Additive Approximation for Bounded Degree Survivable Network Design

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

Mohit Singh†
Tepper School of Business
Carnegie Mellon University
mohits@andrew.cmu.edu

ABSTRACT

We study a general network design problem with additional degree constraints. Given connectivity requirements r_{uv} for all pairs of vertices, a *Steiner network* is a graph in which there are at least r_{uv} edge-disjoint paths between u and v for all pairs of vertices u, v. In the MINIMUM BOUNDED-DEGREE STEINER NETWORK problem, we are given an undirected graph G with an edge cost for each edge, a connectivity requirement r_{uv} for each pair of vertices u and v, and a degree upper bound for each vertex v. The task is to find a minimum cost Steiner network which satisfies all the degree upper bounds.

The aim of this paper is to design approximation algorithms that minimize the total cost and the degree violation simultaneously. Our main results are the following:

- There is a polynomial time algorithm which returns a Steiner forest of cost at most 2OPT and the degree violation at each vertex is at most 3, where OPT is the cost of an optimal solution which satisfies all the degree bounds.

- There is a polynomial time algorithm which returns a Steiner network of cost at most 2OPT and the degree violation at each vertex is at most $6r_{max} + 3$, where OPT is the cost of an optimal solution which satisfies all the degree bounds, and $r_{max} := max_{u,v}\{r_{uv}\}$.

These results achieve the best known guarantees for both the total cost and the degree violation simultaneously. As corollaries, these results provide the first additive approximation algorithms for finding low degree subgraphs including Steiner forests, k-edge-connected subgraphs, and Steiner networks. The algorithms develop on the iterative relaxation method applied to a natural linear programming relaxation as in [10, 16, 22]. The new algorithms avoid paying a multiplicative factor of two on the degree bounds even though the algorithm can only pick edges with fractional value $1/2$. This is based on a stronger characterization of the basic solutions of the linear programming relaxation. The analysis of the algorithm is nearly tight.

*Research supported by RGC grant 2150529.
†Research supported by NSF grant CCF-0728841.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

Approximation Algorithms, Steiner Trees, Survivable Network Design, Bounded Degree, Iterative Rounding.

1. INTRODUCTION

Network design plays a central role in combinatorial optimization and approximation algorithms. Developments in this area have led to general algorithmic techniques, and also provide useful models for practical applications. In recent years, much effort has been put on designing approximation algorithms for network design problems with additional degree constraints. These problems generalize key problems in combinatorial optimization, and also have applications in various areas including VLSI design, vehicle routing and communication networks [3, 6, 18, 20, 24]. In these applications, degree constraints occur as a natural modelling tool for *workload* of nodes. For example, in typical applications of network design problems to multicasting, the degree constraints on switches correspond to a bound on the multicast copies a switch can make in the network [3].

In this paper we study a general network design problem with additional degree constraints. Given connectivity requirements r_{uv} for all pairs of vertices, a *Steiner network* is a graph in which there are at least r_{uv} edge-disjoint paths between u and v for all pairs u, v. In the MINIMUM BOUNDED-DEGREE STEINER NETWORK problem, we are given an undirected graph G with an edge cost for each edge, a connectivity requirement for each pair of vertices, and a degree upper bound B_v for each vertex v. The task is to find a minimum cost Steiner network H of G satisfying all the degree bounds, that is, $deg_H(v) \leq B_v$ for all v. This problem captures many well-studied network design problems as special cases; for instance, a *Steiner forest* is a Steiner network with $r_{uv} \in \{0, 1\}$ for all pairs. Even the feasibility problem of finding a Steiner network satisfying all the degree bounds is already NP-hard. Hence the MINIMUM BOUNDED-DEGREE STEINER NETWORK problem has two optimization objectives: to minimize the total cost and to minimize the degree violation. The aim of this paper is to design approximation algorithms that optimizes both objectives simultaneously. Our main results are the following.

THEOREM 1.1. *There exists a polynomial time algorithm for the* MINIMUM BOUNDED-DEGREE STEINER FOREST *problem which returns a Steiner forest F of cost at most* 2OPT *with degree violation at most 3 (i.e. $d_F(v) \leq B_v + 3\, \forall v \in F$). Here* OPT *is the cost of an optimal solution which satisfies all the degree bounds.*

In this extended abstract, we prove a weaker version of Theorem 1.1 where the degree violation is at most 4. The above result can be extended to the MINIMUM BOUNDED DEGREE STEINER NETWORK problem.

THEOREM 1.2. *There exists a polynomial time algorithm for the* MINIMUM BOUNDED-DEGREE STEINER NETWORK *problem which returns a Steiner network H of cost at most* 2OPT *with degree violation at most $6r_{max} + 3$. Here* OPT *is the cost of an optimal solution which satisfies all the degree bounds, and $r_{max} := \max_{u,v}\{r_{uv}\}$.*

Previously the best known guarantees on the degree for both Steiner forest and Steiner network are $2B_v + 3$ in [16], even when there are no costs on the edges. Theorem 1.1 and 1.2 provide the first *additive* approximation algorithms that violate the degrees by at most a constant for a class of problems, including Steiner forests (+3), k-edge-connected subgraphs ($+O(k)$), and Steiner networks ($+O(r_{max})$). Moreover, these results can be achieved while simultaneously matching the best known guarantees for the minimum cost Steiner forest and Steiner network problems [1, 10], and thus provide a unifying algorithmic framework for many well-studied network design problems. We note that Theorem 1.2 can further be generalized to element connectivity - the most general model where a 2-approximation algorithm for the minimum cost Steiner network problem is known.

The algorithms develop on the iterative relaxation method applied to a linear programming relaxation as in [10, 16, 22]. The analysis on the linear programming relaxation is nearly tight. There are examples in which the optimal fractional solution has maximal degree B but any integer solution would have maximal degree at least $B + 2$ for Steiner forests [16], and at least $B + \Omega(r_{max})$ for Steiner networks as shown at the end of Section 3.

1.1 Techniques

The algorithms develop on the iterative relaxation method first used in [16, 22], which adapts Jain's iterative rounding method [10] to the MINIMUM BOUNDED-DEGREE STEINER NETWORK problem. The approach in [16] relies on the following lemma about the basic solutions of the linear program for Steiner networks: "If every vertex with a degree constraint has degree at least 5, then in any basic solution there is an edge e with $x_e \geq \frac{1}{2}$." This lemma leads to a new relaxation step to Jain's iterative rounding framework to deal with degree bounds: If there is a vertex with degree at most 4, then the degree constraint for that vertex is removed. This relaxation step only incurs an additive constant +3 on the degree bounds. After this step, we can always pick an edge with $x_e \geq \frac{1}{2}$ as in Jain's approach, and hence the cost and the degrees are violated by at most a multiplicative factor of 2. As illustrated in the example in Figure 1, the algorithm in [16] may actually violate the degree bounds by a multiplicative factor of 2.

