Survivable Network Design with Degree or Order Constraints*

Lap Chi Lau† Joseph (Seffi) Naor‡ Mohammad R. Salavatipour§ Mohit Singh¶

Abstract

We present algorithmic and hardness results for network design problems with degree or order constraints. The first problem we consider is the SURVIVABLE NETWORK DESIGN problem with degree constraints on vertices. The objective is to find a minimum cost subgraph which satisfies connectivity requirements between vertices and also degree upper bounds B_v on the vertices. This includes the well-studied MINIMUM BOUNDED DEGREE SPANNING TREE problem as a special case. Our main result is a $(2, 2B_v + 3)$-approximation algorithm for the edge-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints, where the cost of the returned solution is at most twice the cost of an optimum solution (satisfying the degree bounds) and the degree of each vertex v is at most $2B_v + 3$. This implies the first constant factor (bicriteria) approximation algorithms for many degree constrained network design problems, including the MINIMUM BOUNDED DEGREE STEINER FOREST problem. Our results also extend to directed graphs and provide the first constant factor (bicriteria) approximation algorithms for the MINIMUM BOUNDED DEGREE ARBORESCENCE problem and the MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH problem. In contrast, we show that the vertex-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints is hard to approximate, even when the cost of every edge is zero. A striking aspect of our algorithmic results is its simplicity. It is based on the *iterative relaxation* method, which is an extension of Jain's iterative rounding method. This provides an elegant and unifying algorithmic framework for a broad range of network design problems.

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 provide evidence that the (k, λ)-subgraph problem might be hard to approximate for arbitrary λ.

1 Introduction

Network design is a central topic in combinatorial optimization, approximation algorithms, and operations research. The basic setting of network design problems is to find a minimum cost subgraph satisfying connectivity requirements between vertices. This captures a variety of classical problems such as MINIMUM COST FLOW problem, MINIMUM STEINER TREE problem, HAMILTONIAN CYCLE problem, etc. Also, research results in this area provide algorithmic tools and insights (e.g., hardness results) for the design of practical networks such

*A preliminary version appeared in the Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), San Diego, CA (2007), pp. 651-660.

†Department of Computer Science and Engineering, The Chinese University of Hong Kong. Partially supported by RGC Earmark Research Grant 412907.

‡Computer Science Dept., Technion, Haifa 32000, Israel. Part of this work was done while visiting Microsoft Research, Redmond, WA. Supported in part by ISF grant 1366/07 and BSF grant 2002276. E-mail: naor@cs.technion.ac.il

§Department of Computing Science, University of Alberta, Edmonton, Alberta T6G2E8, Canada. Supported by NSERC and a start-up grant from the University of Alberta. E-mail: mreza@cs.ualberta.ca

¶Tepper School of Business, Carnegie Mellon University. Supported by NSF ITR grant CCR-0122581 (The ALADDIN project) and NSF grant CCF- 0430751. Email: mohits@andrew.cmu.edu.

as telecommunication networks. A highlight of this line of research is Jain's 2-approximation algorithm for the SURVIVABLE NETWORK DESIGN problem [23].

A recent research direction 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. Recently, Goemans [19] obtained an approximation algorithm for this problem, with only an additive error of two on the degrees, following a long line of research. We observe that the basis underlying the breakthrough results of Jain [23] and Goemans [19] 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 edge-connectivity SURVIVABLE NETWORK DESIGN problem that simultaneously provides near-optimal bounds on the degrees. This yields the first constant factor (bicriteria) approximation algorithms for many degree-constrained network design problems, including the MINIMUM BOUNDED DEGREE STEINER NETWORK problem and the MINIMUM BOUNDED DEGREE STEINER TREE problem. Our results extend to directed graphs and provide the first constant factor (bicriteria) approximation algorithms for MINIMUM BOUNDED DEGREE ARBORESCENCE and MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH. A striking aspect of our method is its *simplicity*. Our approach is based on the *iterative relaxation* method, which is an extension of Jain's *iterative rounding* method. This provides an elegant and unifying algorithmic framework for a broad range of network design problems. Very recently, the iterative relaxation technique introduced in this paper has been extended to give an $(1, B_v + 1)$-approximation algorithm for the MINIMUM BOUNDED DEGREE SPANNING TREE problem [43], settling a conjecture of Goemans affirmatively. In contrast, we present hardness results for the vertex-connectivity SURVIVABLE NETWORK DESIGN problem with degree constraints, even when 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

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 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 [23] gave a 2-approximation algorithm for the edge-connectivity SURVIVABLE NETWORK DESIGN problem by using an elegant iterative rounding approach. In contrast, the vertex-connectivity SURVIVABLE NETWORK DESIGN problem is shown to be very hard to approximate [29].

Network design problems with degree constraints have been studied extensively lately. A simpler setting is to minimize the maximum degree of a subgraph (without considering the cost) satisfying given 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 [12, 13] gave an approximation algorithm returning a solution with maximum degree at most one off the optimal solution. (Their result holds for the

Connectivity Requirement	Degree	This Paper	Cost and Degree	This Paper
Spanning Tree	$B+1$ [13]	$2B+3$	$(1, B+2)$ [19]	$(2, 2B+3)$
Steiner Tree	$B+1$ [13]	$2B+3$	$(O(\log n), O(B\log n))$ [36]	$(2, 2B+3)$
Steiner Forest	-	$2B+3$	-	$(2, 2B+3)$
k-Edge-Connected Subgraph	$O(k\log n)$ [11]	$2B+3$	-	$(2, 2B+3)$
Steiner Network	-	$2B+3$	-	$(2, 2B+3)$

Table 1: Results on Minimum Cost Bounded-Degree Network Design Problems.

Steiner tree problem as well.) Ravi, Raghavachari, and Klein [37, 25] 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 [11] obtained a polynomial time $O(k\log n)$-approximation algorithm for this problem, for any *fixed* k, thus answering an open question in [37]. Our main result implies the first constant factor approximation algorithm even for general edge-connectivity requirements.

For the 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 which satisfies all the degree bounds. We say an algorithm is an $(\alpha, f(B_v))$-approximation algorithm if the returned solution has cost at most $\alpha \cdot \text{OPT}$ and the degree at each vertex v is at most $f(B_v)$. The first approximation algorithm for the MBDST problem was an $(O(\log n), O(B_v \log n))$-algorithm in [34, 36]. This was subsequently improved in a series of papers [27, 28, 7, 8, 38]. Recently, Goemans [19] made a major step on this problem by giving a $(1, B_v+2)$-approximation algorithm. The proof of Goemans' result is based on the uncrossing technique, and is considerably simpler than the previous results. Not much 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 [36]. This bound was improved to $(O(1), O(B_v + \log n))$-approximation by [26], but the algorithm runs in quasi-polynomial time. Our main result implies the first polynomial time $(2, 2B_v+3)$-approximation algorithm even for general 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 [35, 3] down to constant in [6, 17] and very recently to 2 [18]. 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 [18].

1.2 Our Results

Suppose that we are given an undirected graph with connectivity requirements r_{uv} on pairs of vertices u and v, and a degree bound B_v on each vertex v. The 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. Our main result is the following theorem.

Theorem 1.1 *There is a polynomial time $(2, 2B_v+3)$-approximation algorithm for the* SURVIVABLE NETWORK DESIGN *problem with degree constraints. 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, and the MINIMUM k-EDGE-CONNECTED SUBGRAPH problem. It also implies the first constant factor approximation algorithm for the minimum maximum-degree version of many problems (by setting $B_v = B$ for all v). We remark that Theorem 1.1 holds 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 not 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 both indegree and outdegree constraints (B_v^{in} and B_v^{out}, respectively, for each $v \in V$). This includes the MINIMUM BOUNDED DEGREE ARBORESCENCE problem, MINIMUM BOUNDED DEGREE STRONGLY k-EDGE-CONNECTED SUBGRAPH problem, etc. We obtain the following results.

Theorem 1.2 *There is a polynomial time $(3, 3B_v^{in}+5, 3B_v^{out}+5)$-approximation algorithm to find a minimum cost subgraph that satisfies crossing supermodular connectivity requirements, and indegree and outdegree constraints in directed graphs. For $\{0,1\}$-valued intersecting supermodular connectivity requirements, there is a polynomial time $(2, B_v^{in}, 2B_v^{out}+2)$-approximation algorithm.*

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 when the cost of the subgraph is not considered.

Theorem 1.3 *For any $\epsilon > 0$, there is no polynomial time $(\infty, 2^{\log^{1-\epsilon} n} B_v)$-approximation algorithm for the degree bounded vertex-connectivity* SURVIVABLE NETWORK DESIGN *problem unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$.*

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. For the $(k,2)$-subgraph problem, we are able to obtain the following.[1]

Theorem 1.4 *There is an $O(\log k \cdot \log n)$-approximation algorithm for the $(k,2)$-subgraph problem.*

For general values of λ, it seems that the (k,λ)-subgraph problem 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 $\epsilon > 0$ [14].

1.3 Techniques and Overview

The iterative rounding method for the SURVIVABLE NETWORK DESIGN problem (without degree constraints) works as follows. Formulate the SURVIVABLE NETWORK DESIGN problem as an integer linear program, and then solve the linear relaxation of the problem to find a basic optimal 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 optimal solution, add an edge with highest value to H, and construct the residual problem) until all the

[1] In the conference version of the paper we claimed an $O(\log^3 n)$-approximation algorithm for the $(k,2)$-subgraph problem, with an incorrect proof. Here we prove a better approximation ratio using a different and simpler proof.

connectivity requirements are satisfied. Jain [23] 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 the SURVIVABLE NETWORK DESIGN problem with degree constraints. The starting point is that degree constraints are defined only on single vertices, and so the uncrossing technique as in [23, 19] can be applied to show that a basic optimal solution is characterized by a laminar family of tight sets (see Lemma 2.3). 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, after we have picked some fractional edges in the previous iterations, we need to allow for *non-integral* degree constraints in the residual problem, otherwise the residual problem may be infeasible, or its cost may be significantly higher. By doing so, however, it is not necessarily true that the picked edges in later iterations have value at least 1/2.

