Additive Approximation for Bounded Degree Survivable Network Design*

Lap Chi Lau† Mohit Singh‡

Abstract

In the minimum bounded degree Steiner network problem, we are given an undirected graph 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 b_v for each vertex v. The task is to find a minimum cost subgraph that satisfies all the connectivity requirements and degree upper bounds. Let $r_{\max} := \max_{u,v}\{r_{uv}\}$ and OPT be the cost of an optimal solution that satisfies all the degree bounds. We present approximation algorithms that minimize the total cost and the degree violation simultaneously.

- In the special case when $r_{\max} = 1$, there is a polynomial time algorithm that returns a Steiner forest of cost at most 2OPT and the degree of each vertex v is at most $b_v + 3$.
- In the general case, there is a polynomial time algorithm that returns a Steiner network of cost at most 2OPT and the degree of each vertex v is at most $b_v + 6r_{\max} + 3$.

The algorithms are based on the iterative relaxation method, and the analysis of the algorithms is nearly tight.

*A preliminary version appeared in Proceedings of the 40th Annual ACM Symposium on Theory of Computing, 759-768, 2008.

†The Chinese University of Hong Kong. Research supported by HK RGC grant 413609.

‡Microsoft Research, Redmond.

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 provide useful models for practical applications. In recent years, much effort has been put into designing approximation algorithms for network design problems with additional degree constraints. These problems generalize basic problems in combinatorial optimization, and have applications in various areas including VLSI design, vehicle routing and communication networks. In these applications, degree constraints occur as a natural modeling tool for *workload* of nodes. For example, in typical applications of network design problems to multicasting, the degree constraint on a switch corresponds to a bound on the multicast copies it can make in the network [3].

In this paper, we study the survivable network design problem with degree constraints. Given connectivity requirements r_{uv} for all pairs of vertices, a *Steiner network* is a subgraph 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 other network design problems as special cases; for example, a *Steiner forest* is a Steiner network with $r_{uv} \in \{0, 1\}$ for all pairs. 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. We design bicriteria approximation algorithms that optimize both objectives simultaneously. The main results are the following.

Theorem 1.1 *There is a polynomial time algorithm for the minimum bounded degree Steiner forest problem that returns a Steiner forest F of cost at most 2OPT and $\deg_F(v) \leq b_v + 3$ for all v, where OPT is the cost of an optimal solution that satisfies all the degree bounds.*

Theorem 1.2 *There is a polynomial time algorithm for the minimum bounded degree Steiner network problem that returns a Steiner network H of cost at most 2OPT and $\deg_H(v) \leq b_v + 6r_{\max} + 3$ for all v, where OPT is the cost of an optimal solution that satisfies all the degree bounds and $r_{\max} := \max_{u,v}\{r_{uv}\}$.*

Previously the best guarantees known on the degree for both Steiner forest and Steiner network are $2b_v + 2$ in [16], even when there are no costs on the edges. Theorems 1.1 and 1.2 provide the first *additive* approximation algorithms that violate the degrees by at most a constant for many problems, including Steiner trees and 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 guarantees known for the minimum cost Steiner forest and Steiner network problems [1, 9]. This provides a unifying algorithmic framework for a large class of network design problems.

The algorithms are based on the iterative relaxation method applied to a linear programming relaxation as in [9, 14, 21]. The analysis of the linear programming relaxation is nearly tight. There are examples in which the optimal fractional solution has maximum degree B, but any integral solution would have maximum degree at least $B + 2$ for Steiner forests in [14], and at least $B + \Omega(r_{\max})$ for Steiner networks as shown in Figure 13 in Section 3.3.

1.1 Techniques

The algorithms are based on the iterative relaxation method in [14, 21, 15], which adapts Jain's iterative rounding method [9] to the minimum bounded degree Steiner network problem. The approach in [14] relies on the following lemma about the extreme point solutions of the linear program for Steiner networks: "If every vertex with a degree constraint has degree at least five, then in any extreme point solution there is an

edge e with $x_e \geq 1/2$." This lemma leads to a new relaxation step to Jain's iterative rounding method to deal with degree bounds: If there is a vertex with degree at most four, then the degree constraint for that vertex is removed. This relaxation step only incurs an additive constant three on the degree bounds. After this step, the algorithm can always pick an edge with $x_e \geq 1/2$ as in Jain's approach, and hence the cost and the degrees are violated by at most a multiplicative factor of two. As illustrated in the example in Figure 1, the algorithm in [14] may actually violate the degree bounds by a multiplicative factor of two.