To achieve additive approximation on the degree bounds, the challenge is to avoid paying a multiplicative factor of 2 on the degree bounds when picking edges with value $\frac{1}{2}$. Note that such edges are inevitable as a factor of 2 on the cost is best possible using the linear program relaxation for Steiner networks. In our algorithm for Steiner forests, we first generalize the relaxation step to remove the degree constraint of every vertex with degree at most $B_v + 3$, which is possible since the degree bounds would only be violated by at most 3. The main technical contribution of this paper is the following lemma about the basic solutions: "If every vertex with degree constraint has degree at least $B_v + 4$, then in any basic solution there is an edge with $x_e \geq \frac{1}{2}$ *between two vertices without degree constraints*." This leads us to a modified iterative algorithm that only selects edges with $x_e \geq \frac{1}{2}$ between two vertices without degree constraints, which is a key difference from existing iterative rounding algorithms that pick edges depending only on the fractional values. For example, in Figure 1, the algorithm will only choose edges from $\{x_i, y_i\}$ and return the solution in (c). By only choosing those edges, degree constraints would not be violated when they are present, and are only violated by at most an additive constant 3 when they are removed.

This approach can be extended to Steiner networks, by proving that there are edges with $x_e \geq \frac{1}{2}$ between two *low degree vertices* in any basic solution. Selecting these edges only results in an additive violation which depends only on the parameter r_{max}. The proofs of these characterizations of the basic solutions require new ideas on the counting argument (which has become more involved since edges with $x_e \geq \frac{1}{2}$ are not allowed to be picked if there is an endpoint with degree constraint present), and also crucially exploit the parameter r_{max}.

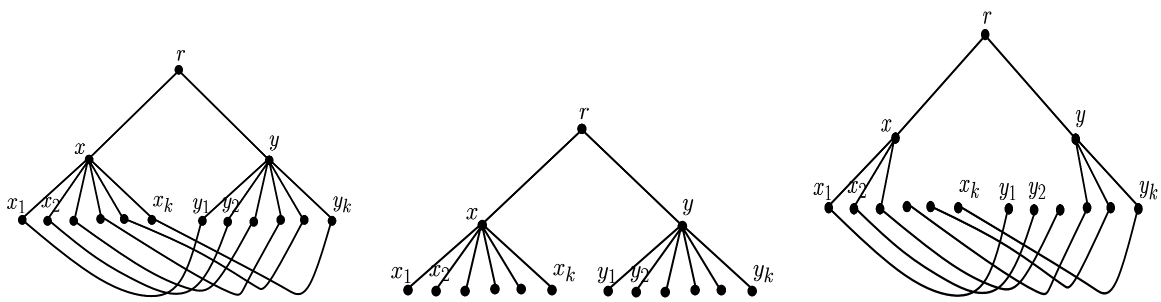
1.2 Related Work

For the MINIMUM STEINER NETWORK problem, Jain [10] introduced the iterative rounding method to obtain a 2 approximation algorithm, improving on a long line of earlier research that applied primal-dual methods to these problems. For degree-bounded Steiner trees, Fürer and Raghavachari [8] gave an approximation algorithm with degree violation at most 1. This result has generated much interest in obtaining approximation algorithms for network design problems with degree constraints [14, 15, 12, 16, 7, 19, 4, 5, 20, 21, 9, 22].

A highlight of this line of research is an $(1, B_v + 2)$- approximation algorithm[1] for the MINIMUM BOUNDED-DEGREE SPANNING TREE problem by Goemans [9]. Recently, the iterative relaxation method is used to obtain the best possible $(1, B_v + 1)$-approximation algorithm for spanning trees [22], and the first constant factor $(2, 2B_v+3)$-approximation algorithm for arborescence, Steiner forests and Steiner networks [16].

In independent work, Bansal, Khandekar and Nagarajan [2] obtained an $(\frac{1}{\epsilon}, \frac{B_v}{1-\epsilon} + 4)$-approximation algorithm for the MINIMUM BOUNDED DEGREE ARBORESCENCE problem for $0 < \epsilon \leq \frac{1}{2}$. Moreover, they obtain the first additive approximation algorithm for the bounded-degree arborescence problem with degree violation at most 2. In order to obtain additive guarantees on the degree bounds, however, the cost of the arborescence becomes unbounded. They show that this cost-degree tradeoff in their result is actually best possible using the natural linear programming relaxation [2], which is an exact formulation when degree constraints are not present. In contrast, our results show that it is possible to achieve additive approximation on the degree bounds for the min-

[1] An $(\alpha, f(B_v))$-approximation algorithm for the MINIMUM BOUNDED-DEGREE STEINER NETWORK problem is a polynomial time algorithm which returns a solution of cost at most $\alpha \cdot$ OPT and $deg(v) \leq f(B_v)$ for all v, where OPT is the optimal cost of a Steiner Network with $deg(v) \leq B_v$ for all v.



(a) **(b)** **(c)**

Figure 1: The original graph is shown in (a). There are degree constraints on x and y, both are $\lceil k/2 \rceil$. The connectivity requirements are 1 for all pair of vertices. The fractional solution with all edges having value $1/2$ is a basic solution. The algorithm insolution in (b) where the degrees bounds are violated by a factor of 2, although there is an integer solution as shown in (c) where the degrees bounds are violated by at most 1.

imum bounded-degree Steiner network problem, while matching the best known approximation on the cost. Finally, we remark that both results develop on the iterative relaxation method in [10, 16, 22], which provides a unifying framework to obtain (nearly) tight analysis for linear programming relaxations of network design problems.

2. MINIMUM BOUNDED DEGREE STEINER FOREST

2.1 Background

Our algorithm develops on the previous work in [10, 16, 22]; we first review some necessary background. We begin by formulating a linear program for the problem. Set $f(S) = max_{u \in S, v \notin S}\ r_{uv}$ for each subset $S \subseteq V$. For the BOUNDED DEGREE STEINER FOREST problem $f(S)$ is $\{0,1\}$-valued since $r_{uv} \in \{0,1\}$ for all $u, v \in V$. It is known that f is a *weakly supermodular* function [10], that is, for every two subsets X and Y, either

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y)$$

or
$$f(X) + f(Y) \leq f(X - Y) + f(Y - X).$$

For a subset $U \subseteq E$, we denote $x(U) := \sum_{e \in U} x_e$; for a subset $S \subseteq V$, $\delta(S)$ denotes the set of edges with exactly one end-point in S, and $d(S) := |\delta(S)|$. The following is a linear programming formulation for the MINIMUM BOUNDED DEGREE STEINER FOREST problem, in which the degree constraints are on a subset of vertices $W \subseteq V$.

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

For a subset $S \subseteq V$, the corresponding constraint $x(\delta(S)) \geq f(S)$ defines a vector in $\mathbb{R}^{|E|}$: the vector has an 1 corresponding to each edge $e \in \delta(S)$, and a 0 otherwise. We call this vector the *characteristic vector* of $\delta(S)$, and denote it by $\chi_{\delta(S)}$. Let $\mathcal{F} = \{S \mid x(\delta(S)) = f(S)\}$ be the set of tight constraints from the connectivity requirement constraints. Two sets X, Y are *intersecting* if $X \cap Y$, $X - Y$ and $Y - X$ are nonempty. A family of sets is *laminar* if no two sets are intersecting. Since

$$x(\delta(X)) + x(\delta(Y)) \geq x(\delta(X \cap Y)) + x(\delta(X \cup Y)) \text{ and}$$

$$x(\delta(X)) + x(\delta(Y)) \geq x(\delta(X - Y)) + x(\delta(Y - X))$$

for any two subsets X and Y and f is weakly supermodular, it follows from standard uncrossing arguments (see e.g. [10, 16]) that a basic solution of the above linear program is characterized by a laminar family of tight constraints. The following Lemma 2.1 is proved in [16].