We introduce the idea of *iterative relaxation* to overcome this difficulty. When there is no edge of value at least 1/2 in a basic optimal solution, we prove in Lemma 2.2 that there is a vertex v with degree constraint and it has degree at most 4. The new idea is to "relax" the problem by removing the degree constraint on v. Then, we recompute a basic optimal solution of the residual problem, and iterate this procedure. So, in each iteration, we either round up an edge of value at least 1/2 or relax the problem by removing the degree constraint of a vertex of degree at most 4. Note that the relaxation step only incurs an extra additive constant 3 in the approximation ratio. 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 directed graph results (Theorem 1.2). Subsequently, the iterative relaxaction method has found various applications [4, 24, 30, 43]. In particular, in has been used [43] to settle the conjecture on the MINIMUM BOUNDED DEGREE SPANNING TREE problem affirmatively.

2 Survivable Network Design with Degree Constraints

In this section we study SNDP with non-uniform upper and lower bounds on vertex degrees and present 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 $c : E \to \mathbb{R}^+$ on the edges, connectivity requirements $r_{i,j}$ on pairs of vertices i, j, and a degree upper bound B_v for each vertex $v \in W \subseteq V$ and a degree lower bound L_v for each vertex $v \in U \subseteq V$. 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, in H there are $r_{u,v}$ edge disjoint paths between vertices u, v, for each pair u, v, and each vertex v has $L_v \leq \deg_H(v) \leq B_v$ and each edge e appears at most U_e times in H.

Our method is an extension of Jain's iterative rounding method, as outlined in Section 1.3. 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:

$$f(A) + f(B) \leq f(A \cap B) + f(A \cup B) \quad \text{or,}$$

$$f(A) + f(B) \leq f(A - B) + f(B - A)$$

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 r_{uv} are specified via a function f on the sets of vertices by setting $f(S) := \max_{u \in S, v \notin S}\{r_{uv}\}$. It is known that f is weakly supermodular in this case, and stays weakly supermodular when updating the LP from one iteration to the next (to account for one or more edges being added to the solution subgraph). We denote $x(U) := \sum_{e \in U} x_e$, and $\delta(S)$ denotes the set of edges with exactly one end-point in S for $S \subset V$.

1. Initialization $F \leftarrow \emptyset$, $f' \leftarrow f$, and $\forall v \in W$: $B'_v = B_v$;

2. While $f' \neq \emptyset$ do

   (a) Find a basic optimal solution x with cut requirement f' and remove every edge e with $x_e = 0$.

   (b) If there exists a vertex $v \in W$ with at most 4 incident edges, remove v from W and goto (a).

   (c) If there exists an edge $e = (u, v)$ with $x_e \geq 1/2$, add $\lceil x_e \rceil$ copies of x_e to F and decrease 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 SNDP Algorithm

Below is the LP relaxation for SNDP with degree bounds:

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

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

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 the constraints on f and $L_v \leq \sum_{e \in \delta(v)} \hat{x}_e \leq 2B_v + 3$ for all $v \in V$. Moreover, $\hat{x}$ can be computed in polynomial time.*

First note that any degree lower bound constraint can be incorporated as a connectivity constraint (with $f(\{v\}) := \max\{f(\{v\}), L_v\}$ 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. Henceforth, we assume that aside from connectivity constraints we only have degree upper bounds. We simply refer to them as degree constraints. Note that the B_v's can be fractional, since the algorithm can change the degree bounds fractionally.

To prove Theorem 2.1, we use the algorithm as shown in Figure 1. This is an iterative relaxation procedure for (LP), which is outlined in Section 1.3. The correctness of the algorithm is based on the following key lemma, which guarantees that the algorithm terminates.

Lemma 2.2 *Let x be a basic solution of (LP), and W be the set of vertices with degree constraints. Then either one of the following is true:*

1. *There exists an edge with value at least 1/2.*

2. *There exists a vertex $v \in W$ such that $deg(v) \leq 4$.*

Before giving the proof of this lemma, we first show that it implies Theorem 2.1.

Proof of Theorem 2.1: The results of Jain [23] and Grötschel et al., [21, Theorem 6.4.9] show that a basic optimal solution x^* (if it exists) of the initial LP can be found in polynomial time. It is known that the updated function f' stays weakly supermodular (since we are subtracting a symmetric submodular function), and the LP for the residual problems can be solved in polynomial time [23]. By Lemma 2.2, the above algorithm will always terminate; clearly, the returned solution satisfies f.

We need to prove that the cost of F is at most $2z_{LP}$, where z_{LP} is the cost of the optimal solution x^* of the initial LP, and that $\deg_F(v) \leq 2B_v + 3$ for every vertex v. We prove the former by induction on the number of iterations in which line (2c) of the algorithm is executed. Suppose an edge e with $x^*_e \geq 1/2$ is added to F in the first iteration. By induction, the algorithm finds an edge set F' of cost at most $2z'_{LP}$ that satisfies the residual connectivity requirements, where z'_{LP} is the optimal solution to the residual problem. Note that $z'_{LP} \leq z_{LP} - c_e x^*_e \leq z_{LP} - \frac{c_e}{2}$; the first inequality holds since the restriction of x^* to the edges in $E - e$ is a feasible solution to the residual LP. 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 $v \in V$: $\deg_F(v) \leq 2B_v + 3$. Consider a degree constraint on a vertex v. Suppose the degree constraint on v has never been removed. Since we only add edges with value at least 1/2 to F, it is easy to see that $deg_F(v) \leq 2B_v$. Now suppose the degree constraint is removed after α edges incident on v are added. So, $B_v > \frac{\alpha}{2}$. Since v is of degree at most 4 when its degree constraint is removed, the degree of v in the final solution is at most $\alpha + 4$. As B_v is an integer (B_v is the degree bound in the first iteration), we have $deg_F(v) \leq \alpha + 4 \leq 2B_v + 3$. □

To prove Lemma 2.2, we need a characterization of the basic feasible solutions of (LP). Consider any solution x of the LP. We call a set of vertices S *tight* (with respect to x) if either $S = \{v\}$ and x satisfies the degree constraint for v with equality, i.e., $x(\delta(v)) = B_v$, or x satisfies the connectivity constraint for S with equality, i.e., $x(\delta(S)) = f(S)$ (in the latter case, S may be a singleton or not). We say that a pair of sets S, T *intersect* if all of the sets $S \cap T, S - T, T - S$ are nonempty, and we say that a family of sets $\mathcal{L} = \{S_1, S_2, \ldots, S_\ell\}$ is *laminar* if no two of its sets intersect. For any set $S \subseteq V$, let χ_S denote the incidence vector of the set of edges $\delta(S)$; note that in the constraint matrix of the LP, χ_S is the row for set S (the constraint for S may be either a connectivity constraint or a degree bound). The following lemma characterizes the tight constraints (i.e. constraints satisfied as equalities) of a basic feasible solution. The proof follows from standard uncrossing arguments. We include a sketch for the sake of completeness.

Lemma 2.3 *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 $S = \{v\} \in \mathcal{L}'$ has $B_v > 0$ and every set $S \in \mathcal{L}''$ has $f(S) \geq 1$.

(ii) $|\mathcal{L}| = |E|$.

(iii) The vectors χ_S, $S \in \mathcal{L}$, are linearly independent.

(iv) x is the unique solution to $\{x(\delta(v)) = B_v, \forall \{v\} \in \mathcal{L}'\} \bigcup \{x(\delta(S)) = f(S), \forall S \in \mathcal{L}''\}$.

Proof: This result follows from the standard uncrossing method, see Lemmas 4.1–4.3 of [23], or Chapter 52.4 of [41]. The main point is that if two tight sets A, B intersect then neither can be a singleton set, so the connectivity constraints for A, B must hold with equality (degree constraints are defined on singletons and do not intersect with other sets); 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 $\chi_A + \chi_B = \chi_{A-B} + \chi_{B-A}$. □

Now we sketch the proof of Lemma 2.2. This is similar to the proof of key lemma in [23] as described in [44, Theorem 23.6]. We derive a contradiction if none of the conditions in Lemma 2.2 holds. Let $\mathcal{L}$ be the laminar family of tight sets obtained in Lemma 2.3 when applied to the basic solution just before executing line (2c) of the algorithm. 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 node 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 the terminology of [44], S is said to *own* an endpoint v of edge $e = (u, v)$ if S is the smallest set in $\mathcal{L}$ containing v. Note that there the total number of endpoints in G is $2m$, where $m = |E(G)|$. The proof is established by showing that if every edge e has $x_e < 1/2$ then we can assign endpoints to the sets in such a way that for every set S, S gets at least 3 endpoints and each of its descendants gets at least 2 endpoints. We get a contradiction of having more than $2m$ endpoints, once this argument is applied to the roots of the trees in the forest of laminar family. We need one more definition from [44]. 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 [44].

Lemma 2.4 *Let T be a subtree rooted at S and assume that $x_e < 1/2$ for all $e \in E$. 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 at least 2 endpoints. 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 [44] shows that every other leaf (which is not a degree constraint) satisfies the requirements of the lemma. We include the proof here to illustrate the importance of threshold $\frac{1}{2}$ and the definition of corequirement. Let S be a leaf of $\mathcal{L}$ which is not a degree constraint. Then S can receive one token for each edge in $\delta(S)$. Since $x(\delta(S)) = f(S) \geq 1$ and there is no edge e with $x_e \geq \frac{1}{2}$, we obtain that $|\delta(S)| \geq 3$. Thus S receives at least three tokens and exactly three when $|\delta(S)| = 1$ and $x(\delta(S)) = f(S) = 1$. In this case, $\mathrm{coreq}(S) = \frac{1}{2}$ and the base case of the induction holds.

We now show that the claim holds for a non-leaf set as well. 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-constrained child, and one endpoint from the surplus of one of 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 of incidence vectors of Lemma 2.3), 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 [44] shows that S satisfies the requirements of the lemma. □

This completes the proof of Lemma 2.2, and hence the proof of Theorem 2.1.

