ADDITIVE GUARANTEES FOR DEGREE BOUNDED DIRECTED NETWORK DESIGN

NIKHIL BANSAL*, ROHIT KHANDEKAR*, AND VISWANATH NAGARAJAN*

Abstract. We present polynomial-time approximation algorithms for some degree-bounded *directed* network design problems. Our main result is for *intersecting supermodular connectivity requirements with degree bounds*: given a directed graph $G = (V, E)$ with non-negative edge-costs, a connectivity requirement specified by an intersecting supermodular function f, and upper bounds $\{a_v, b_v\}_{v \in V}$ on in-degrees and out-degrees of vertices, find a minimum-cost f-connected subgraph of G that satisfies the degree bounds. We give a bi-criteria approximation algorithm for this problem using the natural LP relaxation, and show that our guarantee is the best possible relative to this LP relaxation. We also obtain similar results for the (more general) class of *crossing supermodular* requirements. In the absence of edge-costs, our result gives the first additive $O(1)$-approximation guarantee for degree bounded intersecting/crossing supermodular connectivity problems.

We also consider the *minimum crossing spanning tree problem*: Given an undirected edge-weighted graph G, edge-subsets $\{E_i\}_{i=1}^k$, and non-negative integers $\{b_i\}_{i=1}^k$, find a minimum-cost spanning tree (if it exists) in G that contains at most b_i edges from each set E_i. We obtain a $+(r-1)$ additive approximation for this problem, when each edge lies in at most r sets. A special case of this problem is *degree-bounded minimum spanning tree*, and our techniques give a substantially shorter proof of the recent $+1$ approximation of Singh and Lau [18].

Key words. approximation algorithms, network design, directed graphs

AMS subject classifications. 68W25, 05C85, 68R10, 90C05

1. Introduction. The problem of finding a minimum spanning tree that satisfies given degree bounds on vertices has received much attention in the field of combinatorial optimization recently. This problem was first studied by Fürer and Raghavachari [6]. Their motivation was to find a broadcast tree in a communication network along which the maximum load of any node, proportional to its degree, is minimized. Assuming unit edge-costs, they gave a local-search based polynomial-time algorithm for computing a spanning tree with maximum degree at most $\Delta^* + 1$ as long as there exists a spanning tree with maximum degree at most Δ^*. This is essentially the best possible since computing the optimum is NP-hard.

Earlier in this decade, a variety of techniques were developed in attempts to generalize this result to the case of arbitrary edge-weights. Ravi et al. [17], using a matching-based augmentation technique, gave a bi-criteria approximation algorithm that violates both the cost and the degree bounds by a multiplicative logarithmic factor. Könemann and Ravi [12] used a Lagrangian relaxation based method to get $O(1)$ approximation on the cost while violating the degrees by a constant factor plus an additive logarithmic term. Chaudhuri et al. [3] based their algorithms on the augmenting-path and push-relabel frameworks from the maximum flow problem and obtained either logarithmic additive violation or constant multiplicative violation on degrees. In a recent break-through result, Goemans [8] presented an algorithm, based on matroid intersection techniques, that computes a spanning tree with cost at most that of the optimum and with degrees at most the bounds *plus* 2. This line of research recently culminated in the "best possible" *plus* 1 result of Singh and Lau [18]. Their algorithm used an iterative rounding approach of Jain [9] while obtaining a spanning tree with cost at most that of the optimum while violating the degrees by at most an

*IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY. {nikhil,rohitk,viswanath}@us.ibm.com.

additive +1 term.

In this paper, we consider *directed* network design problems with either in-degree or out-degree (or both) constraints on the vertices. Directed graphs naturally arise in communication networks. In fact our original motivation was a problem that arose at IBM in the context of maximizing throughput in peer to peer networks. Here, we are given a network where a root node r wishes to transmit packets to all the nodes in the network. However, each node has limited network resources which determines how many packets it can transmit per unit time. It turns out that computing the maximum achievable throughput of this network is equivalent to determining the number of r-arborescences that can be packed in the network subject to out-degree bounds.

As we discuss below, the directed setting turns out to be substantially harder than the undirected setting, and much fewer results are known in this case. We begin with some relevant definitions.

1.1. Preliminaries. A family $\mathcal{A}$ of subsets of V is *intersecting* (resp. *crossing*) if $S, T \in \mathcal{A}$ with $S \cap T \neq \emptyset$ (resp. $S \cap T, V \setminus (S \cup T) \neq \emptyset$) implies $S \cap T, S \cup T \in \mathcal{A}$. A set function $f : \mathcal{A} \to \mathbb{Z}_+$ is called *intersecting supermodular* (resp. *crossing supermodular*), if for any $S, T \in \mathcal{A}$ with $S \cap T \neq \emptyset$ (resp. $S \cap T, V \setminus (S \cup T) \neq \emptyset$), it holds that $f(S \cup T) + f(S \cap T) \geq f(S) + f(T)$.

A family of sets $\{S_1, \ldots, S_k\}$ is called *laminar* if for every two sets, either they are disjoint or one is contained in the other; i.e., for every $1 \leq i, j \leq k$, $i \neq j$, either $S_i \cap S_j = \emptyset$ or $S_i \subset S_j$ or $S_j \subset S_i$.

For a directed graph $G = (V, E)$ and a subset S of vertices, we use $\delta^-_G(S)$ (resp. $\delta^+_G(S)$) to denote the set of edges entering (resp. leaving) S. When the graph G is clear from the context, we drop the subscript G. Consider any non-negative real-value assignment $x : E \to \mathbb{R}^+$ to the edges; we use $x(\delta^-(S))$ (resp. $x(\delta^+(S))$) to denote the total x-value of the edges entering (resp. leaving) S.

Given a directed graph $G = (V, E)$ and an intersecting (or crossing) supermodular set function $f : \mathcal{A} \to \mathbb{Z}_+$ for some set-family $\mathcal{A}$, a subgraph $H = (V, E')$ of G is said to be *f-connected* or satisfy requirement f if $|\delta^-_H(S)| \geq f(S)$ for every $S \in \mathcal{A}$. In the basic directed network design problem [5, 15, 7], given an edge-weighted graph and an intersecting or crossing supermodular set function f, the goal is to compute the minimum-cost f-connected subgraph. In the degree-bounded variant of network design, there are additional constraints bounding the in-degree and out-degree at each vertex. The *degree-bounded directed network design* problem is the following: given a directed graph $G = (V, E)$ with edge-costs $c : E \to \mathbb{R}_+$, an intersecting (or crossing) supermodular set function f and integers $\{a_v, b_v\}_{v \in V}$, compute a minimum-cost f-connected subgraph in which each vertex v has in-degree at most a_v and out-degree at most b_v. The intersecting supermodular requirements are general enough to include the problem of packing k-edge disjoint arborescences, and choosing the minimum-cost edges to increase the rooted connectivity of a directed graph [5, 15]. The crossing supermodular requirements include the problem of computing a minimum-cost k strongly-connected spanning subgraph and several other problems on graphs and hypergraphs, a detailed discussion of which can be found in [7].

We shall consider *bi-criteria* approximation algorithms for which the output may violate the degree-constraints to some extent and its cost is compared to the optimal solution that does not violate any constraints. For functions $\alpha, \beta : \mathbb{Z}_+ \to \mathbb{Z}_+$ and value $\rho \geq 1$, an algorithm for degree-bounded directed network design is called an (α, β, ρ) approximation if for each instance $\langle G, c, f, \{a_v, b_v\}_{v \in V} \rangle$, the algorithm returns an f-

connected subgraph H of cost at most ρ times the optimal f-connected subgraph (that satisfies degree-constraints), with $|\delta^-_H(v)| \leq \alpha(a_v)$ and $|\delta^+_H(v)| \leq \beta(b_v)$ for all $v \in V$.

1.2. Our results and previous work.

Degree-bounded arborescence problem (no costs). Let $G = (V, E)$ be a directed graph with root r, and let b_v be the bounds on out-degree for each vertex v. The goal in the degree bounded arborescence problem is to compute an out-arborescence from r that satisfies the degree bounds or declare that it is infeasible. Since in any arborescence, every vertex except the root has in-degree exactly one, we do not consider bounds on in-degree here. This problem was first considered by Fürer and Raghavachari [6] who gave a polynomial time algorithm to compute an arborescence that violates the degree bound by at most a logarithmic multiplicative factor. Subsequently Klein et al. [11] gave a quasi-polynomial time algorithm with degree violation $(1 + \epsilon)b_v + O(\log_{1+\epsilon} n)$ for any $\epsilon > 0$. Their algorithm starts with a solution and successively applies local improvement steps to reduce high degrees. Recently, Lau et al. [13], using an iterative rounding technique, obtained a polynomial-time algorithm that computes an arborescence with degrees at most $2 \cdot b_v + 2$. We obtain the first result with only *additive* violation in the degree bounds: an algorithm that constructs an arborescence with degrees at most $b_v + 2$. Call a directed graph k-arc-strong if every directed cut has at least k edges. Our techniques also imply the following result: any k-arc-strong graph G contains an arborescence T with $\delta^+_T(v) \leq \lceil \frac{\delta^+_G(v)}{k} \rceil + 2$ for all vertices v in G. This almost settles the following conjecture, for which the previously best known result [1] was the existence of an arborescence T with $\delta^+_T(v) \leq \frac{\delta^+_G(v)}{2^{\lfloor \log_2 k \rfloor}} + \lfloor \log_2 k \rfloor$.