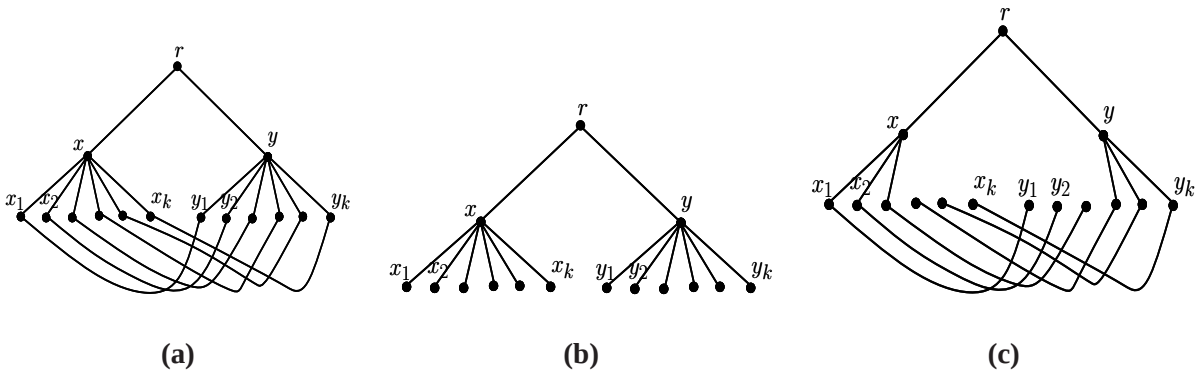

Figure 1: The original graph is shown in (a). There are degree constraints on x and y, both are $\lceil k/2 \rceil$. The connectivity requirements are one for all pair of vertices. The fractional solution with all edges having value $1/2$ is an extreme point solution. The algorithm in [14] may return the integral solution in (b) where the degree bounds are violated by a multiplicative factor of two, although there is an integral solution as shown in (c) where the degree bounds are violated by at most an additive constant one.

To achieve additive approximation on the degree bounds, we need to avoid paying a multiplicative factor of two on the degree bounds when picking edges with value $1/2$. We note that such edges are inevitable as a factor of two 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 an additive constant three. The main technical contribution is the following lemma about the extreme point solutions: "If every vertex v with degree constraint has degree at least $b_v + 4$, then in any extreme point solution there is an edge with $x_e \geq 1/2$ between two vertices *without* degree constraints." This leads us to a modified iterative algorithm that only selects edges with $x_e \geq 1/2$ between two vertices without degree constraints, which departs 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 three when they are removed. This approach can be extended to Steiner networks, by proving that in any extreme point solution there are edges with $x_e \geq 1/2$ between two "low degree vertices". The proofs of these characterizations of the extreme point solutions require new ideas on the counting argument, which crucially exploit the parameter $r_{\max}$.

1.2 Related Work

For the minimum cost Steiner network problem, Jain [9] introduced the iterative rounding method to obtain a 2-approximation algorithm, improving on a line of research that applied primal-dual methods to these problems. For bounded-degree spanning trees and Steiner trees, Fürer and Raghavachari [7] gave an approximation algorithm which violates the degrees by at most an additive constant one. This result has generated much interest in obtaining approximation algorithms for network design problems with degree constraints

[12, 13, 11, 14, 6, 18, 4, 5, 19, 20, 8, 21]. 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 [8]. Recently, the iterative relaxation method has been used to obtain the best bounds known for these problems: $(1, b_v + 1)$ for spanning trees [21], $(2, 2b_v + 2)$ for arborescence [14], and $(2, 2b_v + 2)$ for Steiner forests and Steiner networks [16].

In independent work, Bansal, Khandekar and Nagarajan [2] obtained improved approximation algorithms for degree bounded network design problems in *directed* graphs. They gave an $(\frac{1}{\epsilon}, \frac{b_v}{1-\epsilon} + 4)$ approximation algorithm for the minimum bounded degree arborescence problem (and more generally for problems with intersecting supermodular connectivity requirements) for any $0 < \epsilon \leq \frac{1}{2}$ (see also [17] for a slight improvement). Moreover, they obtained the first additive approximation algorithm for the bounded degree arborescence problem which violates the degrees by at most an additive constant two. In order to obtain additive guarantees on the degree bounds, however, the cost of the arborescence becomes unbounded. They showed that this cost-degree tradeoff in their result is indeed best possible using the natural linear programming relaxation [2], which is an exact formulation when there are no degree constraints. In contrast, our results show that in *undirected* graphs it is possible to achieve additive approximation on the degree bounds for the minimum bounded degree Steiner network problems, while matching the best known approximation on the cost. Finally, we remark that both results are based on the iterative relaxation method in [9, 14, 21], which provides a unifying framework to achieve (nearly) tight analysis for the natural linear programming relaxations for network design problems.