Integrality Gap Example

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 $(2, B_v + 2)$, defined as follows. 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 $B_v + 2$. Take a 3-regular 3-edge connected graph G with no Hamiltonian path. Such graphs exist; the following construction was brought to our attention by Jim Geelen and Jacques Verstraete. Let P denote the Petersen graph, let $P - v$ be the graph obtained from P by deleting vertex v, and let us denote the neighbors of v in P by w_1, w_2, and w_3 (so these three vertices have degree 2 in $P - v$). Now, take three copies of $P - v$, and three new vertices v_1, v_2, and v_3, and attach them as follows: add edges from v_j to each of the three copies of w_j ($1 \leq j \leq 3$) in the three copies of $P - v$. It is not hard to argue that this graph on 30 vertices does not have any Hamiltonian path (a key observation is that in any potential Hamiltonian path, the path must visit all the vertices of a copy of $P - v$ befor exiting it; this is not possible given that the Petersen graph is not Hamiltonian). 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 can be seen 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.

Average Degree Bound

Let us assume that $\tilde{B}$ is the average degree upper bound (i.e. $\tilde{B} = \frac{1}{n}\sum_{v \in V} B_v$). 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) of the algorithm:

(a') If there are any edges $e = (u, v)$ with $x_e \geq 1$ 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 $\alpha_v \geq 0$ 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 $x_e \geq 1$, and the degree bounds were decremented by 1.

Claim 2.5 *If $0 < x_e < 1$ for all edges, then there are at most $2n - 1$ edges in a basic feasible solution of (LP).*

Proof: By Lemma 2.3, 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 a laminar family 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 $x_e \geq 1/2$, the total degree of the solution subgraph is at most one more than the total degree of the LP solution. Since there are at most $2n - 1$ iterations left by the above claim, the total degree of the solution subgraph is larger than the total degree of the LP solution by at most $2n - 1$. This implies that the average degree of the solution subgraph is at most 2 more than the average degree of the LP solution.

Minimizing the Maximum Degree

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 $x(\delta(v)) \leq B_v, \forall v \in V$ by $x(\delta(v)) \leq \Delta, \forall v \in V$. 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 problems.

Theorem 2.6 *If (LP-Δ) has an optimal solution with objective value Δ^*, then there exists an integral solution $\hat{x}$ of maximum degree $\leq 2\lceil \Delta^* \rceil + 3$ that satisfies all of the constraints on f. Moreover, $\hat{x}$ can be computed in polynomial time.*

$$
\begin{array}{rrcll}
\text{(DLP) minimize} & z_{DLP} & = & \displaystyle\sum_{e\in E} c_e\, x_e & \\
\text{subject to} & \displaystyle\sum_{e\in\delta^{in}(S)} x_e & \geq & f(S), & \forall\, S\subseteq V, r\notin S \\
& \displaystyle\sum_{e\in\delta^{in}(v)} x_e & \leq & B_v^{in}, & \forall\, v\in W_1 \\
& \displaystyle\sum_{e\in\delta^{out}(v)} x_e & \leq & B_v^{out}, & \forall\, v\in W_2 \\
& 0 \ \leq \ x_e & \leq & U_e, & \forall\, e\in E
\end{array}
$$

Figure 2: LP for directed case with crossing supermodular function f

3 Directed Network Design with Degree Constraints

In this section we present bi-criteria approximation algorithms for degree bounded network design problems in directed graphs, for some restricted classes of connectivity requirements. Our iterative relaxation technique extends to directed graphs, via the results of Gabow [16].

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

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

holds for every pair of sets $A, B \subseteq V$ such that $A \cap B \neq \emptyset$ and $A \cup B \neq V$. Note that the connectivity requirement of the *k-edge-connected spanning subgraph* problem can be formulated via the crossing supermodular function $f(S) = k,\ \forall \emptyset \neq S \subsetneq V$, yet the connectivity requirement of the *directed Steiner tree* problem cannot be so formulated.

First we address the problem of finding a minimum cost subgraph satisfying crossing supermodular connectivity requirements and non-uniform degree requirements (both in-degrees and out-degrees). In the following the connectivity requirements are specified by a crossing supermodular function f. Figure 2 shows the LP-relaxation for our problem. As before U_e is the upper bound on the multiplicity of edge e. We place out-degree bounds for vertices in $W_1 \subseteq V$ and in-degree bounds for vertices in $W_2 \subseteq V$ both of which can be initialized to V initially.

First we prove the following theorem which implies the first part of Theorem 1.2.

Theorem 3.1 *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 $\leq 3z_{LP}$ that satisfies all of the constraints on f if f is crossing supermodular, and $\hat{x}(\delta^{out}(v)) \leq 3B_v^{out} + 5$ and $\hat{x}(\delta^{in}(v)) \leq 3B_v^{in} + 5$ for all $v \in W_1$ and $v \in W_2$, respectively. Moreover, $\hat{x}$ can be computed in polynomial time.*

The proof of this theorem follows from an extension of the method of Gabow [16] similar to our proof of Theorem 2.1. The algorithm is presented in Figure 3. The following lemma ensures that we always make progress either in step (2b) or step (2c). Observe that the proof of Theorem 3.1 follows from Lemma 3.2 by a similar argument as in proof of Theorem 1.1.

Lemma 3.2 *Given a basic solution x of (DLP) in Figure 2 where f is a crossing supermodular function, one of the following conditions must be true.*

1. Initialization $F \leftarrow \emptyset$, $f' \leftarrow f$, and $\forall v \in W_1$: $B_v'^{in} = B_v^{in}$ and $\forall v \in W_2 : B_v'^{out} = B_v^{out}$.
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 exists a vertex $v \in W_1$ with indegree at most 6, remove v from W_1; if there exists a vertex $v \in W_2$ with outdegree at most 6, remove v from W_2. Goto (a).
   (c) For each edge $e = (u, v)$ with $x_e \geq 1/3$, add e to F and decrease $B_u'^{out}$ and $B_v'^{in}$ by $1/3$.
   (d) For every $S \subseteq V$: $f'(S) \leftarrow f(S) - |\delta_F^{in}(S)|$.
3. Return $H = (V, F)$.

Figure 3: Bounded Degree Directed Graph Algorithm

1. There exists $v \in W_1$ with $|\delta^{in}(v)| \leq 6$,

2. There exists $v \in W_2$ with $|\delta^{out}(v)| \leq 6$,

3. There exists an edge e such that $x_e \geq \frac{1}{3}$.

To prove Lemma 3.2, we first introduce some notation and preliminaries. 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 $\delta^{in}(A)$. By an extension of a result of Frank [15] and Melkonian-Tardos [33] the following lemma is immediate.

Lemma 3.3 *Let the requirement function f of (DLP) be crossing 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 and tight degree constraints for $T_1 \subseteq W_1$ and $T_2 \subseteq W_2$ such that*

(i) $|\mathcal{Q}| + |T_1| + |T_2| = |E|$.

(ii) The vectors χ_A for $A \in \mathcal{Q}$, χ_v for $v \in T_1$, and $\chi_{V \setminus v}$ for $v \in T_2$ are linearly independent.

(iii) x is the unique solution to $\{x(\delta^{in}(v)) = B_v^{in}, \forall v \in T_1\} \bigcup \{x(\delta^{out}(v)) = B_v^{out}, \forall v \in T_2\} \bigcup \{x(\delta^{in}(A)) = f(A), \forall A \in \mathcal{Q}\}$.

The cross free family $\mathcal{Q}$ corresponds to a laminar family $\mathcal{L} = \mathcal{I} \cup \mathcal{O}$ with $|\mathcal{L}| = |\mathcal{Q}|$ such that $x(\delta^{in}(S)) = f(S)$ for each $S \in \mathcal{I}$ and $x(\delta^{out}(S)) = x(\delta^{in}(V - S)) = f(V - S)$ for each $S \in \mathcal{O}$ (see Melkonian-Tardos [33]). Also, we augment the family $\mathcal{L}$ by including in it singleton sets corresponding to tight degree constraints in T_1 and T_2 to obtain $\mathcal{L}' = \mathcal{I}' \cup \mathcal{O}'$ where $\mathcal{I}' = \mathcal{I} \cup \{v\}_{v \in T_1}$ and $\mathcal{O}' = \mathcal{O} \cup \{v\}_{v \in T_2}$. Observe that $|\mathcal{L}'| = |\mathcal{Q}| + |T_1| + |T_2| = |E|$. We call members of $\mathcal{I}'$ *square* sets and members of $\mathcal{O}'$ *round* sets.

We now prove Lemma 3.2. The proof is an extension of a similar result (Theorem 3.1) of Gabow [16] where the existence of an edge $x_e \geq \frac{1}{3}$ is proved when degree constraints are not present. In the presence of degree constraints we show that either we have an edge with $x_e \geq \frac{1}{3}$ or the condition where a degree constraint is removed in Lemma 3.2 is satisfied. The laminar family $\mathcal{L}'$ corresponds to a forest $\mathcal{F}$ over the sets in the laminar family where $B \in \mathcal{L}'$ is a child of $A \in \mathcal{L}'$ if A is the smallest set containing B. A node A of $\mathcal{L}'$ is a *leaf, chain* *node* or *branching node* depending on whether it has $0, 1$ or > 1 children. A chain node is a 1-chain node if it belongs to same family $\mathcal{I}'$ or $\mathcal{O}'$ as its unique child; otherwise it is a 2-chain node.

Proof of Lemma 3.2: The proof is by contradiction. Suppose neither of the three conditions holds. We show this leads to the contradiction to the fact that $|\mathcal{Q}| + |T_1| + |T_2| = |\mathcal{L}'| = |E|$. The argument proceeds by assigning two tokens for each edge (one to each endpoint of e), and showing by a counting argument that we can collect two tokens for each member in the laminar family and are still left with some excess tokens.

The token assignment is a detailed argument following Gabow [16] depending on the different cases of the sets. We point out some simple cases from the argument in Gabow [16] and where the presence of degree constraints lead us to give a different argument.

First, we give the following definitions following Gabow [16]. Consider a chain node S with unique child A. Let e be an edge with an end in $S \setminus A$. We will call e *p-directed* (for parent-directed) if it is oriented consistent with S's family ($\mathcal{I}'$ or $\mathcal{O}'$). Formally, it is p-directed if $S \in \mathcal{I}'$ and e enters S or A, or $S \in \mathcal{O}'$ and e leaves S or A. Similarly, it is called *c-directed* if it is oriented consistent with A's family.

The following rule is used to assign the token for endpoint v of edge e.