Conjecture 1 (Bang-Jensen et al. [1]). *Let G be k-arc-strong directed graph. There exists a spanning arborescence T with $\delta^+_T(v) \leq \frac{\delta^+_G(v)}{k} + 1$ for all vertices v in G.*

General connectivity requirements with degree bounds. We consider the network design problem in directed graphs where the connectivity requirement is specified by an arbitrary *intersecting supermodular* function [5], and there are both in-degree and out-degree bounds $\{(a_v, b_v)\}_{v \in V}$ on vertices. The goal here is to find a minimum-cost subgraph (if it exists) that satisfies the connectivity requirement and degree bounds on vertices. The previously best known results for this problem are a $(3a_v + 4, 3b_v + 4, 3)$ approximation in general, and a $(2a_v + 2, 2b_v + 2, 2)$ approximation for the special case of 0-1 valued functions [13]. We extend and improve this result by giving a $(\lceil \frac{a_v}{1-\epsilon} \rceil + 4, \lceil \frac{b_v}{1-\epsilon} \rceil + 4, \frac{1}{\epsilon})$ approximation algorithm for every $\epsilon \in [0, \frac{1}{2}]$. Here we use the convention that $1/0 = \infty$. Setting $\epsilon = 0$, gives the first additive (plus 4) guarantee for the unweighted (no edge-costs) versions of the these problems. As in Lau et al. [13], our algorithm is based on rounding the fractional solution to a natural linear relaxation of the problem (described later); hence the cost guarantee is relative to the optimal value of this LP relaxation.

It also turns out that the above trade-off between the cost blowup and the degree-bound violation, is the best possible using the natural LP relaxation. In fact the integrality gap holds even for the simpler degree-bounded arborescence problem. This suggests that computing low-cost arborescence subject to degree bounds might be an inherently harder problem in the directed setting unlike the undirected case (where the optimal $(b_v + 1, 1)$ result was obtained via the LP [18]).

For degree-bounded network design under the more general *crossing supermodular* connectivity requirements, Lau et al. [13] gave a $(3a_v + 4, 3b_v + 4, 3)$ approximation algorithm. Our approach gives for any $\epsilon \in [0, \frac{1}{2}]$, a $(\lceil \frac{a_v}{1-\epsilon} \rceil + 4 + f_{\max}, \lceil \frac{b_v}{1-\epsilon} \rceil + 4 + f_{\max}, \frac{2}{\epsilon})$ approximation algorithm, where $f_{max} = \max_{S \subseteq V} f(S)$ is the maximum connectivity requirement. Again setting $\epsilon = 0$, we obtain a plus $(f_{max} + 4)$ additive approximation for the unweighted case. For example, this implies a $+6$ additive approximation for the degree-bounded 2-strongly-connected subgraph problem.

Minimum crossing spanning tree problem (MCSP). Given an undirected graph $G = (V, E)$, costs $c_e \geq 0$ on the edges $e \in E$, subsets of edges $E_i \subseteq E$ for $1 \leq i \leq k$, and integers $b_i \geq 0$ for $1 \leq i \leq k$, the MCSP is to find a minimum-cost spanning tree (if it exists) in G that contains at most b_i edges from set E_i for $1 \leq i \leq k$. We obtain a polynomial-time algorithm for this problem that computes a spanning tree of cost at most the optimum, containing at most $b_i + r - 1$ edges from E_i (for all $1 \leq i \leq k$); Here $r = \max_{e \in E} |\{i \mid e \in E_i, 1 \leq i \leq k\}|$, is the maximum number of sets $\{E_i\}$ that any edge lies in. This significantly improves on the results of Bilò et al. [2], who gave an $O(r \log n)$ multiplicative guarantee on the number of edges chosen in sets E_i.

We mention two special cases of our algorithm for MCSP. If the sets E_i are pairwise-disjoint, our algorithm computes an optimal solution. In this case, the MCSP problem can be cast as finding a minimum-cost basis in the graphic matroid for G that is independent in a partition matroid. This problem is an instance of the *matroid intersection problem* which is known to be solvable in polynomial time [4, 14]. As another example, if E_i denotes the set of edges incident to vertex i and b_i denotes the degree bound on vertex i, the MCSP problem reduces to the *degree-bounded minimum spanning* tree problem. Our algorithm matches the best possible $+1$ bound for this problem obtained by Singh and Lau [18]; we note that our proof of Theorem 6.1 is considerably simpler than that in [18]. In fact, Theorem 6.1 readily extends to a generalization of MCSP: that of computing a minimum-cost basis in a matroid subject to 'degree bounds'. This problem was recently considered by Király et al. [10].

1.3. Our approach. Our algorithms are based on the iterative rounding technique of Jain [9], and an extension of it (iterative relaxation) used in Lau et al. [13] and Singh and Lau [18] in the context of degree-bounded network design. The *iterative rounding* technique introduced in [9], which has been extensively used in network design problems, proceeds as follows. First the problem is formulated as an integer program, and an LP relaxation is obtained. An extreme point solution, a.k.a. basic feasible solution, to this linear program is then computed. The extreme point solutions are proved to exhibit useful structural properties, for example, the existence of a variable with near-integral value. Such variables are then rounded up to integral values and the residual problem is solved iteratively. For example, Jain [9] established the existence of $\frac{1}{2}$ edges in every extreme point to the *survivable network design problem*, and obtained a 2-approximation by iteratively rounding such variables.

Iterative relaxation was introduced in [13, 18] as an extension of the above method, that is useful for degree-bounded network design problems. Here the idea is again to work with a suitable LP relaxation, and prove some properties of extreme point solutions. In each iteration, one of the following steps is performed: (1) Round a near-integral variable (as above), or (2) drop some degree constraint while bounding the violation of this constraint in the subsequent steps. The difference from iterative rounding is the second step (degree relaxation). For example, Singh and Lau [18] use a clever counting argument to show that in any extreme point solution to their

LP formulation of *degree-bounded MST*, either there is an integral edge-variable, or the degree constraint of some vertex can be dropped without violating it by more than +1 in the subsequent steps. In each iteration, the algorithm either sets such an edge to its integral value or drops such a constraint; thereby obtaining a $(b_v + 1, 1)$ approximation.

Challenges in extension to the directed case. In the directed setting, the arborescence polytope (without degree bounds) has a linear formulation using the cut-covering constraints; it is not known to have a formulation similar to the edge-subset formulation for spanning-trees, which was used in [18] for the undirected case. One difficulty in working with the cut formulation is that when used along with degree bounds, the cut-constraints may alone contribute $2|V| - 1$ tight linearly-independent constraints in a basic solution. Using some additional arguments, Lau et al. [13] show that either there exists an edge e with $x_e \geq \frac{1}{2}$ or there is a vertex v with small degree in the support. Based on this, their algorithm iteratively does one of the following: round edge e to 1 or drop the degree-constraint of vertex v. Since this algorithm rounds $\frac{1}{2}$-edges to 1, the degree bounds may be violated by a *multiplicative* factor of two.

We overcome these difficulties by introducing additional iterative rounding steps and stronger counting arguments. We continue to use the idea of dropping degree constraints from Lau et al. [13]; so at any iteration the degree bounds are present only at a subset W of the vertices. The degree-bound relaxation step used in Lau et al. [13] only considers vertices that have a small degree in the support. We extend this step by considering all vertices that have small *spare* (i.e., difference of support degree and fractional degree). We note that such a relaxation step was also used in the +1 algorithm for bounded degree MST [18], but not in the directed counterpart [13]. In addition, we also use some new relaxation steps that involve treating edges leaving W vertices and non-W vertices differently; this is the basis of the cost/degree trade-off. Finally, as is the case with iterative rounding algorithms, we need a careful counting argument to show that progress is possible at every iteration. These arguments [9, 15, 13, 18] usually involve a *token-assignment* scheme that first distributes tokens to variables and then extracts tokens from constraints. The novelty in our counting arguments is that the token-assignment to each variable depends on the *fractional value* of that variable in the basic solution. To the best of our knowledge, the earlier proofs based on iterative rounding used only integral token-assignment schemes.

We note that our token-assignment scheme is quite simple and lends itself to global counting arguments. In this paper we have applied them to (both directed and undirected) degree bounded network design problems. Subsequent to this work, Nagarajan et al. [16] employed a similar token-assignment scheme for the undirected *Steiner network* problem to obtain a substantially simpler proof of Jain's 2-approximation algorithm [9].

1.4. Organization. The rest of the paper is organized as follows. In Section 2, we consider the unweighted degree-bounded arborescence problem. This result contains the basic ideas used in the rest of the paper as well. In Section 3, we consider degree-bounded network design under intersecting supermodular connectivity requirements with costs. We then show in Section 4, that this algorithm can be used to solve the more general degree-bounded network design problem, with crossing supermodular connectivity requirements. In Section 5, we complement our approximation guarantee by showing a tight integrality gap of the natural LP relaxation for even the minimum-cost degree-bounded arborescence problem. In Section 6, we study the

- Set $F \leftarrow \emptyset$ and $W \leftarrow V$.
- If $P(E, F, W)$ is infeasible, output "infeasible".
- Repeat while $E \setminus F \neq \emptyset$
  1. Compute a basic feasible solution x to $P(E, F, W)$.
  2. Remove from E all edges $e \in E \setminus F$ with $x_e = 0$.
  3. Add to F all edges $e \in E \setminus F$ with $x_e = 1$.
  4. For all $v \in W$ such that there are at most $b_v - |\delta_F^+(v)| + 2$ edges leaving v in $E \setminus F$,
     (a) Remove v from W.
     (b) Add to F all out-going edges from v in $E \setminus F$.
- Output any (out-)arborescence rooted at r in F.

FIG. 2.1. *Algorithm for degree-bounded arborescence*

undirected minimum crossing spanning tree problem.