LEMMA 2.1. *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 and a set $T \subseteq W$ such that:*

1. *x is the unique solution to: $\{x(\delta(v)) = B_v, \forall v \in T\} \bigcup \{x(\delta(S)) = f(S), \forall S \in \mathcal{L}\}$.*
2. *The vectors $\chi_{\delta(S)}$ for $S \in \mathcal{L}$ and $\chi_{\delta(v)}$ for $v \in T$ are linearly independent.*
3. *$|E| = |\mathcal{L}| + |T|$.*
4. *For any set $S \in \mathcal{L}$, $\chi_{\delta(S)} \neq \chi_{\delta(v)}$ for any $v \in W$.*

The laminar family $\mathcal{L}$ obtained in Lemma 2.1 defines a directed forest L in which nodes correspond to sets in $\mathcal{L}$ and there exists an edge from set R to set S if R is the smallest set containing S. We call R the *parent* of S and S the *child* of R. A parent-less node is called a *root* and a childless node is called a *leaf.* Given a node R, the *subtree rooted at* R consists of R and all its descendants. We say a vertex v is *owned* by a set S if $v \in S$ and S is the smallest set in $\mathcal{L}$ containing v.

2.2 Iterative Algorithm

In this extended abstract, we present an algorithm proving the relaxed version of Theorem 1.1 which returns a Steiner Forest of cost at most 2OPT with $deg_F(v) \leq B_v + 4$ for each $v \in V$. The proof of Theorem 1.1 goes along similar lines but has more case analysis.

Our algorithm is an iterative relaxation algorithm as shown in Figure 2. The main difference from the previous iterative rounding algorithms is in Step 2d, where a heavy edge is picked only if both endpoints do not have degree constraints. This is the key step to avoid a multiplicative factor of 2 on the degree bounds. Also, by only picking edges with no degree constraints, there is no need to update the degree bounds fractionally as in [16]. Note also that the relaxation step has been generalized to remove a degree constraint when a vertex has degree at most $B_v + 4$.

Minimum Bounded Degree Steiner Forest

1. Initialization $F \leftarrow \emptyset$, $f'(S) \leftarrow f(S)\ \forall S \subseteq V$.
2. While F is not a Steiner forest do
   (a) *Computing a basic optimal solution:*
   Find a basic optimal solution x satisfying f' and remove every edge e with $x_e = 0$.
   (b) *Removing a degree constraint:*
   For every $v \in W$ with degree at most $B_v + 4$, remove v from W.
   (c) *Picking an 1-edge:*
   For each edge $e = \{u, v\}$ with $x_e = 1$, add e to F, remove e from G, and decrease B_u, B_v by 1.
   (d) *Picking a heavy edge with no degree constraints:*
   For each edge $e = \{u, v\}$ with $x_e \geq \frac{1}{2}$ and $u, v \notin W$, add e to F and remove e from G.
   (e) *Updating the connectivity requirements:*
   Set $f'(S) \leftarrow f(S) - \delta_F(S)$.
3. Return F.

Figure 2: An iterative algorithm for the MINIMUM BOUNDED DEGREE STEINER FOREST **problem.**

The following lemma is at the heart of the algorithm, which shows that the algorithm will always terminate successfully.

LEMMA 2.2. *Every basic solution x of (LP) must satisfy one of the following:*

1. *There is an edge e with $x_e = 0$ or $x_e = 1$.*
2. *There is an edge $e = \{u, v\}$ with $x_e \geq \frac{1}{2}$ and $u, v \notin W$.*
3. *There is a vertex $v \in W$ with $deg(v) \leq B_v + 4$.*

Note that the updated connectivity requirement function f' is also a weakly supermodular function. With Lemma 2.2, using a simple inductive argument as in [16], it can be shown that the algorithm returns a Steiner forest of cost at most twice the optimal cost and the degree of each vertex is at most $B_v + 4$. The rest of this section is devoted to the proof of Lemma 2.2.

2.3 A New Counting Argument

The proof of Lemma 2.2 is by contradiction. Let $\mathcal{L}$ be the laminar family and $T \subseteq W$ be the set of tight vertices defining the basic optimal solution x as in Lemma 2.1. The contradiction is obtained by a counting argument. Each edge in E is assigned two tokens. Then the tokens will be redistributed such that each member of $\mathcal{L}$ and each vertex in T get at least two tokens, and there are still some extra tokens left. This will give us a contradiction to Lemma 2.1 that $|E| = |\mathcal{L}| + |T|$.

We say an edge is *heavy* if $x_e \geq \frac{1}{2}$. If all conditions of Lemma 2.2 do not hold, we must have that there is no 0-edge and no 1-edge,

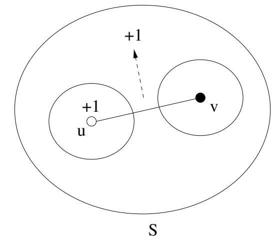


Figure 3: Rule (2) of the token assignment scheme. This is a new rule which is useful in collecting extra token for S. Here, the degree constraint for vertex u has been removed but the degree constraint for vertex v is present.

every heavy edge has an endpoint in W, and each vertex $v \in W$ has at least $B_v + 5$ edges incident at it.

Token assignment scheme: The two tokens for an edge $e = \{u, v\}$ are assigned by the following rules.

1. One token of e is assigned to u and the other token of e is assigned to v.
2. If $e = (u, v)$ is a heavy edge with $v \in W$ and u is not contained in the smallest set in $\mathcal{L}$ containing v, then the token of e for v is reassigned to the smallest set $S \in \mathcal{L}$ containing both u and v.

Classes: Let R be a set in $\mathcal{L}$. An edge $e = \{u, v\}$ is an *out-heavy edge* of R if $u \in R \backslash W$ and $v \in W \backslash R$ and $x_e \geq \frac{1}{2}$. The following definition is important to the analysis. For a set $R \in \mathcal{L}$, we call R is of:

- *Class Ia*: if $|\delta(R)| = 2$ and R has one out-heavy edge e with $x_e > \frac{1}{2}$.
- *Class Ib*: if R has two out-heavy edges.
- *Class IIa*: if $|\delta(R)| = 3$ and $x_e < \frac{1}{2}$ for each edge $e \in \delta(R)$.
- *Class IIb*: if R has one out-heavy edge.
- *Class III*: otherwise.

The following lemma shows that the tokens can be redistributed so that each member of $\mathcal{L}$ and each vertex in W gets at least two tokens. The proof is by induction on the laminar family.

LEMMA 2.3. *For any subtree of the laminar family $\mathcal{L}$ rooted at S, we can redistribute tokens in S such that*

1. *Every vertex in $T \cap S$ gets at least two tokens.*
2. *Class I sets in the subtree get at least two tokens.*
3. *Class II sets in the subtree get at least three tokens.*
4. *Class III sets in the subtree get at least four tokens.*

PROOF. Here is a brief outline of the proof. First we show in Claim 2.4 that a set owning at least two vertices in W can collect enough tokens; this uses the fact that f is a 0-1 function. Then Claim 2.5 and Claim 2.6 are used to show that a set owning exactly one vertex in W can collect enough tokens. Then the remaining cases consider sets which do not own any vertex in W, which rely crucially on Claim 2.7. We remark that Rule (2) of the token assignment scheme and the asymmetry in the definition of out-heavy edges are used in Claim 2.7. Now we start the proof by proving Claim 2.4.