Definition 3.4 *Token for the endpoint v of an edge e is given to node S of $\mathcal{L}'$ if one of the following holds:*

1. *When S is a leaf, $v \in S$, and e is directed consistent with S's family, i.e., either $S \in \mathcal{I}'$ and $e \in \delta^{in}(S)$ or $S \in \mathcal{O}'$ and $e \in \delta^{out}(S)$.*
2. *When S is a 1-chain node, $v \in S \setminus A$ for A child of S and e is p-directed (or equivalently, c-directed).*

Observe that each leaf node corresponding to a degree constraint obtains at least 7 tokens, otherwise the degree constraint can be removed. The leaf nodes only need two tokens for themselves for the counting argument. The five extra tokens are assigned to other nodes in three different steps, the first of which is the following lemma.

Lemma 3.5 *The number of endpoints available to leaves of $\mathcal{L}'$ can be redistributed to give two tokens to each leaf and branching node of $\mathcal{L}'$ and five tokens to each leaf node which is a degree constraint.*

Proof: A leaf node not corresponding to a degree constraint gets at least four tokens, for e.g, $S \in \mathcal{I}$ receives one token for each edge $e \in \delta^{in}(S)$ and $|\delta^{in}(S)| \geq 4$ since $x(\delta^{in}(S)) = f(S) \geq 1$ and there is no edge e with $x_e \geq \frac{1}{3}$. Leaf nodes which correspond to degree constraint receive at least seven tokens. Since, the number of branching nodes in any tree is strictly less than the number of leaves, we can assign two tokens from each of the leaves to branching nodes giving the claim. $\square$

Now, we still have three extra tokens with the sets corresponding to the degree constrained leaves one of which we use in the following lemma.

Lemma 3.6 *Each 1-chain node has at least two available endpoints if each set in $\mathcal{L}'$ corresponding to the degree constraint donates one token.*

Proof: Consider a 1-chain node S with a child A where wlog $S, A \in \mathcal{I}'$. If both $S, A \in \mathcal{I}$ then we have $x(\delta^{in}(S)) = f(S)$ and $x(\delta^{in}(A)) = f(A)$. Observe that each edge in the difference with a non-zero ($+1$ or -1) co-efficient gives one token to S. Independence of the constraints implies that there must be at least one such edge and the integrality of $f(S)$ and $f(A)$ implies that there cannot be exactly one such edge. Hence, S obtains two tokens in this case.

In the other case, we may have that A corresponds to a degree constraint. Then we do not have integrality since $x(\delta^{in}(A)) = B_v^{in}$ where $A = \{v\}$ and B_v^{in} need not be an integer. But, the independence of constraints implies that S receives at least one token and it borrows another token from A for the induction claim to hold. $\square$

Rest of the proof involves analysis of 2-chain nodes. Lemma 3.2 follows from Lemma 3.5 and Lemma 3.6 if we can show that 2-chain nodes can collect two tokens each for themselves from the remaining unassigned tokens and two extra tokens with each degree constraint.

We start the analysis by defining a subtree $\mathcal{F}_S$ for each 2-chain node S. $\mathcal{F}_S$ is the minimal subtree of $\mathcal{F}$ having S as its root and each leaf either a leaf of $\mathcal{L}'$ or a 2-chain node other than S. In particular, S is always an internal node of tree $\mathcal{F}_S$ and not a leaf node.

The various trees $\mathcal{F}_S$ can overlap: A 2-chain node S occurs at the root in $\mathcal{F}_S$ and also as a leaf in $\mathcal{F}_T$ for T the first 2-chain node that is a proper ancestor of S. It is easy to see that these are the only 2-possibilities. Also observe that a set corresponding to a degree constraint can only occur in one tree since it can never be a root in such a tree.

The token assignment is as follows. Each set A corresponding to a degree constraint gives two tokens to the 2-chain node S where $A \in \mathcal{F}_S$. Thus, each 2-chain node S receives two tokens whenever there is a degree constraint in $\mathcal{F}_S$. In the remaining case we have that there is no degree constraint in $\mathcal{F}_S$. The token assignment is identical in this case as to Gabow [16] and we omit it here. This completes the proof of Lemma 3.2. $\square$

3.1 $\{0,1\}$-Valued Intersecting Supermodular Requirement Functions

We now show how to improve the bounds in the case of intersecting supermodular connectivity requirements (i.e. part two of Theorem 1.2). Recall that an integer function on sets of vertices $f : 2^V \rightarrow \mathbb{Z}^+$ is called intersecting supermodular if the inequality

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

holds for every pair of sets $A, B \subseteq V$ such that $A \cap B \neq \emptyset$. 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. The intersecting supermodular connectivity requirement nonetheless captures the problem of finding an arborescence rooted at r where $f(S) = 1$ if $r \notin S$ and 0 otherwise. The linear programming relaxation is identical to the linear program in Figure 2. We prove the following theorem which implies the second part of Theorem 1.2.

Theorem 3.7 *Suppose we are given a directed graph G, a $\{0,1\}$-valued intersecting supermodular function f as the connectivity requirement and degree bounds B_v^{in} and B_v^{out} for each vertex and consider the corresponding LP relaxation as in Figure 2. There exists a polynomial time algorithm which returns a solution H of cost $\leq 2 \cdot z_{LP}$ where z_{LP} is the cost of the optimum solution to the LP relaxation. Moreover, $deg_H^{out}(v) \leq 2B_v^{out} + 2$ and $deg_H^{in}(v) \leq B_v^{in}$ for all $v \in V$.*

The algorithm is almost identical to the algorithm in Figure 3 with the following modifications: (i) we remove all indegree constraints, (ii) in step (2c), we only pick an edge e if $x_e \geq \frac{1}{2}$ and decrease the degree bounds by $1/2$, and (iii) in step (2b), we remove an outdegree constraint if a vertex's outdegree is at most 3. At the end, we take a minimal subset of the edges that satisfy the connectivity requirements. For intersecting supermodular requirement functions, it can be easily shown that in any minimal feasible solution, the indegree bounds are never violated [2]. Henceforth we assume only out-degree constraints are given.

The following lemma is immediate from Frank [15] and Lemma 3.3.

[2]To see this, by way of contradiction, consider a vertex v with indegree at least 2 in a minimal solution. Suppose the ingoing edges of v are uv and wv. If $G - uv$ is not a feasible solution, then there is a tight set U with requirement 1 such that $u \notin U$ and $v \in U$. Similarly, if $G - wv$ is not a feasible solution, then there is a tight set W with requirement 1 such that $w \notin W$ and $v \in W$. Note that

Lemma 3.8 *Let the requirement function f of (DLP) be intersecting supermodular, and let x be a basic solution of (DLP) such that $0 < x_e < 1$ for all edges $e \in E$. Then there exists a laminar family $\mathcal{Q}$ of tight sets and tight outdegree constraints for $T_2 \subseteq W_2$ such that*

(i) $|\mathcal{Q}| + |T_2| = |E|$.

(ii) The vectors χ_A for $A \in \mathcal{Q}$ and $\chi_{V \setminus v}$ for $v \in T_2$ are all linearly independent.

(iii) x is the unique solution to $\{x(\delta^{out}(v)) = B_v^{out}, \forall v \in T_2\} \bigcup \{x(\delta^{in}(A)) = f(A), \forall A \in \mathcal{Q}\}$.

Observe that Lemma 3.8 differs from Lemma 3.3 since in the case of intersecting supermodular functions, we can ensure that an independent set of inequalities corresponds to a laminar family, while in the case of crossing supermodular functions we could only ensure that an independent set of inequalities corresponds to a cross-free family. We now prove the following lemma which proves Theorem 3.7.

Lemma 3.9 *Given a basic solution x of (DLP) in Figure 2 where f is an intersecting supermodular function, one of the following must be true.*

1. *There exists $v \in T_2$ with $|\delta^{out}(v)| \leq 3$,*
2. *There exists an edge e such that $x_e \geq \frac{1}{2}$.*

Proof: Suppose none of the above conditions holds. Then each vertex with a tight out-degree constraint must have at least four out-edges and each edge e must have $x_e < \frac{1}{2}$. Now, we argue that this leads to a contradiction to the fact that $|\mathcal{Q}| + |T_2| = |E|$. We prove this by the following counting argument. For each edge we assign three tokens. We then redistribute these tokens such that each constraint gets assigned at least three tokens and we still have extra tokens.

In the initial assignment, each edge gives one token to the head and two tokens to the tail of the edge. Hence each vertex gets two tokens for each out-edge incident at it and one token for each in-edge incident at it. For a vertex $v \in T_2$, we use one token for each out-edge at v for the out-degree constraint of v. We use rest of the tokens for connectivity constraints.

Observe that each vertex with an out-degree constraints must have at least four out-edges incident at it. Hence, when we take one token for each out-edge, we obtain at least four tokens for the out-degree constraint, i.e., they have one excess tokens.

For each vertex, we have one token for each in-edge and out-edge incident at it remaining. Moreover, if $v \notin T_2$ we still have two tokens for each out-edge incident at v. We re-assign these tokens such that we collect at least three tokens for each connectivity constraint in $\mathcal{Q}$.

For the laminar family $\mathcal{Q}$, let $\mathcal{L}$ be the forest on the members of the laminar family. We say that a vertex v is owned by $S \in \mathcal{Q}$ if S is the smallest set in $\mathcal{Q}$ containing v. Now, we prove the following lemma.

Lemma 3.10 *Given a subtree of $\mathcal{L}$ rooted at S, we can assign three tokens to each tight degree constraint in S and three tokens to each set R in the subtree. Moreover, we can assign $3 + |\delta^{out}(S)|$ tokens to the root S.*

Proof: The proof is by induction on the height of the subtree.

Base Case. S is a leaf. We have $x(\delta^{in}(S)) = f(S)$ where $f(S)$ is a positive integer. The assumption that there is no edge with $x_e \geq \frac{1}{2}$ implies that there must be three edges in $\delta^{in}(S)$. For each out-edge incident at

$u \in W$ and $w \in U$; otherwise it contradicts that U and W have only 1 incoming edge. Now, since v is in the intersection of U and W, this implies that U and W are intersecting. So, by intersecting supermodularity, both $U \cap W$ and $U \cup W$ are tight. However, both uv and wv enter $U \cap W$, which contradicts that $U \cap W$ is tight (since it has requirement 1 as we have a $\{0, 1\}$ requirement function).