2. Degree-bounded arborescence problem. In this section, we prove the following result.

THEOREM 2.1. *There is a polynomial time algorithm that given a directed graph with out-degree bounds $\{b_v\}_{v \in V}$, either constructs an (out-)arborescence such that any vertex v has out-degree at most $b_v + 2$ or shows that no arborescence satisfies the degree bounds exactly.*

Our algorithm, given in Figure 2.1, proceeds in several iterations. In a general iteration of the algorithm, we denote E to be the candidate set of edges, initially containing all the edges. The set $F \subseteq E$ denotes the edges that we have already picked in our solution and the set $W \subseteq V$ denotes the vertices on which the out-degree bounds constraints are present. Initially, $F = \emptyset$ and $W = V$. In any iteration, we work with the following linear program with variables x_e for $e \in E \setminus F$. Let $E' = E \setminus F$. For brevity, we use δ^- (resp. δ^+) to denote $\delta_{E'}^-$ (resp. $\delta_{E'}^+$).

$$
\begin{aligned}
& P(E, F, W): \\
x(\delta^-(S)) \geq 1 - |\delta_F^-(S)| \quad & \forall S \subseteq V \setminus \{r\} \quad \text{(cut-constraints)} \\
x(\delta^+(v)) \leq b_v - |\delta_F^+(v)| \quad & \forall v \in W \ \text{(degree-constraints)} \\
0 \leq x_e \leq 1 \qquad & \forall e \in E' = E \setminus F
\end{aligned}
$$

In the beginning of every iteration, we compute a *basic feasible solution* x in the polytope $P(E, F, W)$ as described in Jain [9]. We then update the sets E, F, and W as explained in Figure 2.1. The algorithm, in the end, outputs any arborescence contained in the set of edges F.

The following lemma is easily seen, and we omit the proof.

LEMMA 2.2. *Assume that $P(E, F, W)$ is feasible at the beginning of the algorithm. If the algorithm terminates, it outputs an arborescence T such that $|\delta_T^+(v)| \leq b_v + 2$ for all $v \in V$.*

The rest of the section is devoted to proving that the algorithm indeed terminates. We show that if $|E|$ and $|F|$ do not change in Steps 2 and 3, then $|W|$ must decrease in this iteration. Assume that the conditions in Steps 2 and 3 do not hold, i.e., all $e \in E'$ satisfy that $0 < x_e < 1$. In such a case, all the tight constraints in the basic feasible solution x come from the cut-constraints and the degree-constraints. Moreover, since all edges leaving v are added to F as soon as v is removed from W, every edge in

$E \setminus F$ must be out-going from a W-vertex.[1] The following lemma is standard and obtained by using the fact that the RHS of the cut-constraints is a supermodular set function. The proof is omitted.

LEMMA 2.3 ([13]). *For any basic solution x to $P(E, F, W)$ such that $0 < x_e < 1$ for all $e \in E'$, there exists a set $T \subseteq W$ and a laminar family $\mathcal{L}$ of subsets of V such that x is the unique solution to the linear system:*

$$\begin{aligned} x(\delta^-(S)) &= 1 && \forall S \in \mathcal{L}, \\ x(\delta^+(v)) &= b_v - |\delta_F^+(v)| && \forall v \in T. \end{aligned}$$

Furthermore, the following two conditions are satisfied

1. *The characteristic vectors $\{\chi_{\delta^-(S)} \mid S \in \mathcal{L}\} \cup \{\chi_{\delta^+(v)} \mid v \in T\}$ are linearly independent.*
2. *The size of the support is equal to $|E'| = |T| + |\mathcal{L}|$.*

For $v \in W$, we define its *spare*, $\mathrm{Sp}(v)$, as the difference between its degree in the support and its fractional degree:

$$\mathrm{Sp}(v) = \sum_{e \in \delta^+(v)} (1 - x_e) = |\delta^+(v)| - \sum_{e \in \delta^+(v)} x_e.$$

For $v \in W$, let $d_v = b_v - |\delta_F^+(v)|$ be the current degree bound on v. Since x_e is a feasible LP solution, $\sum_{e \in \delta^+(v)} x_e \leq d_v$ and hence $\mathrm{Sp}(v) \geq |\delta^+(v)| - d_v$. Thus $\mathrm{Sp}(v)$ is an upper bound on the degree violation of vertex v if its degree bound is dropped.

To complete the proof of Theorem 2.1, we prove the following lemma that shows that if neither Step 2 nor Step 3 in the algorithm apply, then Step 4 applies.

LEMMA 2.4. *If neither Step 2 nor Step 3 is applicable, then there exists $v \in W$ such that $|\delta^+(v)| - d_v \leq 2$.*

Proof. We first argue that it is enough to show that

$$|\mathcal{L}| < \sum_{e \in E'} x_e + 2|W|. \tag{2.1}$$

Suppose (2.1) holds. Consider the quantity $\sum_{v \in W} \mathrm{Sp}(v)$. As each (u, v) in E' has its tail u in W, it follows that $\sum_{v \in W} \mathrm{Sp}(v) = |E'| - \sum_{e \in E'} x_e$. Since $\mathrm{Sp}(v) \geq \delta^+(v) - d_v$, we have

$$\begin{aligned} \sum_{v \in W} (\delta^+(v) - d_v) &\leq |E'| - \sum_{e \in E'} x_e = |\mathcal{L}| + |T| - \sum_{e \in E'} x_e \qquad \text{(by Lemma 2.3)} \\ &\leq |\mathcal{L}| + |W| - \sum_{e \in E'} x_e < 3|W| \qquad \text{(by inequality (2.1))} \end{aligned}$$

This in turn implies that there exists $v \in W$ such that $|\delta^+(v)| - d_v < 3$. Since $|\delta^+(v)| - d_v$ is an integer, it must be at most 2.

The proof of (2.1) is based on a counting argument, as is common in iterative rounding. We assign x_e units of "tokens" to each $e \in E'$ and two "tokens" to each $v \in W$. We shall show that these tokens can be redistributed among the sets $S \in \mathcal{L}$ such that each set in $\mathcal{L}$ gets at least one token, and moreover one token is unused, thereby proving that $|\mathcal{L}|$ is strictly smaller than the total number of tokens $\sum_{e \in E'} x_e + 2|W|$.

[1] A vertex in W is henceforth called a W-vertex.

The laminar family $\mathcal{L}$ naturally defines a forest $\mathcal{T}$ with $S \in \mathcal{L}$ as nodes[2]. We call a node $S \in \mathcal{L}$ *marked* if there is some vertex $w \in W \cap S$; or *unmarked* otherwise. Recall that every edge in E' leaves a W-vertex; hence if S is an unmarked node, no edge of E' leaves a vertex in S and in particular, no edge of E' is contained in S. From Lemma 2.3, for any set $S \in \mathcal{L}$, $x(\delta^-(S)) = 1$. The assignment of tokens to nodes of $\mathcal{T}$ is done as follows.

Leaf nodes in $\mathcal{T}$. Let $S \in \mathcal{L}$ be a leaf in $\mathcal{T}$. Recall that $x(\delta^-(S)) = 1$. The tokens of edges $e \in \delta^-(S)$, which sum up to 1, are assigned to S.

Unmarked non-leaf nodes in $\mathcal{T}$. We in fact show that such nodes do not exist in $\mathcal{T}$ at all. Let on the contrary, $S \in \mathcal{L}$ be such a node, and $C_1, \cdots, C_t \subset S$ with $t \geq 1$ be its children in $\mathcal{T}$. Since S is unmarked, no edge of E' lies completely inside S, hence $\delta^-(C_i) \subseteq \delta^-(S)$ for all i, and thus $\sum_{i=1}^{t} x(\delta^-(C_i)) \leq x(\delta^-(S))$. As $x(\delta^-(S)) = x(\delta^-(C_i)) = 1$ for all i, this implies that $t = 1$ and $\chi_{\delta^-(S)} = \chi_{\delta^-(C_1)}$. But this contradicts the linear independence in Lemma 2.3.

Marked nodes in $\mathcal{T}$. Let $\mathcal{M} \subseteq \mathcal{T}$ denote the sub-forest induced on the marked nodes in $\mathcal{T}$. Call a node $S \in \mathcal{M}$ *high-degree* if S has at least 2 children in $\mathcal{M}$; *low-degree* if S has exactly 1 child in $\mathcal{M}$; all other nodes are leaves in $\mathcal{M}$.

Since leaves in $\mathcal{M}$ correspond to disjoint sets, every such node contains at least one *distinct* W-vertex. We next argue that each low-degree node in $\mathcal{M}$ also contains a *distinct* W-vertex, distinct also from the W-vertices contained in the leaves of $\mathcal{M}$. Let $S \in \mathcal{M}$ be a low-degree node in $\mathcal{M}$, and $C \in \mathcal{M}$ be its unique child in $\mathcal{M}$. To establish the above property, it is enough to show that $W \cap (S \setminus C) \neq \emptyset$. Suppose this is not the case. As $S \setminus C$ does not contain any W-vertex, there are no edges from $S \setminus C$ to C; so $\delta^-(C) \subseteq \delta^-(S)$. As $x(\delta^-(C)) = x(\delta^-(S)) = 1$, we get $\chi_{\delta^-(S)} = \chi_{\delta^-(C)}$ contradicting the linear independence.

Thus we proved that the total number of leaves and low-degree vertices in $\mathcal{M}$ is at most $|W|$. Now since $\mathcal{M}$ is a forest, the number of high-degree nodes in $\mathcal{M}$ is *strictly less* than the number of leaves in $\mathcal{M}$. Therefore the total number of nodes in $\mathcal{M}$ is strictly less than $2|W|$. Assign each node in $\mathcal{M}$ a distinct token out of $2|W|$ tokens from vertices in W leaving at least one token unassigned.

By the token assignment given above, each set in $\mathcal{L}$ gets at least one token with one token unassigned. Thus the proof is complete. □

The above result implies the following slightly weaker version of Conjecture 1 of Bang-Jensen et al. [1].