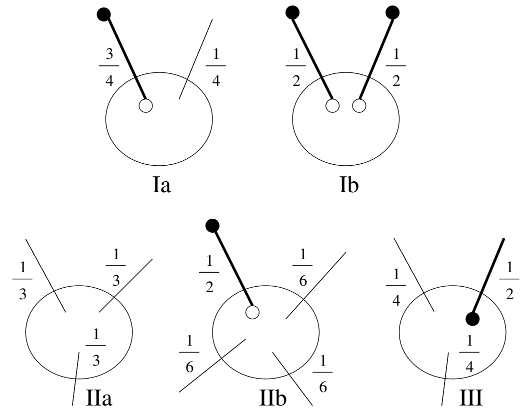


Figure 4: The figure shows examples of sets of each class. A vertex without degree constraint is white, otherwise it is black. (An endpoint without a vertex shown means that this information is not important.) A heavy edge is represented by a thick line. Note the definition of Class Ia, Class Ib and Class II require out-heavy edges. The rightmost example is a Class III set, although it has a heavy edge.

CLAIM 2.4. *For each S in $\mathcal{L}$, we can assume that S owns at most one vertex in W.*

PROOF. Let S owns $w_1, \ldots, w_l \in W$. Since $B_v \geq 1$ for all $v \in W$, by Step 2b of the algorithm, each vertex w_i is of degree at least 6. Since $f(S) = 1$, $\delta(S)$ can have at most two heavy edges. Hence, by the token assignment scheme, there are at least $6r-2$ tokens assigned to $w_1, \ldots, w_l$ which have not be reassigned by Rule (2). Since each vertex in $W \cap S$ needs only two tokens, there are still $4r-2$ extra tokens left. If $r \geq 2$, then S can collect at least 4 tokens, as required. $\square$

Hence suppose w is the only vertex in W owned by S.

CLAIM 2.5. *Let S be the set that owns $w \in W$. Then w is assigned at least five tokens, and is assigned exactly five tokens only if:*

1. *$deg(w) = 6$, $B_w = 1$, and there is one heavy edge of $\delta(S)$ in $\delta(w)$.*

2. *$deg(w) = 7$, $B_w = 2$, and there are two heavy edges of $\delta(S)$ in $\delta(w)$. In this case, $\delta(S) = \delta(S) \cap \delta(w)$.*

PROOF. Since $f(S) = 1$, $\delta(S)$ can have at most two heavy edges. So w receives one token for each edge incident at w except for the heavy edges in $\delta(w) \cap \delta(S)$. By Step 2b of the algorithm, w is of degree at least six. Hence, if there is no heavy edge in $\delta(S) \cap \delta(w)$, then w receives six tokens. Suppose that $\delta(S) \cap \delta(w)$ has only one heavy edge. Thus w receives $deg(w) - 1 \geq 5$ tokens and exactly five tokens only if $deg(w) = 6$ and $B_w = 1$.

Suppose $\delta(w) \cap \delta(S)$ has two heavy edges, then $B_w = x(\delta(w)) \geq 2$ as there are no 0-edges. So $deg(w) \geq 7$ by Step 2b of the algorithm. Thus, w receives at least $deg(w) - 2 \geq 5$ tokens and exactly five tokens only if $deg(w) = 7$, $B_w = 2$ and $\delta(S) = \delta(w) \cap \delta(S)$ contains two heavy edges. $\square$

We now show that when S owns exactly one vertex in W, there are enough tokens for S and the vertices in W it owns.

CLAIM 2.6. *If S owns one vertex in W, then there are enough tokens for w and S.*

PROOF. Let w be the vertex in W that S owns. By Claim 2.5 w receives at least five tokens. Note that w needs only two tokens if $w \in T$, and S only needs $4 - h$ tokens where h is number of out-heavy edges in $\delta(S)$. Thus, if $w \notin T$ or w has at least six tokens, then there are enough tokens. Also, if $\delta(S)$ has an out-heavy edge or S owns an endpoint, then there are enough tokens. Furthermore, if S has a Class II or Class III child, then S can take one excess token from it. Now we show that the remaining cases cannot happen.

In the remaining cases, all children of S, if any, must be of Class I. If there is no child of S, since S does not own an endpoint, this implies that $\chi_{\delta(S)} = \chi_{\delta(w)}$ contradicting Lemma 2.1 (2). So assume S has a Class I child, say R. Since $\delta(S)$ does not contain any out-heavy edges and by definition there is no heavy edge with endpoints in distinct Class I children of S, this implies that every out-heavy edge in $\delta(R)$ must have w as an endpoint. By Claim 2.5, w has exactly five tokens only if $B_w \leq 2$. There are two cases:

1. $B_w = 1$. In this case, we have an heavy edge in $\delta(w) \cap \delta(S)$ and also $deg(w) = 6$ by Claim 2.5. Also, there is another heavy edge incident at $\delta(w)$ from $\delta(R)$. But since there are no 0-edges, $B_w = x(\delta(w)) = 1$ implies that $deg(w) = 2$, a contradiction.

2. $B_w = 2$. In this case, we have two heavy edges in $\delta(S) = \delta(S) \cap \delta(w)$ by Claim 2.5. Since S does not own an endpoint, this implies that $\delta(R) \subset \delta(w)$. But then $x(\delta(R)) = 1$, $|\delta(R)| = 2$, and so $B_w = x(\delta(w)) = 2$ implies $deg(w) = |\delta(S)| + |\delta(R)| = 4$, a contradiction.

This completes the proof of Claim 2.6. $\square$

We now show that the induction hypothesis holds when S does not own a vertex of W.

Base Case of Lemma 2.3: $S \in \mathcal{L}$ is a leaf node in the laminar family. S gets one token for each edge in $\delta(S)$ since S does not own a vertex in W. Therefore, it gets two tokens only if S is of Class I, three tokens only if it is of Class II, and at least four tokens in any other case, as required.

Induction step of Lemma 2.3: The proof is by induction on number of children of S. Let h be the number of out-heavy edges in S, and let t be the number of tokens that S can collect. In the following we say a child R is of *Type A* if R is of Class Ia or of Class IIa. Note that we need $h + t \geq 4$ if S is not of Type A, and $h + t \geq 3$ if S is of Type A. The following Claim 2.7 is crucial and needs the definition of out-heavy edges and Rule (2) of the token assignment scheme.

CLAIM 2.7. *Each Class Ib, Class IIb, or Class III child R of S can contribute at least 2 to $h + t$. And each Class Ia, Class IIa child can contribute at least 1 to $h + t$.*

PROOF. If R is of Class III, then it has 2 excess tokens. If R is of Class IIb, then it has 1 excess token and one out-heavy edge $e \in \delta(R)$. If $e \in \delta(S)$, then it contributes 1 to h. Otherwise if both endpoints of e are in S, then it contributes 1 to t by Rule (2) of the token assignment scheme. Note that, by definition, an edge can be an out-heavy edge of at most one child of S, and so its contribution to t will not be double counted. If R is of Class Ib, then it has 2 out-heavy edges. By the same argument, these edges contributes 2 to $h + t$. Similarly, if R is of Class Ia, then it has 1 out-heavy edge, and thus contributes 1 to $h + t$. Finally, if R is of Class IIa, then it has 1 excess token. $\square$

We now prove the following claim which helps us prove the various cases of the induction.

CLAIM 2.8. *Suppose S is a set which does not own any vertices in W, has $\alpha \geq 1$ children all of which are Type A, own β endpoints and has no out-heavy edges in $\delta(S)$ for which one endpoint is owned by S. If $\alpha + \beta = 3$ then S is of Type A.*

PROOF. We prove the claim by a case analysis on different values of α.