S, S can collect one token. Hence, there must be at least three in-edge tokens and $|\delta^{out}(S)|$ out-edge tokens which can be assigned to S.

Induction Case. S is not a leaf. By induction, we assign $3 + |\delta^{out}(R)|$ tokens to each child R of S. Each child R of S donates one token to S for each edge in $\delta^{out}(R)$. First observe that we can assign one token to S for each out-edge $e \in \delta^{out}(S)$. If the tail of e is in some child R of S, then R has already donated one token for this edge. Else, the tail has been assigned one token for this edge in the initial assignment and can give one token to S. Thus S can be assigned one token for each edge in $\delta^{out}(S)$.

Case 1. S has at least two children R_1, R_2. Since each tight set has connectivity requirement exactly 1, we have $\sum_{R \in \mathcal{Q}} f(R) - f(S) \geq 1$. Let $F_1 = \delta^{in}(S) \setminus (\cup_R \delta^{in}(R))$ and $F_2 = (\cup_R \delta^{in}(R)) \setminus \delta^{in}(S)$. The above inequality implies that $x(F_2) \geq 1$. But then we have $|F_2| \geq 3$, as there is no edge e with $x_e \geq \frac{1}{2}$. So S can collect one token for each edge in F_1 (token assigned to head) and F_2 (one of the two tokens assigned to tail) to get three tokens.

Case 2. S has exactly one child R. Since $f(S) = f(R)$ and χ_S and χ_R are linearly independent, we have $|F_1| \geq 1$ and $|F_2| \geq 1$, where F_1 and F_2 are defined as in previous case. So S can collect one token for each edge in F_1, and one token for each edge in F_2. If S also has a child which is a degree constraint, then we can also collect one excess token from it. Otherwise, the tail of any edge in F_2 does not have a tight degree constraint, and thus can contribute two tokens to S. In either case S can collect the desired three tokens. □

Lemma 3.10 reassigns the tokens so that we have three tokens for each member in $\mathcal{Q}$ and three tokens for each vertex T_2. To prove Lemma 3.9 it remains to show that some tokens are still left in excess. If any root S of the forest $\mathcal{L}$ has at least one out-edge, then S has been assigned at least four tokens and one excess token with S gives us the contradiction. Else, consider any root S. Any $e \in \delta^{in}(S)$ must have their tail at a vertex not owned by any set in $\mathcal{Q}$. If the tail of e has a degree constraint present, it has at least one excess token. Else the out-token for e has not been used in the above assignment and is the excess token which gives us the contradiction. This proves Lemma 3.9. □

4 Hardness Results of Network Design with Degree Constraints

In this section, we show that unlike the degree bounded 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 a degree bound B_v for every $v \in V$, and a connectivity requirement function $r : V \times V \to \mathbb{Z}^+$. The task is to find a minimum cost subgraph G' such that there are at least $r_{u,v}$ *vertex disjoint* paths between u and v and the degree of v is at most B_v. As we will see, it is hard to get an $(\infty, 2^{\log^{1-\epsilon} n} \cdot B_v)$-approximation for this problem under a reasonable complexity assumption. In other words, even when all edge costs are zero and we just have to approximate the degree bounds, the problem remains hard. In fact the same hardness holds for a 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 1.3.

Theorem 4.1 *Unless* NP $\subseteq$ DTIME$(n^{\text{polylog}(n)})$ *there is no* $2^{\log^{1-\epsilon} n}$*-approximation for DkVC for some* $\epsilon > 0$.

Proof: The proof follows essentially the same construction as in [29]. The starting point is the hardness of a graph problem, called MinRep in [29], which is essentially a graph theoretic description of a two-prover one-round proof system. The instance to the MinRep problem consist of a bipartite graph $G(V \cup C, E)$ together with a partitioning of each of V and C to equal parts: $V = \bigcup_{i=1}^{q_v} V_i$ and $C = \bigcup_{i=1}^{q_c} C_j$. Every V_i has size $a_v = |V|/q_v$ and every C_j has size $a_c = |C|/q_c$. Graph G also induces another bipartite super-graph H, where the super-vertices (i.e. vertices of H) are the $q_v + q_c$ sets V_i's and C_j's. There is a super-edge between V_i and C_j if there are vertices $v \in V_i$ and $c \in C_j$ with $vc \in E$. We say that edge vc *covers* the super-edge V_iC_j. A set $S \subseteq V \cup C$ covers a super-edge V_iC_j if there are two vertices $v \in S \cap V$ and $c \in S \cap C$ such that vc covers edge V_iC_j. The goal in the MinRep problem is to select vertices from each V_i and C_j such that every super-edge of H is covered and the total number of vertices selected is minimized. We further assume that for every super edge V_iC_j, every vertex in C_j is adjacent to exactly one vertex in V_i. Also, we can assume that the graph H is regular on each side; say every super-vertex V_i has degree r_v and every super-vertex C_j has degree r_c. So the number of super-edge is $r_vq_v = r_cq_c$. From the PCP theorem [1, 2] together with the parallel repetition theorem [39] it follows that[3]:

Theorem 4.2 *Given an instance ϕ of 3SAT we can build an instance G of MinRep in* DTIME($n^{\text{polylog}(n)}$) *such that:*

- *If ϕ is a yes instance then G has a solution of size $q_v + q_c$.*
- *If ϕ is a no instance then every solution of G has size at least $2^{\log^{1-\epsilon}|V(G)|}(q_v + q_c)$.*

Therefore, unless NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$) *MinRep cannot be approximated within a factor $2^{\log^{1-\epsilon} n}$.*

Given an instance $G(V \cup C, E)$ of MinRep, first we construct a graph $\overline{G}(\overline{V}, \overline{E})$ as follows:

1. Create two new sets of vertices $U = \{u_1, \ldots, u_{q_v}\}$ and $W = \{w_1, \ldots, w_{q_c}\}$, where u_i is adjacent to all the vertices of V_i ($1 \leq i \leq q_v$) and w_j is adjacent to all the vertices of C_j ($1 \leq j \leq q_c$).
2. For every super-edge V_iC_j there are two new vertices x_i^j and y_j^i; and x_i^j is adjacent to u_i and to all the vertices in $V - V_i$ and $C - C_j$. Similarly, y_j^i is adjacent to w_j and to all the vertices in $C - C_j$ and $V - V_i$.
3. Let $X = \bigcup_{i,j} x_i^j$ and $Y = \bigcup_{i,j} y_j^i$. The vertices in $X \cup Y$ form a clique.
4. For every super-edge V_iC_j we require $k = |X| + |Y| + (q_v - 1)a_v + (q_c - 1)a_c$ vertex-disjoint paths between x_i^j and y_j^i.
5. Also, for every vertex $u_i \in U$ we require a degree bound of $r_v + 1$ and for every vertex $w_j \in W$ a degree bound of $r_c + 1$.

See Figure 4. Note $|X| = |Y|$ and is equal to the number of super-edges. Also, the degree of every $u_i \in U$ (every $w_j \in W$) is exactly $a_v + r_v = a_v + |X|/q_v$ (is $a_c + r_c = a_c + |X|/q_c$). The analysis of [29] shows that: In every solution $G' \subset \overline{G}$ which satisfies the connectivity requirements, G' is a spanning subgraph and the number of edges between U and X and between W and Y is exactly $|X|$. This is because every vertex $x_i^j \in X$ can have at most $(q_v - 1)a_v + (q_c - 1)a_c + |Y| + (|X| - 1)$ paths to any other vertex in $X \cup Y$ using the vertices of $(V - V_i) \cup (C - C_j) \cup Y \cup (X - \{x_i^j\})$. Thus for one of its k paths it has to go through u_i. A similar argument works for each vertex $y_j^i \in Y$.

Furthermore, if G is a yes instance of MinRep (i.e. has a solution of size $q_v + q_c$) then there is a solution $G' \subset \overline{G}$ which satisfies the connectivity requirements of S and the number of edges between U and V in G' is q_v (one edge between u_i and V_i, $1 \leq i \leq q_v$) and the number of edges between W and C in G' is q_c (one edge

[3]Readers familiar with 2P1R proof systems can think of each super-vertex V_i as an ℓ-tuple of variables and each C_j as an ℓ-tuple of clauses where we used ℓ parallel repetition; each super-edge corresponds to a pair of queries sent to the two provers; the vertices in each super-vertex correspond to answers to the queries returned by the corresponding prover.

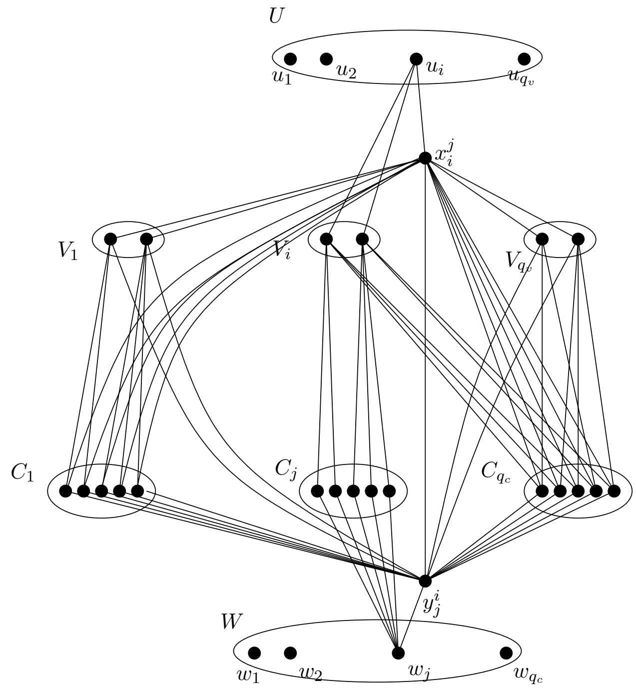

Figure 4: Construction of graph $\overline{G}(\overline{V}, \overline{E})$ from MinRep instance $G(V \cup C, E)$

between w_j and C_j, $1 \leq j \leq q_c$); so the degree of every vertex $u_i \in U$ is $r_v + 1$ and every vertex $w_j \in W$ has degree $r_c + 1$. Conversely, if G is a no instance of MinRep then in every subgraph $G' \subset \overline{G}$ satisfying the connectivity requirements the total number of edges between U and V and between W and C is at least $2^{\log^{1-\epsilon}|V(G)|}(q_v + q_c)$; so at least one $u_i \in U$ or one $w_j \in W$ has a degree larger by a factor of $2^{\log^{1-\epsilon} n}$ from its bound. This, together with Theorem 4.2 implies that deciding between the following two cases is quasi-NP-hard:

- If G has a solution in which all the degree bounds are satisfied.
- If every solution of G has at least one vertex of U with degree at least $2^{\log^{1-\epsilon} n}(r_v + 1)$ or a vertex of W with degree at least $2^{\log^{1-\epsilon} n}(r_c + 1)$.