2 Minimum Bounded Degree Steiner Forests

In the minimum bounded degree Steiner forest problem, we are given a graph $G = (V, E)$, a cost c_e on each edge e, a degree bound b_v for each vertex $v \in V$, and a set of source sink pairs (s_i, t_i). The task is to return a Steiner forest F (a forest that connects each source sink pair) of minimum cost with $\deg_F(v) \leq b_v$ for all $v \in V$. Let OPT be the cost of an optimal solution that satisfies all the degree bounds. We shall give a polynomial time algorithm that returns a Steiner forest F of cost at most 2OPT with $\deg_F(v) \leq b_v + 3$ for all $v \in V$.

2.1 Preliminaries

We begin by formulating a linear program for the problem. Define $f(S) = \max_{u \in S, v \notin S} \{r_{uv}\}$ for each subset $S \subseteq V$. For the Steiner forest problem, $f(S) \in \{0, 1\}$ since $r_{uv} \in \{0, 1\}$ for all $u, v \in V$. It is known that f is a *skew supermodular* function [9], that is, for every two subsets X and Y, either

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y), \text{or}$$

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

For a subset $E' \subseteq E$, we denote $x(E') := \sum_{e \in E'} x_e$. For a subset $S \subseteq V$, $\delta(S)$ denotes the set of edges in E with exactly one endpoint in S, and $d(S) := |\delta(S)|$. For a vertex $v \in V$, we write $\delta(v)$ for $\delta(\{v\})$ and $d(v)$ for $d(\{v\})$. 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 $W \subseteq V$.

[1] An $(\alpha, f(b_v))$-approximation algorithm for the minimum bounded degree Steiner network problem is a polynomial time algorithm that returns a solution of cost at most $\alpha \cdot \text{OPT}$ 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.

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

For a set $S \subseteq V$, the corresponding constraint $x(\delta(S)) \geq f(S)$ defines a vector in $\mathbb{R}^{|E|}$: the vector has an one corresponding to each edge $e \in \delta(S)$ and a zero 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. A family of sets is *laminar*, if for any two sets in the family, either one contains the other or they are disjoint. In undirected graphs, it is well known that

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

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

for any two subsets X and Y. Since f is skew supermodular, it follows from standard uncrossing arguments (see e.g. [9, 14]) that an extreme point solution of the above linear program is characterized by a laminar family of tight constraints. The following Lemma 2.1 is proved in [14].

Lemma 2.1 *Suppose that the requirement function f of (LP) is skew supermodular. Let x be an extreme point solution of (LP) such that $0 < x_e < 1$ for all edges $e \in E$. Then, there exist a laminar family $\mathcal{L}$ of sets and a set $T \subseteq W$ such that x is the unique solution to*

$$\{x(\delta(v)) = b_v \mid v \in T\} \bigcup \{x(\delta(S)) = f(S) \mid S \in \mathcal{L}\}.$$

that satisfies the following properties:

1. *The vectors $\chi_{\delta(S)}$ for $S \in \mathcal{L}$ and $\chi_{\delta(v)}$ for $v \in T$ are linearly independent.*
2. *$|E| = |\mathcal{L}| + |T|$.*
3. *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 a *child* of R. For clarity, we will refer the vertices of forest L by *nodes*. A node with no parent is called a *root* and a node with no child is called a *leaf*. Given a node R, the *subtree rooted at R* consists of R and all its descendants. The following is an important definition that is used in several places in proofs.

Definition 2.2 (Owned by) *A vertex v is* owned *by a set S if $v \in S$ and S is the smallest set in $\mathcal{L}$ containing v.*

Minimum Bounded Degree Steiner Forest

1. Initialization $F \leftarrow \emptyset$, $f'(S) \leftarrow f(S)$ for all $S \subseteq V$.
2. While F is not a Steiner forest do
   (a) *Computing an optimal extreme point solution:*
   Find an optimal extreme point 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 $b_v + 3$, 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 one.
   (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:*
   For every set $S \subseteq V$, set $f'(S) \leftarrow f(S) - d_F(S)$.
3. Return F.

Figure 2: An iterative algorithm for the minimum bounded degree Steiner forest problem.