1. $\alpha = 1$. Thus $\beta = 2$. Let R be the child of S. Since $\chi_{\delta(R)}$ and $\chi_{\delta(S)}$ are independent, there must exist edges $e \in \delta(R) \setminus \delta(S)$ and $f \in \delta(S) \setminus \delta(R)$ and S receives one token for both these edges. Moreover there is no other edge in $\delta(S) \setminus \delta(R)$ or $\delta(R) \setminus \delta(S)$ since $\beta = 2$. Now, if R is of Class Ia then the out-heavy edge in $\delta(R)$ must also be in $\delta(S)$ since S does not own a vertex in W. In this case S is also of Class Ia. If R is of Class IIa then $x_e = x_f < \frac{1}{2}$ and $\delta(S) = \delta(R) \cup \{f\} \setminus \{e\}$ and S is also of Class IIa.

2. $\alpha = 2$. Thus $\beta = 1$. Let R_1 and R_2 be the children of S. R_1 and R_2 cannot both be of Class Ia since the out-heavy edge in $\delta(R_i)$ must be in $\delta(S)$ for $i = 1, 2$, but then $x(\delta(S)) > 1$, a contradiction. First suppose R_1 is of Class Ia and therefore R_2 is of Class IIa. We cannot have $|\delta(R_1, R_2)| \geq 2$ since $|\delta(R_1)| = 2$, and also cannot have $|\delta(R_2) \cap \delta(S)| \geq 2$ since $f(S) = x(\delta(S)) = 1$. So the only possibility is $|\delta(R_1, R_2)| = |\delta(v, R_2)| = |\delta(R_2) \cap \delta(S)| = 1$. Hence $|\delta(S)| = 2$ and thus S is of Class Ia, as required. Finally, suppose both R_1 and R_2 are of Class IIa and $\delta(S)$ does not contain any heavy edges, then by a similar argument as in Jain [10] (see Claim 3.5), S is of Class IIa, as required.

3. $\alpha = 3$. Let R_1, R_2 and R_3 be the children of S. As previously argued in the case of $\alpha = 2$, at most one of R_1, R_2, R_3 can be of Class Ia. First suppose that S has exactly one Class Ia child, say R_1. Let $\delta(R_1) = \{e_1, f_1\}$, where e_1 is the out-heavy edge of R_1. Assume, without loss of generality, that $f_1 \in \delta(R_2)$. Since $f(S) = 1$, we must have $|\delta(R_2, R_3)| = 2$; otherwise $|\delta(R_3) \cap \delta(S)| \geq 2$ and thus $x(\delta(S)) = x_{e_1} + x(\delta(R_3) \cap \delta(S)) > \frac{1}{2} + \frac{1}{2} = 1$, since $|\delta(R_3)| = 3$ and each edge e in $\delta(R_3)$ has $x_e < \frac{1}{2}$ by the definition of a Class IIa child. Since $|\delta(R_2, R_3)| = 2$, this implies that $d(S) = 2$, and hence S is of Class Ia and therefore Type A. In the other case, we have that all three children of S are of Class IIa. Then, using arguments similar to Jain [10] (see Claim 3.5), it follows that S must also be of Class IIa.

Thus the claim follows. $\square$

Now we complete the inductive argument based on the number of children of S.

1. S has at least four children. Then each child can contribute at least 1 to $h + t$, and so $h + t \geq 4$.

2. S has exactly three children. If there is a child which is not of Type A, then $h + t \geq 4$ by Claim 2.7, as required. So assume S has exactly three Type A children R_1, R_2, R_3. If S owns an endpoint then also $h + t \geq 4$. So further assume that S does not own an endpoint. Then S satisfies the conditions of Claim 2.8 and must be of Type A. Thus $h + t \geq 3$ suffices for S.

3. S has exactly two children R_1 and R_2. If both R_1 and R_2 are not of Type A, since each can contribute 2 to $h + t$ by Claim 2.7, then we are done.

   Suppose R_1 is of Type A and R_2 is not of Type A. If S owns an endpoint then we are done. So further assume that S does not own an endpoint. We shall prove that this would not happen. In this case

$$x(\delta(R_1) \cap \delta(S)) + x(\delta(R_1, R_2)) = x(\delta(R_1)) = 1,$$
$$x(\delta(R_1) \cap \delta(S)) + x(\delta(R_2) \cap \delta(S)) = x(\delta(S)) = 1,$$
$$x(\delta(R_2) \cap \delta(S)) + x(\delta(R_1, R_2)) = x(\delta(R_2)) = 1,$$

   Thus we have,

$$x(\delta(R_1) \cap \delta(S)) = x(\delta(R_1, R_2)) = x(\delta(R_2) \cap \delta(S)) = \frac{1}{2}.$$

   R_1 cannot be of Class Ia, since otherwise it has an edge with $x_e > \frac{1}{2}$. Also, R_1 cannot be of Class IIa, since $|\delta(R_1)| = 3$, either $\delta(R_1, R_2)$ or $\delta(R_2) \cap \delta(S)$ is a single edge e with $x_e = \frac{1}{2}$, contradicting R_2 is of Class IIa.

   So suppose R_1 and R_2 are of Type A. If S owns two endpoints, then we are done. By the above argument, S must owns at least one endpoint, and thus $h + t \geq 3$. If S has an out-heavy edge for which S owns one endpoint then we have $h + t \geq 4$ and we are done. Hence assume that S owns exactly one endpoint v and each out-heavy edge in $\delta(S)$ is in $\delta(R_i)$ for some i. Thus S satisfies the condition of Claim 2.8 and is of Type A. Thus $t \geq 3$ suffices for S.

4. S has exactly one child R. By linear independence of $\chi_{\delta(S)}$ and $\chi_{\delta(R)}$, S must own at least two endpoints, and thus $h + t \geq 3$. If R is not of Type A, then $h + t \geq 4$, and we are done. If $\delta(S) \setminus \delta(R)$ has an out-heavy edge or S owns more than two endpoints then also we have $h + t \geq 4$ as required. In the remaining case S satisfies conditions of Claim 2.8 and S is of Type A. Therefore, $h + t \geq 3$ suffices.

This completes the proof of Lemma 2.3. If some root S of the laminar family is not of Class I, then there is some excess token left at S by Lemma 2.3. If every root is of Class I, then there must exist a vertex $w \in W$ that is not contained in any root, and so there is some excess token left at w. This completes the proof of the relaxed version of Theorem 1.1.

Remarks: To prove Theorem 1.1, the same induction hypothesis and token assignment scheme are used, but a more involved counting argument is needed to show that a set owning exactly one vertex in W can collect enough tokens. The details will appear in the journal version of this paper.

3. MINIMUM BOUNDED DEGREE STEINER NETWORK

In this section we prove Theorem 1.2. The linear programming relaxation is exactly the same as in the previous section, except that the function f is not necessarily a $\{0, 1\}$-valued function.

3.1 Algorithm

The iterative algorithm is given in Figure 5. The algorithm is similar to the algorithm for the MINIMUM BOUNDED DEGREE STEINER FOREST problem, with the following main difference. In Step 2a we define a set of *high degree vertices* $W_h = \{v \in W \mid \sum_{e \in \delta(v)} x_e \geq 6 f_{max}\}$, where $f_{max} := \max_S f(S)$. This set plays the same role as the set of vertices with degree constraints in the Steiner forest algorithm. Then in Step 2d we only pick a heavy edge when both of its endpoints are not high degree vertices. This is the key step to ensure that the degree bounds are only violated by an additive term.