This completes the proof of Theorem 4.1. □

We have a similar hardness result for the Low Degree Directed Steiner Forest (LDSF) problem. In LDSF, 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 proof of Theorem 4.3 follows from a very similar construction as in Theorem 4.1.

Theorem 4.3 *Unless* NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$) *there is no* $2^{\log^{1-\epsilon} n}$*-approximation for* LDSF *for some* $\epsilon > 0$.

5 Minimum Cost λ-connected k-subgraphs

In this section we study the following class of problems. Given are a (multi)graph $G(V, E)$ with a cost function $c : E \rightarrow \mathbb{R}^+$ on edges, a positive integers k, and a 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 the edge cost function c is 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 an arbitrary 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 $O(\log k \cdot \log n)$-approximation. 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 5.1 *There is an $O(\log k \cdot \log n)$-approximation algorithm for the rooted $(k, 2)$-subgraph problem.*

As mentioned earlier, we show that for an arbitrary λ, the (k, λ)-subgraph problem seems to be difficult. As an evidence, we show a reduction from the k-dense-subgraph problem. In the k-dense-subgraph problem we are given a graph G and an 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 $\epsilon > 0$ [14]. We will prove Theorem 1.5, which implies obtaining any poly-logarithmic approximation for the (k, λ)-subgraph problem would imply a poly-logarithmic approximation for the k-dense subgraph problem.

Remark: One may hope that the constant factor approximation algorithms for k-MST [17, 18, 6] could be extended to obtain an $O(1)$-approximation for (k, λ)-subgraph for at least the special case of $\lambda = 2$. There are several difficulties in this route which do not seem easy to overcome. Firstly, all the known constant approximations for k-MST are based on the primal-dual 2-approximation algorithm of Goemans and Williamson [20] for prize-collecting Steiner tree. It is not clear how to generalize this primal-dual scheme to work for the 2-edge-connected prize-collecting version. Secondly, the typical second step in these algorithms is to use a Lagrangian relaxation to reduce the k-MST problem to the prize-collecting Steiner tree. For this step to work, the known algorithms rely on the fact that we can assume the cost function is *metric* (i.e. satisfies triangle inequality) by simply taking the metric completion of the input graph. For 2-edge-connected subgraphs we cannot make this assumption. However, if we assume that the cost function is metric, then known results on k-TSP [18] immediately imply a 2-approximation algorithm for $(k, 2)$-subgraph. This uses the fact that the LP relaxation for k-TSP is equivalent to the LP relaxation for $(k, 2)$-subgraph when the cost function is metric [5].

5.1 Proof of Theorem 5.1

Our algorithm uses the solution to a related problem we call density 2-edge-connected subgraph problem, denoted by D2ECS. In this problem, we are given an udirected graph $G = (V, E)$ with a cost function $c : E \to \mathbb{R}^+$ on edges. There is a given vertex r, called the *root*, and a subset of vertices $T \subseteq V - r$, called *terminals*. The goal is to find a subgraph G' containing r and at least one terminal such that there are at least 2 edge-disjoint paths between r and each terminal in G', and the ratio of the total cost of G' to the number of terminals in G' is minimized. We will later prove the following lemma:

Lemma 5.2 *There is an $O(\log n)$-approximation for D2ECS.*

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

1. Guess the value of optimum solution, OPT.
2. Delete all the vertices v with $d_2(v, r) > \text{OPT}$ (recall that r is the root).
3. $\mathcal{S} \leftarrow \emptyset$.
4. While $k > 0$ do
   (a) Run D2ECS algorithm to find a good density subgraph H containing r.
   (b) If $|V(H)| \leq 2k$ then
      i. Add H to $\mathcal{S}$ and update $k \leftarrow k - |V(H)|$
      ii. In G contract the edges of H (i.e. H is contracted into r).
   (c) else
      i. Find a subgraph H' of H with cost at most $O(\log n \cdot \text{OPT})$ and at least k nodes.
      ii. Add H' to $\mathcal{S}$ and update $k = 0$.
5. return $\mathcal{S}$.

Figure 5: The algorithm for $(k, 2)$-subgraph problem

Recall that an instance $\mathcal{I}$ to the rooted $(k, 2)$-subgraph has a graph $G = (V, E)$, parameter k, and a root $r \in V$. First we preprocess the graph by removing some of the vertices that cannot be part of an optimum solution. Let us assume we know OPT, the cost of an optimum solution. Then for every vertex v we find two edge-disjoint paths between v and r of minimum total cost, let us denote it by $d_2(v, r)$. For this we can use a min-cost flow algorithm between v and r [41]. Then we delete those vertices v for which $d_2(v, r) > \text{OPT}$, as clearly they cannot be part of the solution. We work with this pruned version of graph G.

The overview of the algorithm is as follows. The algorithm iteratively finds a 2-edge-connected subgraph containing the root with good density, by calling the algorithm for D2ECS (Lemma 5.2) and adds them to an intially empty solution, contract the partial solution found into the root, and iterate until there are at least k nodes in the solution. Clearly the final solution is 2-edge-connected and contains the root. If the total number of nodes at the end is $\Theta(k)$, then a set-cover type analysis shows that the cost of the solution is at most $O(\log k \cdot \log n \cdot \text{OPT})$, where the $\log n$ in the ratio is from the ratio of algorithm of Lemma 5.2 and the $\log k$ comes from a set-cover type analysis. However, some care is needed as the total number of nodes in a good density solution found in every iteration by calling the algorithm for D2ECS might be much larger than k. In that case we show how to prune the solution to obtain one with $\Theta(k)$ nodes and about the same density. The algorithm is presented in Figure 5.

Here are more details for some of the steps. To guess the value OPT, for every vertex $v \neq r$, first we compute $d_2(r, v)$. Let L be the kth smallest value. Clearly, $L \leq \text{OPT} \leq kL$. So it is enough to start with estimate $\text{OPT} = L$ and then double the estimate of OPT if the algorithm does not succeed; we have to do this at most $O(\log k)$ times.

As said earlier, if at every iteration we find a good density partial solution with at most k nodes then a simple set-cover type analysis shows that the cost of the final solution (which will have $\Theta(k)$ nodes) is within a factor $O(\log k \cdot \log n)$ of the optimum. We have to show what to do in step (4(c)i) if one call to D2ECS returns a solution with more than, let's say, $2k$ nodes. Note that this happens at most once (in the last iteration) as after this the number of nodes in the solution built so far will be at least k and the algorithm terminates.

A nowhere-zero 6-flow in a directed graph $D(V, A)$ is a function $f : A \to \mathbb{Z}_6$ such that for each node v: $f(\delta^{in}(v)) = f(\delta^{out}(v))$ and all values of f are non-zero. For an undirected graph H, we say H has a no-where-zero 6-flow if H has an orientation of its edges which has a no-where-zero 6-flow. Seymour [42] proved that every 2-edge-connected graph has a no-where-zero 6-flow and such a flow can be found in polynomial time (see [45, 41]).[4] Let $H(U, F)$ be a good density subgraph returned by D2ECS algorithm with more than $2k$ nodes and let f be a no-where-zero 6-flow on H. We can obtain a directed multi-graph $D(U, A)$ from f and H by placing $f(e)$ copies of each edge e in the direction defined by the flow. Note that D is an Eulerian graph with no directed cycle of length 2. Starting from r in D build an Eulerian walk and partition this walk into arc-disjoint segments $P_1, P_2, \ldots, P_\ell$, each of which induces a subgraph of H on k nodes, except possibly P_ℓ which will have between k and $2k$ nodes. So each P_i is a walk, say from node u_i to u_{i+1}; let's denote the connected subgraph induced by P_i in H by H_i. Since the total cost of the edges of all these walks is at most 5 times the cost of H, at least one of H_i's, say H_j, has density at most 5 times the density of H. Now find two edge-disjoint paths of minimum cost (in G) from u_j to r, call them Q_j^1, Q_j^2, and also two edge-disjoint paths of minimum cost (again in G) from u_{j+1} to r, call them Q_{j+1}^1, Q_{j+1}^2. We claim that the subgraph induced by $H_j \cup Q_j^1 \cup Q_j^2 \cup Q_{j+1}^1 \cup Q_{j+1}^2$, denoted by H'_j, is 2-edge-connected and has cost at most $O(\log n \cdot \textsc{opt})$. To prove the cost, note that H_j has between k and $2k$ nodes and has a density at most five times that of H, which in turn is within $O(\log n)$ of the optimum density. Thus cost of H_j is at most $O(\log n \cdot \textsc{opt})$. Also the total cost of $Q_j^1 \cup Q_j^2 \cup Q_{j+1}^1 \cup Q_{j+1}^2$ is at most $O(\textsc{opt})$ (because of the preprocess step); hence the total cost of H'_j is $O(\log n \cdot \textsc{opt})$. To prove 2-edge-connectivity of H'_j, by way of contradiction suppose that there is a cut-edge $e \in H'_j$ and C_1, C_2 are the two components of $H'_j - e$. Note that because of $Q_j^1, Q_j^2, Q_{j+1}^1, Q_{j+1}^2$, both of u_j and u_{j+1} are in the same component as r, say C_1. But because P_j is a directed walk from u_j to u_{j+1} in D we must be able to travel from C_1 to C_2 in both directions in H'_j (as both u_j and u_{j+1} are in the same side); since there is no directed cycle of length 2, there must be another edge $e' \neq e$ between C_1 and C_2 with a direction opposite of e; so e is not a cut-edge in H'_j.