2.2 Iterative Algorithm

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 step to avoid a multiplicative factor of two 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 [14]. Another difference is that the relaxation step has been generalized to remove a degree constraint when a vertex has degree at most $b_v + 3$.

The Step 2a of the algorithm can be implemented in polynomial time since the separation problem for the linear programming formulation is the min-cut problem [9]. Moreover, the number of the iterations is bounded by $m + n$, since in each iteration we either remove a degree constraint or pick an edge. Hence, the algorithm can be implemented in polynomial time. The following lemma is in the heart of the algorithm, which shows that the algorithm always succeeds.

Lemma 2.3 *Every extreme point solution x to the above linear program 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 degree at most $b_v + 3$.*

We note that the updated connectivity requirement function f' is also a skew supermodular function. With Lemma 2.3, using a simple inductive argument as in [14], 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 $b_v + 3$. The rest of this section is devoted to the proof of Lemma 2.3.

2.3 A New Counting Argument

The proof of Lemma 2.3 is by contradiction. Let $\mathcal{L}$ be the laminar family and $T \subseteq W$ be the set of tight vertices defining the extreme point optimal solution x as described 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 reassigned such that each member of $\mathcal{L}$ and each vertex in W get at least two tokens, and there are still some extra tokens left. This will give us a contradiction to property 2 of Lemma 2.1.

Definition 2.4 (Heavy edge) *An edge e is* heavy *if $x_e \geq 1/2$.*

Assumption 2.5 *If conditions of Lemma 2.3 do not hold, then*

1. *there is no 0-edge and no 1-edge,*
2. *every heavy edge has an endpoint in W,*
3. *each vertex $v \in W$ has at least $b_v + 4 \geq 5$ edges incident to it.*

Initial token assignment scheme: An edge $e = (u, v)$ has two tokens. One token of e is assigned to u and the other token of e is assigned to v except in the following special rule when we assign the tokens as follows.

1. 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 v from e is given to the smallest set $S \in \mathcal{L}$ containing both u and v. If such a set S does not exist, then that token is unassigned (i.e. an extra token that will not be used).

   We note that v gives up at most two tokens by this rule. This is because each such edge e is a heavy edge in $\delta(R)$ where R be the smallest set in $\mathcal{L}$ that contains v, and $f(R) = 1$ and so there are at most two heavy edges in $\delta(R)$. This is where we use the fact that $r_{\max} = 1$ for the Steiner forest problem.

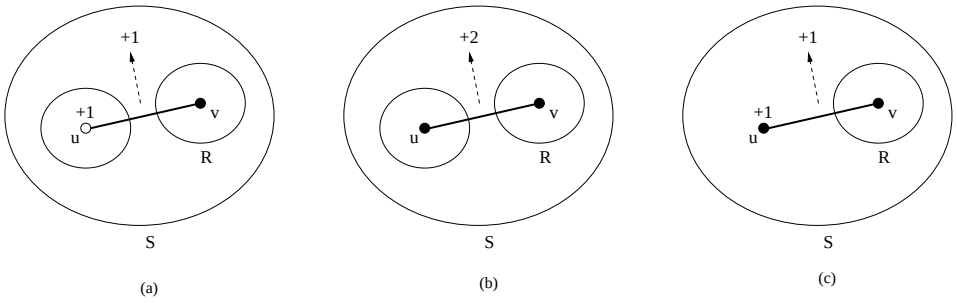

Figure 3: Rule (1) of the initial token assignment scheme. This is a new rule that is useful in collecting extra tokens for S. In this figure, a vertex is black if it is in W, and it is white if it is not in W. In (a), v gives its token from e to S by rule (1). In (b), rule (1) applies to both u and v, and thus both u and v give its token from e to S. In (c), rule (1) only applies to v and so u keeps its token. Note that uv must be a heavy edge for this rule to apply.

The following definitions are important to the analysis.

