Raghavachari, and P. Klein, *Approximation through local optimality: designing networks with small degree*, In Proc. of the 12 Conference on Foundations of Sofware Tech. and Theoret. Comput. Sci., LNCS 652, pp 279-290, 1992.

[31] R. Ravi and M. Singh, *Delegate and conquer: An LP-based approximation algorithm for minimum degree MSTs.* In Proc. of ICALP 2006.

[32] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency, Springer-Verlag, New York, 2005. Approximation

[33] M. Singh, L.C. Lau, *Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal*, In Proc. of the 39th STOC, 2007.

[34] V. Vazirani, Approximation Algorithms, Springer, 2001.