5.2 Proof of Lemma 5.2

The algorithm and analysis are built upon the LP relaxation and uses ideas from [9]. Consider the following IP/LP relaxation of D2ECS which is based on the LP relaxation of SNDP without degree bounds (see the first LP in Section 2). For each edge e and vertex v we have indicator variables x_e and y_v, respectively, which indicate whether they participate in the solution or not. We have normalized the sum $\sum_{v \in T} y_v$ to 1.

$$
\begin{array}{llrcll}
\text{(LP-D2ECS)} & \text{minimize} & & & \displaystyle\sum_{e \in E} c_e\, x_e & \\
\text{subject to} & & x(\delta(S)) & \geq & 2y_t & S \subseteq V - r, t \in S \\
\text{subject to} & & x(\delta(S)) - x_{e'} & \geq & y_t & S \subseteq V - r, t \in S, e' \in \delta(S) \\
& & \displaystyle\sum_{v \in T} y_v & = & 1 & \\
& 0 \ \leq & x_e, y_v & \leq & 1 & \forall\, e \in E, v \in T
\end{array}
$$

Note that we have added the second set of constraints above to obtain a stronger LP which will help us in the rounding phase. This LP can be solved in polynomial time via the ellipsoid method since there is a polynomial time seperation oracle for this problem.

[4] A weaker and simpler result is that every 2-edge-connected graph has a no-where-zero 8-flow and such a flow can be easily obtained in polynomial time by applying Matroid partition theorem (or Nash-Williams/Tutte theorem for disjoint spanning trees). Our proof works with a no-where-zero 8-flow as well.

Lemma 5.3 *For an instance of D2ECS, let σ be the density of the optimum solution and σ^* be the value of the above LP. Then $\sigma^* \leq \sigma$.*

Proof: Let G' be an optimum solution to the given instance of D2ECS and let $T' \subseteq T$ be the terminals in G'; assume $|T'| = \ell$. So $\sigma = (\sum_{e \in G'} c(e))/\ell$. For each $t \in T'$ define $y_t = \frac{1}{\ell}$ and for each edge $e \in V$ let $x_e = \frac{1}{\ell}$ iff $e \in G'$ and zero otherwise. All other variables are set to zero. It is easy to verify that this is a feasible solution to LP-D2ECS. □

Now we present the algorithm for the D2ECS problem. Given an instance of D2ECS first we solve the LP-D2ECS to obtain an optimum fractional solution (x^*, y^*); let its value be α. For $p = 1 + 2\lceil \log n \rceil$ we obtain disjoint subsets of the terminals $T_1, T_2, \ldots, T_p$ of T as follows. Let $y_{\max} = \max_t y_t$. For $0 \leq a \leq 2\lceil \log n \rceil$, let $T_a = \{t \mid y_{\max}/2^{a+1} < y_t \leq y_{\max}/2^a\}$. Since $\sum_{t \in T} y_t = 1$ there is an index b such that $\sum_{t \in T_b} y_t \geq 1/p$. From this we also have that $|T_b| \cdot y_{\max}/2^b \geq 1/p$. Now find a minimum cost subgraph with connectivity requirement 2 between the terminals in T_b and r. For this we can use Jain's algorithm on the following standard LP for SNDP:

$$
\begin{array}{llrcll}
\text{(LP-2ECS)} & \text{minimize} & \multicolumn{3}{l}{\displaystyle\sum_{e \in E} c_e\, x_e} & \\
 & \text{subject to} & x(\delta(S)) & \geq & 2, & \forall\, S \subseteq V - r, S \cap T_b \neq \emptyset \\
 & & 0 \;\leq\; x_e & \leq & 1 & \forall\, e \in E
\end{array}
$$

Observe that if we take the optimum solution (x^*, y^*) of LP-D2ECS and define $\tilde{x}_e = \min\{1, x_e^* \cdot 2^{b+1}/y_{\max}\}$ this yields a feasible solution to LP-2ECS on terminal set T_b of cost at most $2^{b+1}\alpha/y_{\max}$. To see this, take any set $S \subseteq V - r$ with $S \cap T_b \neq \emptyset$ and the corresponding constraint $x(\delta(S)) \geq 2$ in LP-2ECS. This has corresponding constraints $x(\delta(S)) \geq 2y_t$ in LP-D2ECS for each $t \in S$. Suppose we define $\tilde{x}_e = \min\{1, x_e^* \cdot 2^{b+1}/y_{\max}\}$ and $\tilde{y}_t = \min\{1, y_t^* \cdot 2^{b+1}/y_{\max}\}$. Note that for each $t \in T_b$, because $y_t^* > y_{\max}/2^{b+1}$: $\tilde{y}_t = 1$. If all the edges $e \in \delta(S)$ have values $x_e^* \leq y_t^*$ then after scaling we will have $\tilde{x}(\delta(S)) \geq 2$ because the left hand side of $x(\delta(S)) \geq 2y_t$ is grown at least as much as the RHS is scaled. If there is at least one edge $e' \in \delta(S)$ with $x_{e'}^* > y_t^*$ then because of the second set of constraints in LP-D2ECS, we have $x^*(\delta(S)) - x_{e'}^* \geq y_t^*$. Thus after the scaling we still have $\tilde{x}(\delta(S)) - \tilde{x}_{e'} \geq 1$ because again the LHS is grown at least as much as the RHS. Also $\tilde{x}_{e'} = 1$ because $\tilde{y}_t = 1$ and $x_t^* > y_t^*$; so $\tilde{x}(\delta(S)) \geq 2$. This shows there is a feasible solution to LP-2ECS with terminal set T_b and r with cost at most $2^{b+1}\alpha/y_{\max}$. Jain proved that the integrality gap of LP-2ECS is 2. Therefore, we can obtain an integral solution with connectivity of at least 2 between the terminals in T_b and r such that cost of the solution is at most $2 \times 2^{b+1} \cdot \alpha/y_{\max}$. The density of this solution is therefore $2^{b+2} \cdot \alpha/(y_{\max}|T_b|)$ which is $O(p\alpha)$. Since $p = O(\log n)$ the density is $O(\log n) \cdot \alpha$. Thus, we have an $O(\log n)$-approximation for D2ECS. This completes the proof of Lemma 5.2 and thus Theorem 5.1.

5.3 Proof of Theorem 1.5

In this subsection we present the hardness proof of the (k, λ)-problem, based on the hardness of the densest k-subgraph problem. The overall structure of the proof is as follows. Observe that a solution to the (k, λ)-subgraph problem implies a subgraph on at least k vertices with minimum degree at least λ. We prove that even finding such a subgraph is hard, assuming that densest k-subgraph problem is hard. This implies a hardness for the (k, λ)-subgraph problem too, based on the conjecture that the densest k-subgraph problem is hard.

To prove our hardness result we need the following auxiliary lemma. Given a graph $G = (V, E)$ and integers k, λ, first construct a graph $\hat{G} = (\hat{V}, \hat{E})$ from G by adding a new universal vertex x i.e. $\hat{V} = V \cup \{x\}$ and $\hat{E} = E \cup \{ux | u \in V\}$.

Lemma 5.4 *If $\hat{G}$ has a $(\lambda+1)$-edge-connected subgraph $\hat{H}$ with $B+1$ vertices then $\hat{H}-\{x\} \subseteq G$ has at least B vertices and min-degree at least λ. Conversely, if G has a subgraph H with min-degree at least λ and at least B vertices then $\hat{H} \subseteq \hat{G}$ obtained from H by adding x and all its edges incident to the vertices of H is $(\lambda+1)$-edge connected with at least $B+1$ vertices.*

Proof: First consider the easy part: if $\hat{H} \subseteq \hat{G}$ is $(\lambda+1)$-edge-connected with $B+1$ vertices then clearly $H = \hat{H}-\{x\}$ has minimum degree at least λ and has at least B vertices. Conversely, suppose that $H \subseteq G$ has min-degree at least λ and B vertices. Clearly for every vertex $u \in \hat{H}-\{x\}$, there are at least $|N_H(u)| \geq \lambda$ edge-disjoint paths of length 2 (using $N_H(u)$) from x to u, plus one path which is the single edge between u and x. By transitivity of edge-connectivity, $\hat{H}$ is $(\lambda+1)$-edge-connected with $B+1$ vertices. $\square$

Now we are ready to prove Theorem 1.5. Suppose we are given a graph $G=(V,E)$ and integer k as the instance of densest k-subgraph and let $\mathcal{A}$ be our approximation algorithm for the (k,λ)-subgraph. Without loss of generality, we assume that $k \geq 4$ as for constant values of k the densest k-subgraph is polynomially solvable. Let us suppose that the optimum solution to densest k-subgraph instance is a graph $G'(V',E') \subseteq G$ with $|V'|=k$ and $|E'|=\sigma k$ and furthermore suppose for now that we know the solution G' (this will be cleared later). Note that the density of G' (i.e. $|E'|/|V'|$) is σ. We are going to obtain a subgraph of G' in which the minimum degree is large (i.e. close to the density) and the number of edges of the subgraph is within a constant factor of $|E'|$. To do this, we delete vertices from G' in several rounds. Let $G_i(V_i,E_i)$ be the graph at the beginning of i-th round, δ_i be the minimum degree in G_i, and $d_i = |E_i|/|V_i|$ be its density. From this definition: $G_1 = G'$, $d_1 = \sigma$, and $|V_1| = k$. Without loss of generality, we may assume that $\delta_1 \geq 1$, i.e. G' has no isolated vertices as deleting them does not remove any edges and can only increase the density.