COROLLARY 2.5. *Let $G = (V, E)$ be a k-arc-strong graph, i.e., a directed graph in which every directed cut has at least k edges. For any $r \in V$, there exists an r-rooted arborescence T satisfying $\delta_T^+(v) \leq \lceil \frac{\delta_G^+(v)}{k} \rceil + 2$ for every $v \in V$.*

Proof. Consider the degree-bounded arborescence problem on G with any root $r \in V$ and degree bounds $b_v = \lceil \delta_G^+(v)/k \rceil$ at each $v \in V$. It is clear that $x = \frac{1}{k} \cdot \chi_E$ is a feasible *fractional* solution to the linear relaxation $P(E, \emptyset, V)$ of this problem. Thus our algorithm obtains an arborescence rooted at r with the desired property. □

3. Intersecting supermodular connectivity with costs. We now consider degree-bounded network design under an intersecting supermodular connectivity re-

[2]Throughout, we use node to refer to a vertex in the laminar tree, and vertex to refer to a vertex in G.

quirement, and prove the following theorem.

THEOREM 3.1. *For any $\epsilon \in [0, \frac{1}{2}]$, there is a polynomial time $(\lceil \frac{a_v}{1-\epsilon} \rceil + 4, \lceil \frac{b_v}{1-\epsilon} \rceil + 4, \frac{1}{\epsilon})$ approximation algorithm for degree bounded network design with intersecting supermodular requirement.*

The algorithm is again iterative. Let $F \subseteq E$ denote the set of edges that have been fixed to value 1, $I \subseteq V$ the vertices for which there is an in-degree bound, and $O \subseteq V$ the vertices for which there is an out-degree bound at some generic iteration. Consider the following LP which we refer to as $P(E, F, I, O)$.

$$\begin{array}{ll} \min & \sum_{e \in E \setminus F} c_e x_e \\ s.t. & \\ & x(\delta^-(S)) \geq f(S) - |\delta_F^-(S)| \quad \forall S \subseteq V \\ & x(\delta^-(v)) \leq a_v - (1-\epsilon)|\delta_F^-(v)| \quad \forall v \in I \\ & x(\delta^+(v)) \leq b_v - (1-\epsilon)|\delta_F^+(v)| \quad \forall v \in O \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \setminus F \end{array} \tag{3.1}$$

In such an iteration, the algorithm computes an optimal basic feasible solution x. Let $E' = E \setminus F$. The algorithm works with a parameter $0 \leq \epsilon \leq 1/2$ and performs one of the following steps in each iteration where $E' \neq \emptyset$:

1. If there is an edge $e \in E'$ with $x_e = 0$, set $E \leftarrow E \setminus \{e\}$.
2. If there is an edge $e \in E'$ with $x_e \geq 1 - \epsilon$, set $F \leftarrow F \cup \{e\}$.
3. If there is an edge $e = (u, v) \in E'$ with $u \notin O$ *and* $v \notin I$ and $x_e \geq \epsilon$, set $F \leftarrow F \cup \{e\}$.
4. If there is $v \in I$ with strictly less than $a_v - (1-\epsilon)|\delta_F^-(v)| + 5$ edges in E' entering it, set $I \leftarrow I \setminus \{v\}$.
5. If there is $v \in O$ with strictly less than $b_v - (1-\epsilon)|\delta_F^+(v)| + 5$ edges in E' leaving it, set $O \leftarrow O \setminus \{v\}$.

Note that steps 2 and 3 ensure that any edge adjacent to a vertex with degree bound is chosen only if $x_e \geq 1 - \epsilon$. Moreover, 3 ensures that any other edge that is chosen has x_e value at least ϵ. It is easily verified that if at least one of these conditions holds at each iteration, then the algorithm results in a solution F satisfying the connectivity requirement, of cost at most $\frac{1}{\epsilon}$ times the optimal, while having in-degree at most $\lceil \frac{a_v}{1-\epsilon} \rceil + 4$ and out-degree at most $\lceil \frac{b_v}{1-\epsilon} \rceil + 4$ at each vertex $v \in V$.

The rest of this section proves that one of the above conditions is true in any iteration. In particular, we show that if none of the conditions (1)-(3) are satisfied in some iteration, then at least one of (4) and (5) must be true. To this end, fix an iteration and assume that none of (1)-(3) are satisfied. As in the previous section, since conditions (1) and (2) do not hold, all the tight constraints in a basic feasible solution x come from the cut-constraints and the degree-constraints. Based on standard uncrossing arguments, we have the following. The proof is omitted.

LEMMA 3.2 ([13]). *For any basic solution x to $P(E, F, I, O)$ such that $0 < x_e < 1$ for all $e \in E'$, there exist sets $I' \subseteq I$, $O' \subseteq O$, and a laminar family $\mathcal{L}$ of subsets of V such that x is the unique solution to the linear system:*

$$\begin{array}{ll} x(\delta^-(v)) = a_v - (1-\epsilon)|\delta_F^-(v)| & \forall v \in I' \\ x(\delta^+(v)) = b_v - (1-\epsilon)|\delta_F^+(v)| & \forall v \in O' \\ x(\delta^-(S)) = f(S) - |\delta_F^-(S)| & \forall S \in \mathcal{L} \end{array}$$

Furthermore, the following three conditions hold:

1. *For every $S \in \mathcal{L}$, $f(S) - |\delta_F^-(S)| \geq 1$ and is integral.*

2. *The characteristic vectors* $\{\chi_{\delta^-(S)} \mid S \in \mathcal{L}\} \cup \{\chi_{\delta^-(v)} \mid v \in I'\} \cup \{\chi_{\delta^+(v)} \mid v \in O'\}$ *are linearly independent; and*
3. *The size of the support* $|E'| = |I'| + |O'| + |\mathcal{L}|$.

Let $W = I \cup O$. We now classify the various types of edges in the support E':

1. Let E_0 be the set of edges $(u, v) \in E'$ such that $u \notin O$ and $v \notin I$. There are the edges which do not affect the degree bounds.
2. Let E_+ be the set of edges $(u, v) \in E'$ such that $u \in O$ and $v \notin I$. Similarly, let E_- denote the set of edges for which $v \in I$ but $u \notin O$.
3. Let $E_\pm$ be the remaining edges in E' that have both $u \in O$ and $v \in I$.

For an edge e, define the spare $\mathrm{Sp}(e) = 1 - x_e$. For a set H of edges, define $\mathrm{Sp}(H) = \sum_{e \in H}(1 - x_e)$ and $\mathrm{Val}(H) = \sum_{e \in H} x_e$. We also define

$$\mathrm{Sp}_I = \sum_{e=(u,v):v \in I} \mathrm{Sp}(e) \qquad \text{and} \qquad \mathrm{Sp}_O = \sum_{e=(u,v):u \in O} \mathrm{Sp}(e)$$

that is, the sum of spares of all incoming edges into vertices in I and the sum of spares of all outgoing edges from vertices in O respectively. Note that $\mathrm{Sp}_I \leq \mathrm{Sp}(E_-) + \mathrm{Sp}(E_\pm)$ and $\mathrm{Sp}_O \leq \mathrm{Sp}(E_+) + \mathrm{Sp}(E_\pm)$ and hence,

$$\mathrm{Sp}_I + \mathrm{Sp}_O \leq \mathrm{Sp}(E_+) + \mathrm{Sp}(E_-) + 2\mathrm{Sp}(E_\pm) \tag{3.2}$$

LEMMA 3.3. *To prove Theorem 3.1, it suffices to show that*

$$2|\mathcal{L}| < 2|E_0| + |E_+| + \mathrm{Val}(E_+) + |E_-| + \mathrm{Val}(E_-) + \mathrm{Val}(E_\pm) + 3|W| \tag{3.3}$$

Proof. Since $|E| = |\mathcal{L}| + |T'| + |T''| \leq |\mathcal{L}| + |I| + |O|$ and $|W| \leq |I| + |O|$, the inequality (3.3) implies that

$$2|E| < 2|E_0| + |E_+| + \mathrm{Val}(E_+) + |E_-| + \mathrm{Val}(E_-) + \mathrm{Val}(E_\pm) + 5|I| + 5|O| \tag{3.4}$$

As $|E| = |E_0| + |E_+| + |E_-| + |E_\pm|$, (3.4) can be written as

$$|E_+| + |E_-| + 2|E_\pm| < \mathrm{Val}(E_+) + \mathrm{Val}(E_-) + \mathrm{Val}(E_\pm) + 5|I| + 5|O|. \tag{3.5}$$

As $\mathrm{Sp}(X) = |X| - \mathrm{Val}(X) \leq |X|$ for any subset of edges X, the inequalities (3.5) and (3.2) imply that

$$\mathrm{Sp}_I + \mathrm{Sp}_O \leq \mathrm{Sp}(E_+) + \mathrm{Sp}(E_-) + 2 \cdot \mathrm{Sp}(E_\pm) < 5|I| + 5|O|.$$

This implies that either there is $v \in I$ with $\sum_{e \in \delta^-(v)} \mathrm{Sp}(e) < 5$ or there is $v \in O$ with $\sum_{e \in \delta^+(v)} \mathrm{Sp}(e) < 5$. This, in turn, implies that either the condition in step (4) holds for some $v \in I$ or the condition in step (5) holds for some $v \in O$, which will prove Theorem 3.1. □

Our goal now is to prove (3.3).

3.1. Token assignment: Proof of inequality (3.3). The proof of (3.3) is done via a "token" assignment scheme. We give some tokens to the edges in E' and vertices in W so that the total number of tokens equals the RHS of (3.3). We then reassign these tokens to obtain at least 2 tokens per node in $\mathcal{L}$; leaving at least one token unassigned, thereby proving (3.3).