Definition 2.6 (Out-heavy edge) *An edge $e = \{u, v\}$ is an* out-heavy edge *of $S \in \mathcal{L}$ if $u \in S \setminus W$ and $v \in W \setminus S$ and $x_e \geq \frac{1}{2}$.*

Definition 2.7 (Classes) *For a set $S \in \mathcal{L}$, we say S is of:*

- Class Ia*: if $|\delta(S)| = 2$ and S has one out-heavy edge e with $x_e > \frac{1}{2}$.*

- Class Ib*: if S has two out-heavy edges.*
- Class IIa*: if $|\delta(S)| = 3$ and $x_e < \frac{1}{2}$ for each edge $e \in \delta(S)$.*
- Class IIb*: if S has one out-heavy edge but S is not of Class I.*
- Class III*: otherwise.*

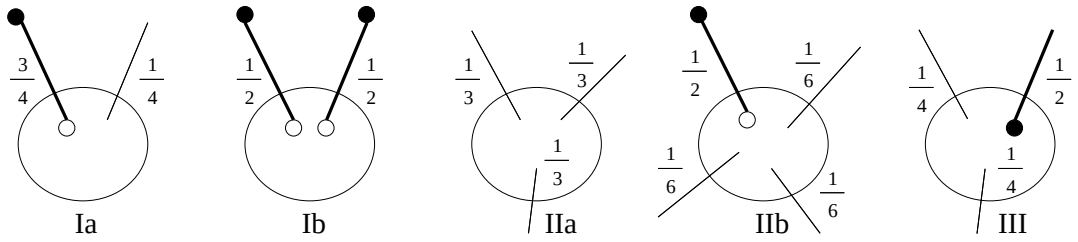

Figure 4: The figure shows examples of sets of each class. A vertex is white if it is not in W, and it is black if it is in W. (An endpoint without a vertex shown means that this information is not important.) A heavy edge is represented by a thick line. Note the definitions of Class Ia, Class Ib and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

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

Lemma 2.8 *Suppose that Assumption 2.5 holds. Then, for any subtree of the laminar family $\mathcal{L}$ rooted at S, we can reassign tokens assigned to vertices in S and nodes $R \subseteq S$ in the initial token assignment scheme such that*

1. *Every vertex in $W \cap S$ gets at least two tokens.*
2. *Every node $R \subset S$ and in the subtree gets at least two tokens.*
3. *S gets at least two tokens if S is of Class I.*
4. *S gets at least three tokens if S is of Class II.*
5. *S gets at least four tokens if S is of Class III.*

Before proving Lemma 2.8, we show that Lemma 2.3 follows from Lemma 2.8. Suppose by contradiction that the conditions in Lemma 2.3 do not hold. Then Assumption 2.5 holds. Initially, each edge in E is assigned two tokens, and so there are totally $2|E|$ tokens. After the reassignment of the tokens by applying Lemma 2.8 to each of the root nodes in $\mathcal{L}$, each vertex in W gets two tokens and each node in the laminar family gets two tokens, and so there are $2|W| + 2|\mathcal{L}|$ tokens. If there are some extra token left, then this would imply that $2|E| > 2|W| + 2|\mathcal{L}| \geq 2|T| + 2|\mathcal{L}|$, contradicting property 2 of Lemma 2.1. To complete the proof, we just need to show that there are some extra tokens left. If there is a root node S in $\mathcal{L}$ of Class II or Class III, then there is some extra token left in S as it has at least three tokens. It remains to consider the case when all root nodes are of Class I. Let S be one such root node and uv be an out-heavy edge of S with $v \in W - S$. Note that v cannot be in another root node by the definition of Class I nodes. Therefore, v is not contained in any root node. By Assumption 2.5(3), each vertex $v \in W$ has at least five edges incident on it. As Rule (1) does not apply to v, it gets at least five tokens from the initial token assignment scheme, and thus there is some extra token left on v. To summarize, if the conditions of Lemma 2.3 do not hold, then it would lead to a contradiction by Lemma 2.8. Henceforth, it remains to prove Lemma 2.8.

2.4 Proof of Lemma 2.8

Throughout the proof we assume that Assumption 2.5 holds. The proof will follow from induction on the height of node S in the rooted directed forest L. Before we proceed with the induction, we give the following lemma which shows that vertices in W are assigned many tokens.

Claim 2.9 *Each vertex $w \in W$ is assigned at least four tokens by the initial token assignment scheme. Let S be the set that owns w. Then w is assigned exactly four tokens only if:*

1. *$d(w) = 5$, $b_w = 1$, and there is one heavy edge of S in $\delta(w)$.*
2. *$d(w) = 6$, $b_w = 2$, and there are two heavy edges of S in $\delta(w)$.*

Proof: The initial token assignment scheme assigns one token to w for each of its incident edges e, except when Rule (1) applies in which case e is a heavy edge in $\delta(S)$. Since S is a tight set in the laminar family $\mathcal{L}$, we have $f(S) = 1$, and therefore $\delta(S)$ can have at most two heavy edges. By Assumption 2.5(3), $d(w) \geq b_w + 4 \geq 5$. There are two cases to consider. The first case is when there is at most one heavy edge of $\delta(S)$ incident at w. Then, w is assigned at least $d(w) - 1 \geq 4$ tokens, and it is exactly four only if $d(w) = 5$.