Each round $i \geq 1$ may have several iterations and in each iteration we remove exactly one vertex. At the beginning of round i, if $\delta_i < \frac{d_i}{2\log k} = \frac{|E_i|}{2|V_i|\log k}$ then set $b = 2\delta_i$ and iteratively delete every vertex of G_i with degree at most b. The number of edges deleted in round i is at most $b|V_i| < \frac{|E_i|}{\log k}$. Therefore $|E_{i+1}| \geq \left(1-\frac{1}{\log k}\right)|E_i|$. Round i stops when the minimum degree is at least $b = 2\delta_i$; thus $\delta_{i+1} \geq 2\delta_i$. Since $\delta_1 \geq 1$ and for every $i \geq 1$, $\delta_{i+1} \geq 2\delta_i$, we have strictly smaller than $\log k$ rounds (because for the maximum degree of G': $\Delta(G') \leq k-1$). So after at most $t < \log k$ rounds the algorithm stops. At this point we have: $|E_t| \geq \left(1-\frac{1}{\log k}\right)^t |E_1| \geq \frac{|E'|}{4}$ (because $\log k \geq 2$ and $t < \log k$ which implies $(1-\frac{1}{\log k})^t \geq \frac{1}{4}$) and $\delta_t \geq \frac{d_t}{2\log k}$. Hence:

$$\delta_t|V_t| \geq \frac{|E_t|}{2\log k} \geq \frac{|E'|}{8\log k}. \tag{1}$$

Suppose we guess the values of $|V_t|$ and δ_t (we can simply try all possible values). By the argument above, G has a subgraph with min-degree at least δ_t and at least $|V_t|$ vertices. Furthermore: $\frac{|E'|}{8\log k} \leq \delta_t|V_t| \leq |E'|$. Now we run algorithm $\mathcal{A}$ for $(|V_t|+1, \delta_t+1)$-edge-subgraph on graph $\hat{G}$ which is obtained from G by adding a universal vertex x. Since there is a subgraph of G (namely G_t) with at least $|V_t|$ vertices and minimum degree δ_t and by Lemma 5.4, algorithm $\mathcal{A}$ finds a subgraph $\hat{H}(\hat{V},\hat{E}) \subseteq \hat{G}$ which is (δ_t+1)-edge-connected and $|\hat{V}| \geq |V_t|+1$. Furthermore, because $\mathcal{A}$ is an α-approximation and $\frac{|E'|}{4} \leq |E_t| \leq |E'|$: $|\hat{E}| \leq \alpha|E'|$. Delete vertex x (if it belongs to $\hat{H}$) to obtain graph $\overline{H}(\overline{V},\overline{E})$. So $\overline{H} \subseteq G$ has min-degree at least δ_t and at least $|V_t|$ vertices.

Case 1: if $|\overline{V}| < k$ then we can add $k-|\overline{V}|$ arbitrary vertices from $G-\overline{H}$ to $\overline{H}$ to obtain a graph H with k vertices and at least $\delta_t|\overline{V}| \geq \delta_t|V_t|$ edges. Since $\delta_t|V_t| \geq \frac{|E'|}{8\log k}$ by (1), H is a subgraph with k vertices whose number of edges is at least $\Omega(1/\log k)$ of the optimum solution G'.

Case 2: Suppose that $|\overline{V}| = \beta k$ for some $\beta > 1$. Thus $\delta_t|\overline{V}| = \delta_t\beta k \leq |\overline{E}|$. On the other hand $|\overline{E}| \leq \alpha|E'| \leq 8\alpha\cdot\log k\cdot\delta_t|V_t|$, by (1). These two inequalities, together with the fact that $|V_t| \leq k$, imply that: $\delta_t \beta \cdot k \leq 8\alpha \cdot \log k \cdot \delta_t \cdot k$, which in turn implies

$$\beta \leq 8\alpha \cdot \log k. \tag{2}$$

Now select uniformly at random k vertices out of the βk vertices in $\overline{H}$ and obtain a graph H with k vertices. The expected number of induced edges of H is $\frac{|\overline{E}|}{\beta^2}$. [5] Since $|\overline{E}| \geq \delta_t |\overline{V}| = \delta_t \beta k$, the expected number of edges of H is at least $\frac{\delta_t k}{\beta} \geq \frac{\delta_t |V_t|}{\beta} \geq \frac{|E'|}{8\beta \log k}$, which by (2) is at least $\Omega(\frac{|E'|}{\alpha \cdot \log^2 k})$.

In either case, we can obtain a solution for the densest k-subgraph whose number of edges is within a factor $\Omega(1/\alpha \cdot \log^2 k)$ of the optimum.

6 Concluding Remarks

We present the first constant factor bicriteria approximation algorithms for SNDP with degree constraints. As a corollary, this implies the first constant factor approximation algorithms for finding low degree subgraphs. (E.g. the best previous algorithm for the Minimum Degree k-Edge-Connected Subgraph problem had ratio $O(k \log n)$ for only fixed values of k [11].) Our techniques were recently generalized by [32] to hold for SNDP with weighted degrees. In this problem there is both a weight function and a cost function defined on the edges which are independent of each other. The goal is to find a minimum cost subgraph satisfying connectivity requirements while not violating given *weighted* degree bounds.

Subsequent to this paper, the iterative relaxation method has been successfully applied to obtain *additive* approximation algorithms for degree bounded network design problems. Singh and Lau [43] extended the iterative relaxation method to obtain an $(1, B_v+1)$-approximation algorithm for the MINIMUM BOUNDED DEGREE SPANNING TREE problem [43]. More recently, Lau and Singh [31] have improved Theorem 1.1 and obtained a $(2, B_v + 3)$-approximation algorithm for the MINIMUM BOUNDED DEGREE STEINER FOREST problem, and a $(2, B_v + 6r_{max} + 3)$-approximation algorithm for the bounded degree SNDP where $r_{max} = \max_{u,v}\{r_{uv}\}$. Bansal et al [4] have improved Theorem 1.2 and obtained an $(\frac{1}{\epsilon}, \frac{B_v}{1-\epsilon} + 4)$-approximation algorithm for the MINIMUM BOUNDED DEGREE ARBORESCENCE for $0 < \epsilon \leq \frac{1}{2}$. They also obtain an additive approximation algorithm for the minimum maximum-degree arborescence problem which violates the degrees by at most 2. In all of the above results the iterative relaxation method has been used to obtain (almost) tight analysis for the standard LP formulations for the bounded degree network design problems. This method has also been applied to obtain new approximation results for other combinatorial optimization problems [24], and to obtain simple proofs of classical results in combinatorial optimization and approximation algorithms [30]. We hope this method will find further applications.

With regards to the (k, λ)-subgraph problem the situation is less clear. Although the problem seems difficult to approximate for arbitrary values of λ, the complexity of the problem for small values of λ (even $\lambda = 2$ or 3) is unknown. We do not even know whether the problem has a constant approximation factor for $\lambda = 2$, or if it has a polylogarithmic approximation for any $\lambda \geq 3$. Recently, Chekuri and Korula [10] have presented an $O(\log n \cdot \log k)$-approximation for the $(k, 2)$-subgraph problem where we require 2-vertex-connectivity between the vertices of the solution. Their algorithm (which was obtained independently) is similar to the one in Theorem 1.4, but their analysis is different. More recently, Safari and Salavatipour [40] have shown that if the edge costs of the graph satisfy triangle inequality, then there is an $O(1)$-approximation for the (k, λ)-subgraph problem.

Acknowledgments

Some of the results in this paper were obtained in collaboration with Joseph Cheriyan. We thank him for letting us to include these proofs here. Last author thanks R. Ravi for discussions on results for directed graphs. We

[5] we can actually do this deterministically too, using the method of conditional probabilities.

also thank Chandra Chekuri and Nitish Korula for pointing out an error in the proof of Theorem 5.1 in the extended abstract version of the paper.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and the hardness of approximation problems*, J. ACM 45(3):501-555, 1998.

[2] S. Arora and M. Safra, *Probabilistic Checking of Proofs: A New Characterization of NP*, J. ACM, 45(1):70–122, 1998.

[3] 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.

[4] N. Bansal, R. Khandekar, and V. Nagarajan, *Additive guarantees for degree bounded directed network design*, In Proc. of the 40th ACM Symposium on Theory of Computing (STOC), 2008.

[5] D. Bienstock, M. Goemans, D. Simchi-Levi, and D. Williamson, *A note on the prize collecting traveling salesman problem*, Mathematical Programming 59:413-420, 1993.

[6] 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.

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

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

[9] C. Chekuri, M. Hajiaghayi, G. Kortsarz, M.R. Salavatipour, *Approximation Algorithms for Non-Uniform Buy-at-Bulk Network Design problems*, In Proceedings of FOCS 2006.

[10] C. Chekuri and N. Korula, *Min-Cost 2-Connected Subgraphs with k Terminals*, manuscript 2008, available at http://arxiv.org/abs/0802.2528.

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

[12] 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 Communication, Control and Computing, pages 274–281, 1990.

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

[14] 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 Symp. on Foundations of Computer Science (FOCS) pp 692-701, 1993.

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

[16] H. Gabow, *On the L_∞-norm of extreme points for crossing supermodular directed network LPs*, In Proceedings of IPCO 2005.

[17] N. Garg, *A 3-Approximation for the minimum tree spanning k vertices*, In Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS), 302-309, 1996.

[18] N. Garg, *Saving an epsilon: a 2-approximation for the k-MST problem in graphs*, In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC), 396 - 402, 2005.

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

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

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

[22] M. Hajiaghayi, G. Kortsarz and M. Salavatipour, *Approximating Buy-at-Bulk and Shallow-light k-Steiner trees*, In proceedings of APPROX 2006, LNCS 4110, pp 153-163, 2006.

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

[24] T. Király, L.C. Lau, M. Singh, *Degree bounded matroids and submodular flows*, in Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO), 2008.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] L.C. Lau, R. Ravi, M. Singh, *Iterative Relaxations*, manuscript, 2008.

[31] L.C. Lau and M. Singh, *Additive Approximation for Bounded Degree Survivable Network Design*, In Proc. of the 40th ACM Symposium on Theory of Computing (STOC), 2008.

[32] L. Lewin-Eytan, J. Naor, A. Orda, and M. Singh, *Maximum-lifetime routing in wireless networks: system optimization & game-theoretic perspectives*. manuscript, 2007.

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

[34] 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 Proceedings of the 25th ACM-Symposium on Theory of Computing (STOC), pp 438-447, 1993.

[35] 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.

[36] 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.

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

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

[39] R. Raz, *A parallel repetition theorem*, SIAM J. on computing, 27(3):763-803, 1998.

[40] M. Safari and M.R. Salavatipour, *A constant factor approximation for minimum λ-edge-connected k-subgraph with metric costs*, in Proceedings of APPROX 2008.

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

[42] P.D. Seymour, In Graph Theory and related topics (Proc. Waterloo, 1977). Academic Press (1979), 341-355.

[43] M. Singh, L.C. Lau, *Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal*, In Proc. of the 39th ACM Symposium on Theory of Computng (STOC) 2007.

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

[45] D.H. Younger, *Integer Flows*, J. of Graph Theory 7:349-357, 1983.