We give 2 tokens to each edge $e = (u, v) \in E_0$. Of these, $1 + x_e$ units "lie" at the head v, and $1 - x_e$ tokens "lie" in the "middle" of the edge. We give $1 + x_e$ tokens

to each edge $e \in E_+ \cup E_-$. For an edge $(u,v) \in E_+$, the $1 + x_e$ tokens lie at the head v. For an edge $(u,v) \in E_-$, the x_e tokens lie at the head v and 1 token lies in the middle. The remaining edges $e = (u,v) \in E_\pm$ are given x_e tokens that lie at the head v. We also give 3 tokens to each W-vertex. The tokens lying at a vertex are initially assigned to the inclusion-wise minimal set in $\mathcal{L}$ that contains that vertex; while the tokens in the middle of an edge are assigned to the inclusion-wise minimal set in $\mathcal{L}$ that contains both end-points of that edge.

We call a node $S \in \mathcal{L}$ *marked* if $W \cap S \neq \emptyset$; or *unmarked* otherwise. Note that for any $S \in \mathcal{L}$, we have $x(\delta^-(S)) \geq 1$ and is an integer. The reassignment of tokens to nodes of $\mathcal{L}$ proceeds using the following steps.

3.1.1. Unmarked leaf nodes. Let $S \in \mathcal{L}$ be such a node. Since $x(\delta^-(S)) \geq 1$, there are at least two edges of E' entering S (as each edge has $x_e < 1$). Assign the tokens at the heads of these edges to S. As S is unmarked, these must be edges of type E_0 or E_+, and S receives at least $2 + x(\delta^-(S)) \geq 3$ tokens. One extra token of these nodes is going to be reassigned to other nodes in $\mathcal{L}$ as described later.

3.1.2. Unmarked non-leaf nodes. Let $S \in \mathcal{L}$ be such a node, and $C_1, \cdots, C_t \subset S$ its children. Let $z = x(E'(V \setminus S, S \setminus \cup_{i=1}^t C_i))$ denote the total x-value entering $S \setminus \cup_{i=1}^t C_i$ from outside S. See also Figure 3.1.

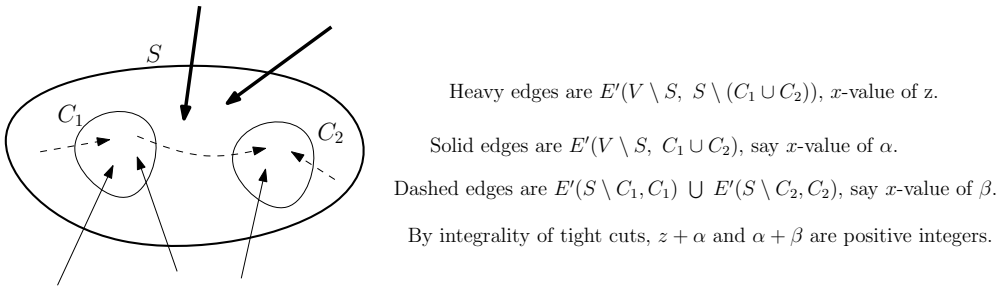

FIG. 3.1. *Unmarked node S with $t = 2$ children (illustration for Section 3.1.2).*

We first consider the case when $z > 0$. Note that edges in $E'(V \setminus S, S \setminus \cup_{i=1}^t C_i)$ lie either in E_0 or E_+, thus if $z > 0$, then they contribute at least $1 + z$ tokens to S. Thus, if $z \geq 1$, then S obtains two tokens from them. Now, suppose that $z < 1$. By integrality of the tight cuts, it follows that $\sum_{i=1}^t x(E'(S \setminus C_i, C_i)) \geq z$. Since these are all edges in E_0, they contribute at least $1 - z$ middle tokens to S. Thus S gets at least $(1 + z) + (1 - z) = 2$ tokens.

We now consider the case $z = 0$. By linear independence it follows that $\sum_i \chi_{\delta^-(C_i)} \neq \chi_{\delta^-(S)}$. By the integrality of connectivity requirements and since $z = 0$, it follows that $\sum_i x(\delta^-(C_i)) - x(\delta^-(S)) \geq 1$ and is an integer. I.e. $\sum_{i=1}^t x(E'(S \setminus C_i, C_i)) = k$ where $k \geq 1$ is an integer. Note that each edge $e \in \cup_{i=1}^t E'(S \setminus C_i, C_i)$ is a type E_0 edge and contributes $1 - x_e$ middle tokens to S; hence S receives a total of at least $|\cup_{i=1}^t E'(S \setminus C_i, C_i)| - k$ middle tokens. Moreover, $|\cup_{i=1}^t E'(S \setminus C_i, C_i)| \geq 2k+1$ since the x-value of any E_0-edge is less than $\epsilon \leq \frac{1}{2}$, and $x(\cup_{i=1}^t E'(S \setminus C_i, C_i)) = k$. Thus S receives at least $k + 1 \geq 2$ middle tokens.

3.1.3. Marked nodes. Let $\mathcal{M} \subseteq \mathcal{L}$ denote the laminar family consisting of only marked nodes. Call a node $S \in \mathcal{M}$ *high-degree* if it has at least 2 children in $\mathcal{M}$; *low-degree* if it has exactly 1 child in $\mathcal{M}$; and *leaf* if it has no children in $\mathcal{M}$. We now

show how to assign tokens to each of these nodes.

High-degree nodes. Note that the number of high-degree nodes in $\mathcal{M}$ is strictly less than the number of leaf-nodes in $\mathcal{M}$. Arbitrarily assign each high-degree node in $\mathcal{M}$ one token each from a distinct W-vertex (in a distinct leaf node of $\mathcal{M}$). This will provide at least two tokens.

Leaf-nodes. For each leaf node S in $\mathcal{M}$, we assign 1 token from some W-vertex contained in it. For the remaining token, we argue as follows: If S is also a leaf in $\mathcal{L}$, then S has $x(\delta^-(S)) \geq 1$ and hence S receives at least 1 unit of tokens from edges in $\delta^-(S)$ (since every edge carries at least x_e tokens at its head). If S is not a leaf in $\mathcal{L}$, then consider the subtree rooted S. This subtree has at least one unmarked leaf node. Since each unmarked leaf node has at least 3 tokens assigned to it thus far, S borrows one token arbitrarily from one of these nodes. Note that any unmarked leaf node can be charged at most once.

Also note that each W-vertex has been charged at most 3 tokens so far.

Low-degree marked nodes. Let $S \in \mathcal{M}$ be such a node, and $C \in \mathcal{M}$ be its unique child.

Suppose that $W \cap (S \setminus C) \neq \emptyset$, and $w \in W \cap (S \setminus C)$ be such a vertex. As no node of $\mathcal{M}$ is contained in $S \setminus C$, S is the smallest set in $\mathcal{M}$ that contains w. Assign node S two tokens from vertex w. Note that this vertex w cannot be charged by more than one such set S in this step. Moreover, w could not have been used in the earlier charging to W-vertices since it is not contained in any leaf node of $\mathcal{M}$.

Henceforth we assume that $W \cap (S \setminus C) = \emptyset$. Let r denote the number of unmarked leaves of $\mathcal{L}$ contained in $S \setminus C$. Consider the following cases:

1. $r = 0$. In this case, there are no unmarked nodes in $S \setminus C$. Let $z = x(E'(V \setminus S, S \setminus C))$ denote the total x-value entering $S \setminus C$ from outside S. We first consider the case when $z = 0$. By linear independence it follows that $\chi_{\delta^-(C)} \neq \chi_{\delta^-(S)}$. By the integrality of connectivity requirements and since $z = 0$, it follows that $x(\delta^-(C)) - x(\delta^-(S)) \geq 1$ is an integer. Consider the edges $E'(S \setminus C, C)$. They must be either E_0 or E_- edges as $S \setminus C$ does not have a W-vertex. If they are all E_0 edges, then their middle tokens must contribute at least 2 tokens to S. If at least two of them are E_- edges, their middle tokens also contribute at least two tokens to S. If there is exactly one E_- edge, then it has x-value strictly less than $1 - \epsilon$. Since edges in E_0 have x-value less than ϵ, we need at least two more edges from E_0 (and each has at least $\frac{1}{2}$ middle tokens) to ensure that $x(\delta^-(C)) - x(\delta^-(S)) \geq 1$. These edges together provide the two tokens for S.
   We now consider the case when $z > 0$. The edges in $E'(V \setminus S, S \setminus C)$ are either E_0 or E_+ edges, so they contribute at least $1 + z$ tokens to S. Thus, if $z \geq 1$, then S obtains two tokens from them. Now, suppose that $z < 1$. By integrality of the tight cuts, it follows that at least z amount of x-value must also enter C from $S \setminus C$. Since these are either E_0 or E_- edges, they contribute at least $1 - z$ tokens to S. Thus together S has at least $(1 + z) + (1 - z) = 2$ tokens.
2. $r \geq 2$. Consider the unmarked leaf nodes in $\mathcal{L}$ contained in $S \setminus C$. Note that each of them has been assigned at least 3 tokens thus far (they could not have given a token to handle marked leaf node in the previous step). S is assigned 2 tokens by borrowing 1 token each from any two unmarked leaf nodes in $S \setminus C$.