Minimum Bounded Degree Steiner Network

1. Initialization $F \leftarrow \emptyset$, $f'(S) \leftarrow f(S)\ \forall S \subseteq V$.
2. While F is not a Steiner network do
   (a) *Computing a basic optimal solution:*
   Find a basic optimal solution x satisfying f' and remove every edge e with $x_e = 0$.
   Set $W_h = \{v \in W | \sum_{e\in\delta(v)} x_e \geq 6f_{max}\}$ and $B_v = \sum_{e\in\delta(v)} x_e$ for $v \in W$.
   (b) *Removing a degree constraint:*
   For every $v \in W$ with degree at most 4, remove v from W.
   (c) *Picking an 1-edge:*
   For each edge $e = (u, v)$ with $x_e = 1$, add e to F, remove e from G, and decrease B_u, B_v by 1.
   (d) *Picking a heavy edge with both endpoints low:*
   For each edge $e = (u, v)$ with $x_e \geq 1/2$ and $u, v \notin W_h$, add e to F, remove e from G, and decrease B_u and B_v by $1/2$.
   (e) *Updating the connectivity requirement function:*
   For every $S \subseteq V$: $f'(S) \leftarrow f(S) - |\delta_F(S)|$.
3. Return F.

Figure 5: An iterative algorithm for the MINIMUM BOUNDED DEGREE STEINER NETWORK **problem.**

First we show that the algorithm returns the solution with the claimed guarantees for cost and degree in Theorem 1.2 assuming that the algorithm always proceed in one of the iterations. Then we show in Lemma 3.2 that for any basic feasible solution to the linear program one of the conditions must be satisfied.

LEMMA 3.1. *If in each iteration one of the conditions in Step 2b, Step 2c or Step 2d is satisfied then the algorithm returns a Steiner network with cost at most twice the optimal linear programming solution and degree bound of each vertex is violated by at most* $6r_{max} + 3$.

PROOF. The proof is by a standard inductive argument. We give a short explanation. Note that f' is a weakly supermodular function. Since we always pick an edge with $x_e \geq \frac{1}{2}$ and the remaining fractional solution is a feasible solution for the residual problem, the cost of the solution returned is at most 2 times the cost of the linear programming solution as claimed in Theorem 1.2.

For the guarantee on the degree bound, firstly observe that for any vertex v, we pick at most $B_v - 6f_{max}$ edges in Step 2c incident at v since the degree bound of v is reduced by one whenever such an edge is picked. In Step 2d, we pick at most $12f_{max} - 1$ edges incident at v since the degree bound is reduced by $\frac{1}{2}$ whenever we include such an edge. Moreover, at most 4 edges can be picked incident at v once the degree constraint for v is removed. Hence, the number of edges picked which are incident at v is at most

$$B_v - 6f_{max} + 12f_{max} - 1 + 4 = B_v + 6f_{max} + 3,$$

as required. □

For the correctness of the algorithm, we shall prove the following key lemma in Section 3.2 which will ensure that the algorithm terminates with a feasible solution and complete the proof of Theorem 1.2. The rest of this section is devoted to the proof of Lemma 3.2.

LEMMA 3.2. *Let x be a basic feasible solution of (LP), and W be the set of vertices with degree constraints, and $W_h = \{v \in W \mid \sum_{e\in\delta(v)} x_e \geq 6f_{max}\}$. Then at least one of the following holds.*

1. *There exists an edge e with $x_e = 1$.*
2. *There exists an edge $e = \{u, v\}$ with $x_e \geq 1/2$ and $u, v \notin W_h$.*
3. *There exists a vertex $v \in W$ such that $deg_E(v) \leq 4$.*

3.2 A Counting Argument

We shall prove Lemma 3.2 by a counting argument. Suppose, by way of contradiction, that none of the conditions in the lemma holds. Then each edge e has $0 < x_e < 1$, and each edge e with $1 > x_e \geq 1/2$ (we call such an edge a *heavy edge*) must have at least one endpoint in W_h, and each vertex in W must have degree at least five. We shall also give two tokens for each edge (the token assignment scheme is explained below) for a total of $2|E|$ tokens. Then, the tokens will be reassigned so that each member of $\mathcal{L}$ gets at least two tokens, each vertex in T gets at least two tokens and we still have some excess token left. This will contradict $|E| = |\mathcal{L}| + |T|$ of Lemma 2.1, and thus completes the proof.

The main difference from Jain's analysis is the existence of heavy edges (with an endpoint in W_h) which our algorithm is not allowed to pick. In the following, we say a vertex in W_h is a *high* vertex. Since there are heavy edges, a set $S \in \mathcal{L}$ may only have two edges in $\delta(S)$, and hence S may not be able to collect three tokens. To overcome this, we use a different token assignment scheme so that a similar induction hypothesis as Jain's would work.

Token assignment scheme: If $e = \{u, v\}$ is a heavy edge, $u \in W_h$ and $v \notin W$, then v gets two tokens from e and u gets zero token. For every other edge e, one token is assigned to each endpoint of e.

Co-requirement: We also need to refine the definition of co-requirement for the presence of heavy edges:

$$coreq(S) = \sum_{e\in\delta(S),\ x_e<1/2} (1/2 - x_e) + \sum_{e\in\delta(S),\ x_e\geq 1/2} (1 - x_e).$$

It is useful to note that this definition reduces to Jain's definition if every edge e with $x_e \geq \frac{1}{2}$ is thought of as two parallel edges aiming to each achieves a value of $\frac{1}{2}$ and sharing the current x_e value equally (i.e. each gets $\frac{x_e}{2}$): summing $\frac{1}{2} - \frac{x_e}{2}$ over the two parallel edges gives $1 - x_e$.

After this initial assignment, each vertex in $V \setminus W_h$ receives at least as many tokens as their degree. Moreover, each vertex in $W \setminus W_h$ receive at least five tokens (as their degree is at least five). Note that a vertex $v \in W_h$ might not have any tokens if all the edges incident at it are heavy edges. By exploiting the fact that $f(S) \leq f_{max}$, however, we shall show that vertices in W_h can *get back* enough tokens during the inductive counting argument. Now we prove the following lemma which shows that the tokens can be reassigned as discussed previously.

LEMMA 3.3. *For any subtree of $\mathcal{L}$ rooted at S, we can reassign tokens such that each vertex in $T \cap S$ gets at least two tokens, each set in the subtree gets at least two tokens, and the root S gets at least three tokens. Moreover, root S gets exactly three tokens only if $coreq(S) = \frac{1}{2}$.*

PROOF. We now proceed by induction on the height of the subtree to prove Lemma 3.3. We first prove the base case of the induction hypothesis where we also show a crucial Claim 3.4, which handle all sets that own some vertices in W. We then use this claim in the main induction proof to complete the proof of Lemma 3.3.