The second case is when there are two heavy edges of $\delta(S)$ incident at w. Then $x(\delta(w) \cap \delta(S)) \geq 1$. Combining this with Assumption 2.5(1) that there are no edges e with $x_e = 0$, we get that $b_w = x(\delta(w)) > 1$ as the two heavy edges contribute one and the other edges contribute some positive value. Being an integer, $b_w \geq 2$. Therefore, $d(w) \geq b_w + 4 \geq 6$, and so w receives at least $d(w) - 2 \geq 4$ tokens, and it is exactly four only if $d(w) = 6$. □

Now we proceed with the induction. First we prove the base case when the node S is a leaf node of the forest L.

Claim 2.10 (Base Case) *Lemma 2.8 is true for leaf nodes of $\mathcal{L}$.*

Proof: Let $S \in \mathcal{L}$ be a leaf node in the laminar family. Let $l = |S \cap W|$ be the number of vertices in W that S owns. We prove the claim by considering different cases of l. If $l \geq 2$, then each of the l vertices in W has degree at least five and receives one token for each edge incident at it in the initial token assignment scheme, except for heavy edges incident at it whose other endpoint is not in S. But since $x(\delta(S)) = 1$, there can be at most two such heavy edges. Hence, vertices in $W \cap S$ receive at least $5l - 2 \geq 2l + 4$ tokens. Thus we can reassign these tokens such that each of l vertices gets at two tokens and S gets four tokens.

If $l = 1$, let w be the only vertex in W owned by S, then w has at least four tokens by Claim 2.9. So, S can collect at least two tokens from w, and only needs at most two more tokens. Since $\chi_{\delta(S)}$ and $\chi_{\delta(w)}$ are linearly independent, there must be at least one edge e in $(\delta(S) \setminus \delta(w)) \cup (\delta(w) \setminus \delta(S))$. Each such edge has at least one endpoint $u \notin W$ in $S - w$, and u is assigned one token by the initial token assignment scheme, and thus S can collect one token from u. So, if there are at least two such edges, then S can collect two more tokens. We will argue that there must be at least two such edges. Suppose to the contrary that there is only one edge $e \in (\delta(S) \setminus \delta(w)) \cup (\delta(w) \setminus \delta(S))$, say $e \in \delta(w) \setminus \delta(S)$. Then $b_w = x(\delta(w)) = x(\delta(S)) + x_e = f(S) + x_e$. Since both $f(S)$ and b_w are integers, this implies that $x_e = 0$ or $x_e = 1$, a contradiction to Assumption 2.5(1). Therefore, there are at least two such edges, and so S can collect two more tokens.

The final case that we have not addressed yet is when $l = 0$. In this case, S gets at least $d(S)$ tokens. 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, proving Lemma 2.8. □

2.4.1 Induction Step

Let S be a set that has some children in $\mathcal{L}$. By induction, we assume that Lemma 2.8 holds for each child R of S. We will not reassign the tokens assigned to vertices and sets within R and use the same reassignment as given by the induction hypothesis. What we will do is to collect the excess tokens of R to give to S, i.e. R gives at least one token to S if R is of Class II, and R gives at least two tokens to S if R is of Class III. Moreover, S will collect excess tokens assigned to vertices owned by S and tokens assigned by Rule (1) in the initial token assignment scheme. We will show that these tokens are enough to satisfy Lemma 2.8 for set S.

We do the same case analysis, as we did for the base case, on the number of vertices of W owned by S, say l. While the argument in the base case was simple, here we will need to do further case analysis. The case when $l \geq 2$ is quite simple and dealt in Claim 2.11. The case when $l = 1$ is the most complex and is shown by further case analysis based on the number and classes of the children of S. This is done in Claim 2.12-2.14. The final case when $l = 0$ is dealt in Claim 2.15. We remark that Rule (1) of the initial token assignment scheme and the asymmetry in the definition of out-heavy edges are crucial in Claim 2.15 and also in many places in Claim 2.13 and Claim 2.14.

Claim 2.11 *Suppose that $S \in \mathcal{L}$ owns at least two vertices in W. Then Lemma 2.8 also holds for S.*

Proof: The argument here is exactly the same as in base case. We do it again for completeness. Suppose that S owns $w_1, \ldots, w_l \in W$ where $l \geq 2$. Each vertex w_i needs only two tokens to satisfy Lemma 2.8. We will redistribute the excess tokens from these vertices to S, so that S has at least four tokens to satisfy Lemma 2.8.