3. $r = 1$. In this case, $\mathcal{L} \cap (S \setminus C)$ corresponds to a chain of $k \geq 1$ unmarked nodes $\mathcal{D} = \{D_k \subseteq D_{k-1} \subseteq \cdots \subseteq D_1\}$. Let $D = D_1$ be the unmarked child of S. We first consider the case that there is an edge e from $V \setminus S$ to $S \setminus (C \cup D)$. Here, the edge e provides at least 1 token to S. For the remaining token, we observe that the subtree rooted at D in $\mathcal{L}$ has at least one unmarked leaf (node D_k). This node still has at least 3 tokens since none of its tokens could have been used for earlier reassignments. Thus S can borrow 1 token from D and get at least 2 tokens.
   Henceforth, we assume that all edges from $V \setminus S$ enter either C or D. Suppose that some unmarked node (say D_i) in the chain $\mathcal{D}$ has a cut value more than 1 (i.e., $x(\delta^-(D_i)) = f(D_i) - |\delta_F^-(D_i)| \geq 2$). In this case, we use the following Claim which is proved at the end of this section.
   CLAIM 1. *Let $\mathcal{D} = \{D_k \subseteq D_{k-1} \subseteq \cdots \subseteq D_1\}$ be a chain of unmarked nodes with D_k being a leaf node. Then a total of at least $2k + x(\delta^-(D_1))$ tokens are assigned to nodes of $\mathcal{D}$.* Applying Claim 1 to the chain $\mathcal{D}' = \{D_i, \cdots, D_k\}$, we obtain that at least $2(k - i + 1) + 2$ tokens are assigned to the nodes of $\mathcal{D}'$. Thus there are at least 2 extra tokens, which can be reassigned to node S (note that these unmarked nodes have not been used in earlier reassignments).
   In the remaining, we assume that all nodes in $\mathcal{D}$ have cut value exactly 1. Let $z = x(E'(V \setminus S, D))$ be the x-value entering D from $V \setminus S$; note that $0 \leq z \leq 1$ since D has cut value 1. We consider the following four cases.
   Case 1: $z = 0$. In this case, $\delta^-(S) \subseteq \delta^-(C)$. From linear independence and integrality of the cut values, this implies $x(E'(S \setminus C, C)) \geq 1$. Hence as in step 1, S obtains at least 2 middle tokens from $E'(S \setminus C, C)$ (which are type E_0 or E_- edges).
   Case 2: $0 < z < \epsilon$. In this case, $x(E'(S \setminus D, D)) = 1 - z > 1 - \epsilon$. Since every edge has x-value less than $1 - \epsilon$, $|E'(S \setminus D, D)| \geq 2$. Also, $|E'(V \setminus S, D)| \geq 1$ since $z > 0$. Thus $|\delta^-(D)| \geq 3$. We now use the following Claim which is again proved at the end of this section.
   CLAIM 2. *Let $\mathcal{D} = \{D_k \subseteq D_{k-1} \subseteq \cdots \subseteq D_1\}$ be a chain of unmarked nodes with D_k being a leaf node, such that each node D_i has cut value $x(\delta^-(D_i)) = 1$. Then a total of at least $2(k-1) + |\delta^-(D_1)| + 1$ tokens are assigned to nodes of $\mathcal{D}$.* Now applying Claim 2 to chain $\mathcal{D}$, there are at least $2k + 2$ tokens assigned to the nodes of $\mathcal{D}$. Since there are 2 extra tokens, these can be reassigned to S.
   Case 3: $\epsilon \leq z < 1$. From the integrality of the cut values of S and C, $x(E'(S \setminus C, C)) \geq z \geq \epsilon$. Since each edge in $E'(S \setminus C, C)$ is type E_0 or E_-, $E'(S \setminus C, C)$ has either at least one E_- edge or at least two E_0 edges (each has x-value less than ϵ). In either case S obtains at least 1 unit of middle tokens. Borrowing one token from the unmarked leaf D_k, S is assigned at least 2 tokens.
   Case 4: $z = 1$. Here it must be that $x(E'(S \setminus C, C)) \geq 1$: this follows from the linear independence and integrality of cuts S, C, D and the fact that $x(\delta^-(D)) = 1$. As in step 1, S has at least 2 units of middle tokens.

Thus the proof of inequality (3.3) is complete. We now present the proofs of Claims 1 and 2.

PROOF OF CLAIM 1. Note that every edge (u, v) induced on D_1 is an E_0 edge and has 2 tokens: we think of it having one token at each of u and v. Every edge (u, v) in $\delta^-(D_1)$ is of type E_0 or E_+ and has $1 + x_{(u,v)}$ tokens at $v \in D_1$: we think of $x_{(u,v)}$

units contributing to the $x(\delta^-(D_1))$ term and the remaining one token lying at v. It now suffices to show that the total number of *end-points* of the support E' inside D_1 is at least $2k$. We claim that for every $1 \le i \le k$, $D_i \setminus D_{i+1}$ has at least 2 end-points (setting $D_{k+1} = \emptyset$). First consider D_k: since $x(\delta^-(D_k)) \ge 1$ there are at least 2 edges entering D_k that contribute the 2 (head) end-points. Now consider node D_i and its child D_{i+1}: let $z = x(V \setminus D_i, D_i \setminus D_{i+1})$ and consider the following cases.

1. $z = 0$. Due to linear independence and integrality of D_i and D_{i+1}, we have $x(D_i \setminus D_{i+1}, D_{i+1}) \ge 1$, which gives at least 2 (tail) end-points.
2. $0 < z < 1$. This immediately gives at least 1 (head) end-point. Also we have $x(D_i \setminus D_{i+1}, D_{i+1}) \ge z$ (same reasons as above) which gives at least 1 (tail) end-point.
3. $z \ge 1$. Here $|E'(V \setminus D_i, D_i \setminus D_{i+1})| \ge 2$ which gives at least 2 (head) end-points.

In each case, we have at least 2 end-points in $D_i \setminus D_{i+1}$. Thus we have the claim.

Proof of Claim 2. We first show that $|E'(D_i \setminus D_{i+1}, D_{i+1})| \ge 1$ for all $1 \le i < k$. Consider any node D_i ($1 \le i < k$) and its child D_{i+1}. Since $x(\delta^-(D_i)) = x(\delta^-(D_{i+1})) = 1$, using linear independence it follows that there must be an edge in $E'(D_i \setminus D_{i+1}, D_{i+1})$. These $k-1$ edges (all type E_0) provide $2(k-1)$ tokens. Together with the tokens on edges of $\delta^-(D_1)$ (that total to at least $|\delta^-(D_1)|+1$ since each such edge contributes $(1 + x_e)$ tokens), we have the claim.

4. Crossing supermodular connectivity with costs. In this section, we note an immediate consequence of Theorem 3.1 to the more general case of crossing supermodular connectivity requirements with degree bounds.

Theorem 4.1. *For any $\epsilon \in [0, \frac{1}{2}]$, there is a polynomial time $(\lceil \frac{a_v}{1-\epsilon} \rceil + 4 + f_{\max}, \lceil \frac{b_v}{1-\epsilon} \rceil + 4 + f_{\max}, \frac{2}{\epsilon})$ approximation algorithm for degree-bounded network design with crossing supermodular requirement f, where $f_{max} = \max_{S \subseteq V} f(S)$.*

We begin with the following lemma which upper bounds the in-degree of any vertex in a minimal f-connected subgraph when f is intersecting supermodular. Consider a directed graph $G = (V, E)$ with an intersecting supermodular requirement function $f : 2^V \to \mathbb{Z}^+$ on V. Let $f_{\max} = \max_{S \subseteq V} f(S)$ be the maximum requirement of any set. A subgraph $H = (V, E')$ of G is called *minimally* f-connected if H is f-connected and no strict subgraph of H is f-connected.

Lemma 4.2. *Let H be a minimally f-connected subgraph of G. Then the in-degree of any vertex v in H is at most $f_{\max}$, i.e., $|\delta^-_H(v)| \le f_{\max}$.*

Proof. Fix a vertex $v \in V$ and let $\delta^-_H(v) = \{(u_i, v) \mid i = 1, \ldots, k\}$. Since H is minimally f-connected, each edge (u_i, v) belongs to a tight cut constraint, i.e., there exist subsets $S_1, \ldots, S_k \subset V$ such that for all $1 \le i \le k$ we have $(u_i, v) \in \delta^-_H(S_i)$ and $|\delta^-_H(S_i)| = f(S_i)$. Note that $v \in S_i$ and $u_i \notin S_i$ for all i.

We next use the fact that if two subsets $S, S' \subset V$ intersect and are tight, i.e., $|\delta^-_H(S)| = f(S)$ and $|\delta^-_H(S')| = f(S')$, then their intersection is also tight: $|\delta^-_H(S \cap S')| = f(S \cap S')$. This follows since the following chain of inequalities must hold with equalities: $f(S \cup S') + f(S \cap S') \ge f(S) + f(S') = |\delta^-_H(S)| + |\delta^-_H(S')| \ge |\delta^-_H(S \cup S')| + |\delta^-_H(S \cap S')| \ge f(S \cup S') + f(S \cap S')$.

Applying this to the subsets $S_1, \ldots, S_k$ repeatedly, we get that their intersection $\cap_{i=1}^k S_i$ is also tight. Since $(u_i, v) \in \delta^-_H(\cap_{i=1}^k S_i)$ for each i, we get $k \le |\delta^-_H(\cap_{i=1}^k S_i)| = f(\cap_{i=1}^k S_i) \le f_{\max}$. □

We now prove Theorem 4.1.

PROOF OF THEOREM 4.1. We use Theorem 3.1, Lemma 4.2 and a reduction from crossing supermodular requirements to intersecting supermodular requirements as in [15]. Let OPT denote the cost of the optimal f-connected subgraph satisfying the degree bounds (a_v, b_v). Let $r \in V$ be an arbitrary but fixed vertex. Define two functions $g, h : 2^V \to \mathbb{Z}_+$ as follows:

$$g(S) = \begin{cases} 0 & \text{if } r \in S \\ f(S) & \text{otherwise,} \end{cases} \qquad h(S) = \begin{cases} 0 & \text{if } r \in S \\ f(V \setminus S) & \text{otherwise.} \end{cases}$$

It is easy to check that $f(S) = g(S) + h(V \setminus S)$ for all S and that g and h are intersecting supermodular functions on the set family $\{S \subset V \mid r \notin S\}$ (see [15] for details).