Base Case of Lemma 3.3: S is a leaf node. First suppose that $S \cap W_h = \emptyset$. If there exists $v \in S \cap (W \setminus W_h)$, then v has at least five tokens. Since v only needs two tokens, it has three excess tokens which it can give to S. If there are two such vertices or S owns another endpoint, then S gets at least four tokens as required. Otherwise, we have $\chi_{\delta(v)} = \chi_{\delta(S)}$ which is a contradiction to the linear independence of characteristic vectors in Lemma 2.1. Hence, we assume $S \cap W = \emptyset$. Then S can get at least $\delta(S)$ tokens from the vertices owned by S. Note that $|\delta(S)| \geq 2$, as $x(\delta(S))$ is an integer and there is no 1-edge. If $|\delta(S)| \geq 4$, then S gets four tokens. If $|\delta(S)| = 3$ and $|\delta(S)|$ contains a heavy edge, then S can get four tokens from the vertices it owns, since an endpoint $v \notin W$ of a heavy edge has 2 tokens by the token assignment scheme. If it does not contain a heavy edge, then S receives three tokens and $coreq(S) = \frac{1}{2}$. If $|\delta(S)| = 2$, then at least one edge is a heavy edge. If both edges are heavy then S can get four tokens, else if only one edge is heavy then it gets three tokens and $coreq(S) = \frac{1}{2}$.

We now consider the case that S owns a vertex in W_h, and show that S can collect enough tokens for the inductive argument. The following claim is the key to deal with degree constraints, which uses crucially the parameter f_{max}. This claim holds even when S is not a leaf in the laminar family, and will also be used in the induction step.

CLAIM 3.4. *Suppose S owns $r \geq 1$ vertices in W_h. Then the number of excess tokens from the children of S, plus the number of tokens owned by S, plus the number of tokens left with vertices in W_h owned by S is at least $2r + 4$.*

PROOF. Let S have c children. As each child has at least one excess token by the induction hypothesis, if $c \geq 6r$ then we have $6r$ tokens which is at least $2r + 4$. Hence, we assume that $c < 6r$.

Let $B := \sum_v B_v \geq \sum_v 6f_{max} = 6rf_{max}$, where the sum is over all vertices $v \in W_h$ owned by S. Intuitively, vertices in W_h owned by S would have collected a total of B tokens if the two tokens at each edge is distributed evenly. But by the token assignment scheme, vertices in W_h owned by S may not get any token for heavy edges incident on them. We are going to show that these vertices can still "get back" the two tokens they need for the inductive argument.

For a child R of S, as $x(\delta(R)) = f(R) \leq f_{max}$, at most f_{max} units of B come from the edges in $\delta(R)$. Similarly, at most f_{max} units of B come from the edges in $\delta(S)$. Hence, there are at least $f_{max}(6r - c - 1)$ units of B coming from the edges with both endpoints owned by S. Since there is no 1-edge, there are at least $f_{max}(6r - c - 1) + 1$ such endpoints from those edges. Let $e = \{u, v\}$ be such an edge with $v \in W_h$ owned by S. If $u \in W$, then both u and v get one token from e in the initial assignment. If $u \notin W$, then u gets two tokens from e in the initial assignment, but these two tokens are owned by S. So, the number of tokens owned by S plus the number of tokens left with vertices in W_h owned by S is at least $f_{max}(6r - c - 1) + 1$. Furthermore, S can also collect one excess token from each child. So, the total number of tokens S can collect is at least $f_{max}(6r - c - 1) + c + 1$, which is a decreasing function of c. As $c < 6r$, the number of tokens is minimized at $c = 6r - 1$, which is at least $6r \geq 2r + 4$. □

In the base case when S owns a vertex in W_h, by Claim 3.4 S can collect $2r + 4$ tokens. So these tokens can be redistributed so that S has 4 tokens and each vertex in W_h owned by S has 2 tokens, which is enough for the induction hypothesis.

Induction Step: The presence of heavy edges with $x_e \geq \frac{1}{2}$ introduces some difficulties in carrying out the inductive argument in [10]. We need to prove some lemmas which work with the new notion of co-requirement and the presence of heavy edges.

For any set S, let $wdeg(\delta(S))$

$$= |\{e \in \delta(S) : 0 < x_e < \frac{1}{2}\}| + 2|\{e \in \delta(S) : x_e \geq \frac{1}{2}\}|$$

be the *weighted degree* of S. This definition is keeping with the idea that each edge with $x_e \geq \frac{1}{2}$ is thought of as two parallel edges. Observe that for any $v \notin W$, it receives exactly $wdeg(v)$ tokens in the initial assignment as it gets one token for each edge and two tokens for all heavy edges incident at it. S can take all the tokens for all the vertices it owns which are not in W. We call these the *tokens owned by* S. Let $G' = (V, E')$ be the graph formed by replacing each heavy edge e by two edges e' and e'' such that $x_{e'} = x_{e''} = \frac{x_e}{2}$. Observe that

$$coreq(S) = \sum_{e \in \delta(S),\, x_e < 1/2} (1/2 - x_e) + \sum_{e \in \delta(S),\, x_e \geq 1/2} (1 - x_e)$$

$$= \sum_{e \in \delta(S) \cap E'} (1/2 - x_e),$$

and $wdeg(\delta(S)) = |\delta'(S)|$ where $\delta'(S) = \{e \in E' : e \in \delta(S)\}$. Observe that $coreq(S)$ is integral or *semi-integral* (half-integral but not integral) depending on whether $\delta'(S)$ is even or odd. We first prove the same technical lemma as in [10] with the new definitions of co-requirements and weighted degrees.

CLAIM 3.5. *Let S be a set in $\mathcal{L}$ which owns α tokens and has β children where $\alpha + \beta = 3$ and does not own any vertex of W. Furthermore, each child of S, if any, has a co-requirement of $\frac{1}{2}$. Then the co-requirement of S is $\frac{1}{2}$.*

PROOF. Since each child R of S has a co-requirement of half, this implies that $|\delta'(R)|$ is odd. Note that we assume S does not own any vertex of W. Using these facts and that $\alpha + \beta = 3$, the same argument as in Exercise 23.3 of [23] can be used to show that $|\delta'(S)|$ is odd. Hence, the co-requirement of S is semi-integral. Now, we show that $coreq(S) < \frac{3}{2}$ proving the claim. Clearly,

$$coreq(S) = \sum_{e \in \delta'(S)} (1/2 - x_e) \leq \sum_R coreq(R) + \sum_e (1/2 - x_e),$$

where the first sum is over all children R of S and second sum is over all edges for which S owns a token. Since $\alpha + \beta = 3$, there are a total of three terms in the sum. Since, any term in the first sum is $\frac{1}{2}$ and in the second sum is strictly less than $\frac{1}{2}$, if $\alpha > 0$, we then have $coreq(S) < \frac{3}{2}$ which proves the claim. So, assume $\alpha = 0$, i.e. S does not own any tokens. In this case, edges incident to children of S cannot all be incident at S since otherwise it will violate the linear independence of characteristic vectors in $\mathcal{L}$ in Lemma 2.1, and therefore we have $coreq(S) < \sum_R coreq(R) = \frac{3}{2}$ proving the claim. □

We are now ready to prove that the induction step holds, in which S has at least one child. If S owns a vertex in W_h then Claim 3.4 shows that the induction hypothesis holds. Henceforth, we assume that S does not own any vertices of W_h. Suppose S owns some vertices in $W \setminus W_h$. Each such vertex gets at least five tokens. It needs only two tokens and hence can give three excess tokens to S.

As S has at least one child R, R can give at least one excess token to S, and hence S gets at least four tokens as required.

For the rest of the cases, we assume that S does not own any vertex of W, and hence the remaining case analysis is very similar to that of Jain, with a different definition of co-requirement.