By Assumption 2.5(3), each vertex w_i is of degree at least five. By the token assignment scheme, each vertex w_i would have at least five tokens, unless it gives up some token by Rule (1). For Rule (1) to apply, there must be a heavy edge in $\delta(S)$. Since $f(S) = 1$, $\delta(S)$ can have at most two heavy edges, and so Rule (1) is applied at most twice for all vertices in W owned by S. Therefore, there are still at least $5l - 2$ tokens assigned to $w_1, \ldots, w_l$ by the token assignment scheme. Since each vertex in W owned by S needs only two tokens, there are $3l - 2$ excess tokens. If $l \geq 2$, then S can collect at least four tokens from these excess tokens, and thus Lemma 2.8 holds for S. □

Next, we consider the case when S owns exactly one vertex in W. We will prove that S satisfies Lemma 2.8 in the following three claims. First, we begin with an easy claim.

Claim 2.12 *Suppose that S owns a vertex w in W. If S has a Class III child or at least two Class II children, then Lemma 2.8 holds for S.*

Proof: Let S own a vertex $w \in W$. By Claim 2.9, w has at least four tokens, and so S can collect at least two tokens from w. If S has a Class III child R, then R has four tokens and S can collect two tokens from R. Similarly, if S has two Class II children R_1 and R_2, then S can collect one token from each. In any case, S can collect at least four tokens, proving the claim. □

The following claim solves the case when S has one Class II child.

Claim 2.13 *Suppose that S owns a vertex w in W. If S has at least one Class II child, then Lemma 2.8 holds for S.*

Proof: Let the children of S be $R_1, \ldots, R_l$ and R_1 be a class II child. If S has at least two Class II children or at least one Class III child, then Lemma 2.8 holds for S by Claim 2.12. So, we assume that S has no Class III child and at most one Class II child, and $R_2, \ldots, R_l$ are Class I children. Since R_1 is a class II child and has at least three tokens, S can collect at least one token from R_1. If w is assigned at least five tokens, then S

can collect three more tokens from R_1, and that would be enough for Lemma 2.8. Therefore, by Claim 2.9, we assume that w is assigned exactly four tokens, and there is at least one heavy edge f in $\delta(w) \cap \delta(S)$. Then S can collect two more tokens from w, and needs one more token. If S owns an endpoint of an edge in E, then S can collect one more token. So, we further assume that S does not own a vertex. The assumptions in this claim so far are summarized in Figure 5. We will distinguish two cases to finish the proof. Recall that S has already collected three tokens, and just needs one more token (if S is of Class III).