We now consider a problem on $G = (V, E)$ with the intersecting supermodular connectivity requirement g and out-degree upper bounds b_v on vertices v; and use Theorem 3.1 to compute a solution[3]. We then extract a minimal g-connected subgraph H_g from this solution by iteratively dropping edges that do not violate g-connectivity requirements. From Lemma 4.2, the in-degree of any vertex v in H_g is at most $f_{\max}$. Note that the optimal g-connected subgraph with (out-)degree bounds b_v has cost at most OPT; so the cost of H_g is at most $\frac{1}{\epsilon} \cdot \text{OPT}$.

Next we consider a problem on the graph G^r obtained by reversing all edges in G with the intersecting supermodular connectivity requirement h and out-degree upper bounds a_v on vertices v; and use Theorem 3.1 to compute a solution. We then extract a minimal h-connected subgraph H_h from this solution by iteratively dropping edges that do not violate h-connectivity requirements. From Lemma 4.2, the in-degree of any vertex v in H_h is at most $f_{\max}$. Again, the optimal h-connected subgraph in G^r with (out-)degree bounds a_v has cost at most OPT; so the cost of H_h is at most $\frac{1}{\epsilon} \cdot \text{OPT}$.

Let H_h^r be the subgraph obtained from H_h by reversing all edges. It is now easy to see that the subgraph $H_g \cup H_h^r$ of G satisfies f-connectivity requirements and the claimed degree bounds. Since each of H_g and H_h^r costs at most $\frac{1}{\epsilon} \cdot \text{OPT}$, we get the desired bound on the cost as well.

5. Integrality Gap Instance. In this section, we describe an integrality gap for the LP relaxation of the degree-bounded arborescence problem.

THEOREM 5.1. *For any $0 < \epsilon < 1$, there is an instance of the minimum-cost degree-bounded arborescence problem such that, any arborescence with out-degrees at most $\frac{b_v}{(1-\epsilon)} + O(1)$ for all vertices v has cost at least $(\frac{1-o(1)}{\epsilon})$ times the optimal LP value.*

Given an arbitrarily small but fixed constant $\epsilon \in (0, 1)$, set $\delta = \epsilon + \epsilon^c$ where c is a sufficiently large constant independent of ϵ. Consider a directed graph $G(\delta)$ constructed as follows. See Figure 5.1 for an illustration. Start with a complete k-ary outward directed tree T rooted at vertex r, with t levels (the solid edges in Figure 5.1), where we set $k = 1/\delta^{2c}$ and $t = c\delta^{-c-1} \ln(2/\delta)$. These tree edges, called T-edges, have cost 0. Consider the natural drawing of the tree on the plane (as in Figure 5.1) and label the leaves from right to left as $1, \ldots, k^t$. The vertices of T are naturally partitioned into levels $0, 1, \ldots, t$ such that the root is at level 0 and the leaves are at level t. We also label the vertices on level i as $1, \ldots, k^i$ in the right to left order. For a vertex v, let T_v denote the subtree rooted at v and let r_v and l_v

[3]It is easy to test if the given graph is g-connected by adding $f_{\max}$ edges from every vertex to r and testing if the resulting graph is f-connected. A similar reduction also holds for h-connectivity.

denote the smallest and largest indices of leaves in T_v (formally if v is the jth node from the right on level i, then $l_v = jk^{t-i}$ and $r_v = (j-1)k^{t-i}+1$).

We add the following additional edges to obtain $G(\delta)$. For each internal vertex v, we add an edge from the leaf l_v to v (these are the light dotted edges in Figure 5.1). All these edges also have cost 0. Finally, we add a path from the root, visiting the leaves in the order $1, \ldots, k^t$ (these are the heavy dashed edges) and each of these edges has cost 1.

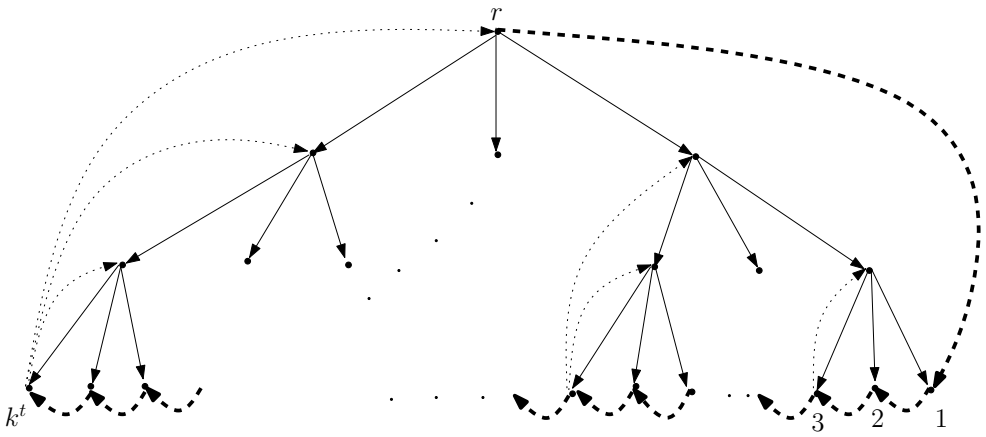

FIG. 5.1. *The integrality gap instance with $k = 3$, $t = 3$. Solid arcs (on complete t-level k-ary tree T) have cost 0 and LP-value $1-\delta$. Dotted arcs have cost 0 and LP-value δ. Heavy dashed arcs have cost 1 and LP-value δ.*

Intuitively, the graph is a union of two arborescences rooted at r: The first arborescence is the tree T, and the second arborescence is formed by the dotted and dashed edges. The first arborescence has high degree and low cost, while the second has low degrees and high cost.

Consider the problem of constructing the minimum-cost arborescence rooted at r, where each internal vertex has an upper bound of $b = (1-\delta)k$ on the out-degree. Consider a fractional assignment to the edges where each (solid) edge in T has value $x_e = 1-\delta$ and every other edge has value $x_e = \delta$. Observe that each vertex receives 1 unit of flow from the root and the fractional out-degree of each internal vertex is $(1-\delta)k$ and hence this is a feasible LP solution with cost $LP^* = \delta k^t$.

We now show that any integral solution I where the degree at each internal vertex is at most $b/(1-\epsilon) + O(1)$ has cost at least $(1-o(1))LP^*/\epsilon$. The idea is that any solution with maximum internal out-degree smaller than k, must necessarily choose all of heavy dashed edges, thereby incurring a high cost. The crucial observation is the following.

PROPOSITION 5.2. *Suppose a leaf ℓ does not have a path from root to itself in I using only T-edges, then the edge $(\ell-1, \ell)$ must necessarily lie in I.*

Proof. To see this, consider the unique path from r to ℓ in T and let (u, v) be some edge along this path that does not lie in I (such an edge must exist since ℓ is unreachable from r using T-edges). Let L denote the set of leaves $\{\ell, \ldots, l_v\}$, and let S_v denote the set of all nodes in T_v from which some vertex in L can be reached using T-edges. We claim that the (heavy dashed) edge $(\ell-1, \ell)$ is only edge in I entering the set S_v. Indeed, no T-edge enters S_v since $(u, v) \notin I$. Moreover, no dotted edge enters S_v since such edge must be of the form (ℓ', w) where ℓ' is a leaf not in S_v and hence

$\ell' \in \{r_v, \ldots, \ell-1\}$ and $w \in S_v$. Now by the construction of dotted edges in $G(\delta)$, this means that $\ell' = l_w$, and hence $\ell_w \in \{r_v, \ldots, \ell-1\}$. But the only leaves reachable by T-edges from w have indices at most ℓ_w which is at most $\ell-1$; this implies that none of the leaves in L can be reached by w which contradicts that $w \in S_v$. Thus, $(\ell-1, \ell)$ is unique edge entering S_v and must necessarily lie in I. □

To finish the proof, consider the solution I where each internal vertex has degree at most $b/(1-\epsilon) + O(1) = (1-\delta)k/(1-\epsilon) + O(1) = (1-\epsilon^c/(1-\epsilon))k + O(1)$ which is at most $(1-\delta^{c+1})k$. Thus the total number of leaves that have a path from root r using T-edges is at most $(1-\delta^{c+1})^t k^t \leq (\delta/2)^c k^t < \epsilon^c k^t$. Thus by the above claim, at least $(1-\epsilon^c)k^t$ cost 1 edges must lie in I, which implies that the total cost is at least $(1-\epsilon^c)k^t = ((1-\epsilon^c)LP^*)/\delta = (1-\epsilon^c)LP^*/(\epsilon+\epsilon^c) \geq (1-2\epsilon^{c-1})LP^*/\epsilon$. Since c is arbitrarily large, this implies the result.

From the above example, we see that to achieve a purely additive $O(1)$ guarantee for degree using the LP (3.1), the cost has to be violated by a factor at least $\Omega(\frac{\log n}{\log\log n})$, where n is the number of vertices in the graph.

6. Minimum crossing spanning tree problem. We consider the MCSP problem in this section, for which we obtain the following.