- S has at least four children. Then S can take one excess token from each child.

- S has exactly three children. If any child S has two excess tokens or if S owns a vertex then S can get four tokens. Else, each of the three children of S has a co-requirement of half and S owns no vertices. Then, by Lemma 3.5, we have that S has co-requirement of $\frac{1}{2}$ and it only needs three tokens.

- S has exactly two children R_1 and R_2. If both of them have two excess tokens then we are done. Else, let R_1 have exactly one token and hence it has co-requirement of $\frac{1}{2}$ by the induction hypothesis. We now claim that S owns an endpoint. For sake of contradiction suppose S does not own any endpoint. Then, if there are α edge between R_1 and R_2 in E' (where we replace each heavy edge by two parallel edges), we have

$$|\delta'(S)| = |\delta'(R_1)| + |\delta'(R_2)| - 2\alpha$$

  As R_1 has a co-requirement of half, we have $|\delta'(R_1)|$ is odd and hence $\delta'(S)$ and $\delta'(R_2)$ have different parity and hence different co-requirements. The co-requirements of S and R_2 can differ by at most the co-requirement of R_1 which is exactly half. Since, $\chi_{\delta'(S)} \neq \chi_{\delta'(R_1)} + \chi_{d'(R_2)}$, there must be an edge between R_1 and R_2 and therefore, $coreq(S) < coreq(R_2) + \frac{1}{2}$. Similarly, $\chi_{\delta'(R_2)} \neq \chi_{\delta'(S)} + \chi_{\delta'(R_1)}$ and therefore there is an edge in $\delta'(S) \cap \delta'(R_1)$ which implies that $coreq(R_2) < coreq(S) + \frac{1}{2}$. Thus, their co-requirements are equal which is a contradiction. Thus S owns at least one endpoint.

  If S owns at least two endpoints or R_2 has two excess tokens, then we have four tokens for S. Otherwise, by Lemma 3.5, we have that co-requirement of S is half and it needs only three tokens.

- S has exactly one child R. Since both sets S and R are tight we have that $x(\delta(S)) = f'(S)$ and $x(\delta(R)) = f'(R)$. Since $\chi_{\delta(S)}$ and $\chi_{\delta(R)}$ are linearly independent, subtracting the two equations we have that $x(\delta(S)\Delta\delta(R))$ (Δ denotes symmetric difference) is an positive integer. Also, there are no 1-edges present and so $|\delta(S)\Delta\delta(R)| \geq 2$, and each edge in the symmetric difference gives one token to S. Thus S owns at least two endpoints. If S owns three endpoints or R has two excess tokens then S can get four tokens. Otherwise, S has exactly two endpoints and exactly one child which has co-requirement of $\frac{1}{2}$. Then by Lemma 3.5, S has a co-requirement of $\frac{1}{2}$.

This completes the proof of Lemma 3.3, which assigns two tokens to each set in the laminar family $\mathcal{L}$ and each vertex in T which is contained in some set $S \in \mathcal{L}$. For vertices in T which are not contained in any set $S \in \mathcal{L}$ we also have enough tokens. Observe that each vertex $v \in W \setminus W_h$ receives at least five tokens. For vertices in W_h not contained in any set $S \in \mathcal{L}$, an argument identical to Claim 3.4 with $S = V$ will give at least two tokens to each vertex in W_h.

Thus we have that $2|E| > 2|\mathcal{L}|+2|T|$, which contradicts Lemma 2.1. Therefore, one of the conditions in Lemma 3.2 holds, and hence we have Theorem 1.2.

Integrality Gap Example. In Figure 6 we show that the linear program (LP) has an integrality gap of $B + \Omega(r_{max})$ and therefore Theorem 1.2 is nearly tight.

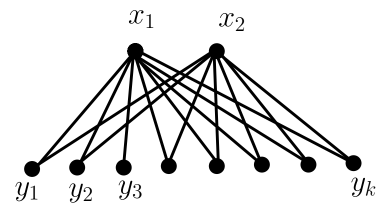


Figure 6: In this example, we have a complete bipartite graph $B = (X, Y, E)$ where $X = \{x_1, x_2\}$ and $Y = \{y_1, \ldots, y_k\}$. We set the connectivity requirements between y_i and y_j to be 1 for all i, j, between x_1 and x_2 to be $\frac{k}{2}$, and 0 otherwise. The fractional solution where all edges have fractional value $\frac{1}{2}$ is the optimal solution, in which the degree of x_1 and x_2 is equal to $\frac{k}{2} = \Delta_f^*$. On the other hand, it can be seen that in any integer solution, the degree of x_1 and x_2 must be at least $\frac{3}{4}k = \frac{3}{2}\Delta_f^*$. This example also shows that the integrality gap is at least $(2, B_v + \frac{r_{max}}{2})$.

Concluding Remarks

The iterative relaxation method has been successfully applied to network design problems with degree constraints [16, 22, 2], and recently it has also been applied to other combinatorial optimization problems [11]. In fact, this method can also be used to give simple proofs of classical results in combinatorial optimization and approximation algorithms [17]. We hope this method will find further applications.

4. REFERENCES

[1] A. Agrawal, P. Klein and R. Ravi, *When trees collide: an approximation algorithm for the generalized Steiner problem on networks*, in Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing (STOC), 134-144, 1991.

[2] N. Bansal, R. Khandekar and V. Nagarajan, *Additive guarantees for degree bounded directed network design*, in Proceedings of the Fourtieth Annual ACM Symposium on Theory of Computing (STOC), 2008.

[3] F. Bauer and A. Varma, *Degree-constrained multicasting in point-to-point networks*, in Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (Vol. 1), INFOCOM '95.

[4] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar, *What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs*, in Proceedings of Eighth International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 26-39, 2005.

[5] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. *A push relabel algorithm for approximating degree bounded MSTs*, in Proceedings of Thirty-third International Colloquium on Automata, Languages and Programming (ICALP), 853-865, 2006.

[6] N. Deo and S.L. Hakimi, *The shortest generalized Hamiltonian tree,* in Proceedings of Sixth Annual Allerton Conference, (1968), pages 879-888.

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

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

[9] M.X. Goemans, *Minimum bounded-degree spanning trees,* Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2006, pp. 273–282.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] L.C. Lau, S. Naor, M. Salavatipour and M. Singh, *Survivable network design with degree or order constraints*, in Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), 651-660, 2007.

[17] L.C. Lau, R. Ravi, M. Singh, *Iterative Relaxations*, in preparation.

[18] Carlos A. S. Oliveira and Panos M. Pardalos, *A survey of combinatorial optimization problems in multicast routing*, Computers and Operations Research, volume 32:8, 2005, pp: 1953–1981.

[19] B. Raghavachari, *Algorithms for finding low degree structures*, in Approximation Algorithms, Dorit Hochbaum (ed.), PWS Publishers Inc., pages 266-295, 1996.

[20] R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, Harry B. Hunt, III , *Approximation algorithms for degree-constrained minimum-cost network-design problems*, Algorithmica 2001.

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

[22] M. Singh and L.C. Lau, *Approximating minimum bounded degree spanning trees to within one of optimal*, In Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), 661-670, 2007.

[23] V. Vazirani, *Approximation Algorithms,* Springer, 2001.

[24] Stefan Voß, *Problems with generalized Steiner problems.* Algorithmica 7(2): 333-335 (1992).