THEOREM 6.1. *There is a polynomial-time algorithm that for any instance $\langle G, c, \{E_i, b_i\}_{i=1}^k\rangle$ of the MCSP problem, either computes a spanning tree of cost at most the optimum and with at most $b_i + r - 1$ edges from E_i (for all $1 \leq i \leq k$); or shows that the instance is infeasible. Here $r = \max_{e\in E} |\{i \mid e \in E_i, 1 \leq i \leq k\}|$, is the maximum number of sets $\{E_i\}$ that any edge lies in.*

Our algorithm is again based on iterative relaxation. We either choose or delete edges, or drop some constraints. Consider a general iteration. Let E denote the candidate edges which are not yet discarded, let $F \subseteq E$ denote the set of edges that we have already picked in our solution, and let $W \subseteq \{i \mid 1 \leq i \leq k\}$ denote the indices of the crossing constraints corresponding to E_i that we have not yet dropped. In the beginning E is the entire edge-set, $F = \emptyset$, and $W = \{i \mid 1 \leq i \leq k\}$. In a general iteration, we work with the following linear relaxation $P(E, F, W)$ with variables x_e for $e \in E' = E \setminus F$.

$$
\begin{array}{lll}
\min & \sum_{e\in E'} c_e \cdot x_e & \\
s.t. & & \\
& x(E'(V)) = V - 1 - |F(V)| & \\
& x(E'(S)) \leq S - 1 - |F(S)| & \forall S : 2 \leq |S| \leq |V| - 1 \\
& x(E' \cap E_i) \leq b_i - |F \cap E_i| & \forall i \in W \\
& 0 \leq x_e \leq 1 & \forall e \in E' = E - F
\end{array}
$$

where $H(S)$ (for $H \subseteq E$ and $S \subseteq V$) is the set of edges in H with both end-points in S. In this iteration, the algorithm computes a basic feasible solution x to $P(E, F, W)$ and performs one of the following steps while $E' = E \setminus F \neq \emptyset$:

1. If there is an edge $e \in E'$ with $x_e = 0$, set $E \leftarrow E \setminus \{e\}$.
2. If there is an edge $e \in E'$ with $x_e = 1$, set $F \leftarrow F \cup \{e\}$.
3. If for some $i \in W$, $|E' \cap E_i| \leq b_i - |F \cap E_i| + r - 1$, i.e. $|E \cap E_i| \leq b_i + r - 1$, set $W \leftarrow W \setminus \{i\}$.

It is clear that if the algorithm terminates, it terminates with a set F containing a spanning tree with cost at most the optimum and which contains at most $b_i + r - 1$ edges from E_i for $1 \leq i \leq k$.

We now argue that in each iteration, one of the above steps is always applicable. The following lemma follows by uncrossing [8, 18].

LEMMA 6.2. *For any basic solution x to $P(E,F,W)$ such that $0 < x_e < 1$ for all $e \in E'$, there exists a set $T \subseteq W$ and a laminar family $\mathcal{L}$ of subsets of V such that x is the unique solution to the linear system:*

$$\begin{aligned} x(E'(S)) &= |S| - 1 - |F(S)| \quad \forall S \in \mathcal{L} \\ x(E' \cap E_i) &= b_i - |F \cap E_i| \quad \forall i \in T \end{aligned}$$

Furthermore, the characteristic vectors $\{\chi_{E'(S)} \mid S \in \mathcal{L}\} \cup \{\chi_{E' \cap E_i} \mid i \in T\}$ are linearly independent, and the size of the support $|E'| = |T| + |\mathcal{L}|$.

Assume that the conditions in steps (1) and (2) do not hold; then we prove that step (3) holds. The key component of our proof is the following lemma which is proved by a simple counting argument.

CLAIM 3. *We have $|\mathcal{L}| \leq x(E'(V))$. Moreover the equality holds if and only if each edge in E' is contained in some inclusion-wise maximal set $S \in \mathcal{L}$.*

Proof. Suppose each edge $e \in E'$ is given x_e tokens. These tokens are assigned to the sets $S \in \mathcal{L}$ as follows. An edge e is said to belong to S if S is the inclusion-wise minimal set in $\mathcal{L}$ that contains both the end-points of e. If e belongs to S, then x_e tokens are assigned to S. We argue that each set in the laminar family is assigned a total of unit tokens, thereby proving the claim.

Since $x_e > 0$ for all $e \in E'$, each set $S \in \mathcal{L}$ has the RHS $|S| - 1 - |F(S)|$ at least 1, and hence $x(E'(S)) \geq 1$. This gives every *leaf* set $S \in \mathcal{L}$ at least a total of unit tokens. Now consider a *non-leaf* set $S \in \mathcal{L}$ with t children $C_1, \cdots, C_t \in \mathcal{L}$. Now $\chi_{E'(S)} = \sum_{j=1}^{t} \chi_{E'(C_j)} + \sum\{\chi_e \mid e \in E' \text{ belongs to } S\}$. Since $\chi_{E'(S)} \cup \{\chi_{E'(C_j)}\}_{j=1}^{t}$ is a linearly independent set, we have $\{e \mid e \in E' \text{ belongs to } S\} \neq \emptyset$. So, the RHS $|S| - 1 - |F(S)|$ of the constraint for S is at least 1 more than the sum of the RHS of constraints of $\{C_j\}_{j=1}^{t}$. Thus S gets at least a total of unit tokens. □

Now for $i \in W$, define $\mathrm{Sp}(i) = \sum_{e \in E' \cap E_i} (1 - x_e) = |E' \cap E_i| - x(E' \cap E_i)$ and for $e \in E'$, define $r(e) = |\{i \in W \mid e \in E' \cap E_i\}|$.

LEMMA 6.3. *We have $\sum_{i \in W} \mathrm{Sp}(i) < r|W|$.* Before proving Lemma 6.3, we argue that it implies that the condition in step (3) holds. Lemma 6.3 implies that there exists $i \in W$ such that $\mathrm{Sp}(i) < r$. Since $x(E' \cap E_i) \leq b_i - |F \cap E_i|$, we have

$$|E' \cap E_i| = \mathrm{Sp}(i) + x(E' \cap E_i) < r + b_i - |F \cap E_i|.$$

Since $|E' \cap E_i|$ and $|F \cap E_i|$ are integers, $|E' \cap E_i| \leq r + b_i - |F \cap E_i| - 1$, i.e., the condition in step (3) holds for i.

Proof. (Lemma 6.3) Lemma 6.2 and Claim 3 imply that

$$\sum_{e \in E'} (1 - x_e) = |E'| - x(E'(V)) = |\mathcal{L}| + |T| - x(E'(V)) \leq |T| = |W| - |W \setminus T|.$$

Therefore

$$\begin{aligned} \sum_{i \in W} \mathrm{Sp}(i) &= \sum_{e \in E'} r(e)(1 - x_e) \\ &= r \sum_{e \in E'} (1 - x_e) - \sum_{e \in E'} (r - r(e))(1 - x_e) \\ &\leq r|W| - r|W \setminus T| - \sum_{e \in E'} (r - r(e))(1 - x_e). \end{aligned}$$

Moreover, the equality $\sum_{i \in W} \mathrm{Sp}(i) = r|W|$ holds if and only if $|\mathcal{L}| = x(E'(V))$, $W = T$ and $r = r(e)$ for each edge $e \in E'$. The final requirement $r = r(e)$ follows as $x_e < 1$ and hence $(1 - x_e) > 0$. We will show that this cannot happen, since this violates the linear independence condition. In particular, we will show that the incidence vector $\chi_{E'}$ can be expressed in two different ways using the characteristic vectors $\{\chi_{E'(S)} \mid S \in \mathcal{L}\} \cup \{\chi_{E' \cap E_i} \mid i \in T\}$.

First, by Claim 3 (equality condition), we have that $\sum_{i=1}^{p} \chi_{E'(S_i)} = \chi_{E'}$, where $S_1, \ldots, S_p$ are the inclusion-wise maximal sets in $\mathcal{L}$. Second, since $|T| = |W|$ and $r(e) = r$ for all $e \in E'$, we have that

$$\sum_{i \in T} \chi_{E' \cap E_i} = \sum_{i \in W} \chi_{E' \cap E_i} = r \cdot \chi_{E'}.$$

This gives us the desired contradiction which completes the proof. □

Generalization to matroids and polymatroids. Király et al. [10] consider the problem of computing a minimum-cost basis in a matroid subject to 'degree bounds' and show that the above algorithm generalizes directly to this case.

Acknowledgments. We thank Zhenghua Fu for suggesting to us the problem of packing arborescences subject to degree bounds, which started this project. We also thank Mohit Singh and R. Ravi for several useful discussions.

REFERENCES

[1] Jørgen Bang-Jensen, Stphan Thomass, and Anders Yeo. Small degree out-branchings. *Journal of Graph Theory*, 42(4), 297-307, 2003.

[2] Vittorio Bilò, Vineet Goyal, R. Ravi, and Mohit Singh. On the crossing spanning tree problem. In Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, 51–60.

[3] 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 33rd International Colloquium on Automata, Languages and Programming (ICALP), 191–201, 2006.

[4] J. Edmonds. Matroid Intersection in Discrete Optimization I. *Annals of Discrete Mathematics*, 4:39–49, 1979.

[5] Andras Frank. Increasing the rooted-connectivity of a digraph by one. *Mathematical Programming (B)*, 84(3):565–576, 1999.

[6] M. Furer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994.

[7] Harold N. Gabow. On the L_∞-Norm of Extreme Points for Crossing Supermodular Directed Network LPs. *Mathematical Programming (B)*, 110(1):111–144, 2007.

[8] Michel X. Goemans. Minimum bounded degree spanning trees. In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 273–282, 2006.

[9] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, pages 39–61, 2001.

[10] Tamás Király, Lap Chi Lau, and Mohit Singh. Degree bounded matroids and submodular flows. In Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 259–272, 2008.

[11] Philip N. Klein, Radha Krishnan, Balaji Raghavachari, and R. Ravi. Approximation algorithms for finding low-degree subgraphs. *Networks*, 44(3):203–215, 2004.

[12] J. Konemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree bounded minimum spanning trees. *SIAM Journal on Computing*, 31(3):1783–1793, 2002.

[13] Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints (Full version). In Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), 651–660, 2007.

[14] E.L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9:31–56, 1975.

[15] Vardges Melkonian and Éva Tardos. Algorithms for a Network Design Problem with Crossing Supermodular Demands. *Networks*, 43(4), 256-265, 2004.

[16] Viswanath Nagarajan, R. Ravi, and Mohit Singh. Unified Analysis of LP Extreme Points for Steiner Networks and Traveling Salesman. *Manuscript*, 2008.

[17] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1), 58-78, 2001.

[18] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), 661–670, 2007.