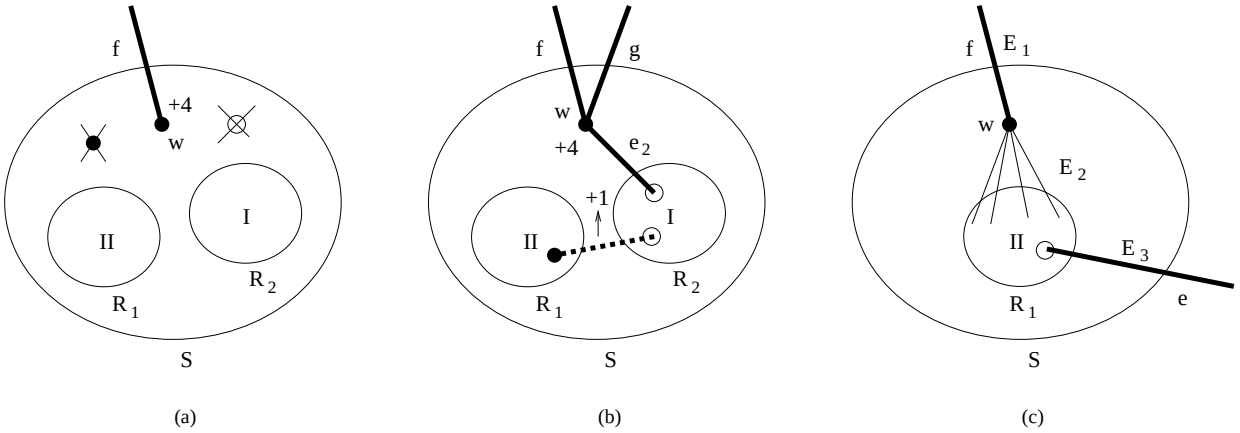

Figure 5: This is the case analysis when S owns a vertex in W and has one Class II child R_1.
In (a), we summarize the assumptions after the first paragraph of Claim 2.13: (1) R_1 is a Class II child, while all other children (if any) are Class I children; (2) w is the only vertex in W owned by S, and it received exactly four tokens from the token assignment scheme, and f is a heavy edge in $\delta(w) \cap \delta(S)$. (3) S does not own any vertex other than w.
In (b), we summarize the argument in the first case. In the first paragraph, we argue that we can assume that any out-heavy edge of R_i for $i \geq 2$ has w as an endpoint. The interesting case here is that if the other endpoint is in R_j for $j \neq i$, then S can collect one more token by Rule (1) of the token assignment scheme. In the second paragraph, we show that this implies that $d(w) = 6$ and there are two heavy edges f, g in $\delta(S)$. In the third paragraph, we argue that the remaining case left is when R_2 is the only Class I child of S and R_2 is of Class Ia. Finally, in the fourth paragraph, we show that the remaining case is not possible.
In (c), we summarize the argument in the second case. We argue that $E_3 \neq \emptyset$, and $x(E_1) = x(E_2) = x(E_3) = 1/2$. Then we argue that R_1 must be of Class IIb, and E_3 has an out-heavy edge, and so S is also of Class IIb.

1. The first case is when there are some Class I children in S. Let R_i be a Class I child for $i \geq 2$. By the definition of Class I, R_i has an out-heavy edge uv with $u \in R_i \setminus W$ and $v \in W \setminus R_i$. Since S does not own any vertex other than w (see Figure 5(a)), either $v = w$ or $v \in R_j$ for $j \neq i$ or $v \notin S$. In the second case when $v \in R_j$ for $j \neq i$, then this edge gives one token to S by Rule (1) of the token assignment scheme, and thus S can collect one more token, and we are done. In the last case when $v \notin S$, then uv is also an out-heavy edge of S, and thus S is of Class IIb and only requires three tokens, and we are also done. Hence, we assume that every out-heavy edge of R_i has w as an endpoint, as shown Figure 5(b).

   Let e_2 be an out-heavy edge of R_2. Then $b_w = x(\delta(w)) > x_{e_2} + x_f \geq 1$, as both e_2 and f are heavy and other edges in $\delta(w)$ (which exist as $d(w) \geq 5$ by Assumption 2.5(3)) have positive values by Assumption 2.5(1). This implies that $b_w \geq 2$ as b_w is an integer, and so $d(w) \geq 6$ by Assumption 2.5(3). We assumed in the first paragraph that w is assigned exactly four tokens in the token assignment scheme (see Figure 5(a)). By Claim 2.9, this implies that $b_w = 2$ and $d(w) = 6$ and there are two heavy edges in $\delta(S) \cap \delta(w)$.

   Let g be the other heavy edge in $\delta(w) \cap \delta(S)$. If there is another heavy edge e_3 incident on w, then $b_w = x(\delta(w)) > x_{e_2} + x_{e_3} + x_f + x_g \geq 2$, as $d(w) = 6$ and every edge has a positive value by

Assumption 2.5(1), contradicting $b_w = 2$. So, there are no other heavy edges incident on w except f, g, e_2. Therefore, R_2 is the only Class I child in S and R_2 is of Class Ib (as each other Class I child has at least one other out-heavy edge with w as an endpoint, and if R_2 is of Class Ib child then it has another out-heavy edge with w as an endpoint).

We will show that this remaining case is not possible. As we have assumed that S does not own any endpoint other than w (see Figure 5(a)), it follows that $\delta(w) \setminus \delta(S) = \delta(w, R_1) \cup \delta(w, R_2)$, and $\delta(R_1) = \delta(w, R_1) \cup \delta(R_1, R_2)$, and $\delta(R_2) = \delta(w, R_2) \cup \delta(R_1, R_2)$. Since $b_w = 2$ and $x(\delta(w) \cap \delta(S)) = 1$, it follows that $x(\delta(w) \setminus \delta(S)) = 1$. Also, $x(\delta(R_1)) = x(\delta(R_2)) = 1$ as $f(R_1) = f(R_2) = 1$. Therefore, we must have $x(\delta(w, R_1)) = x(\delta(w, R_2)) = x(\delta(R_1, R_2)) = 1/2$. Hence, R_2 cannot be of Class Ia, as otherwise $x(\delta(w, R_2)) \geq x_{e_2} > \frac{1}{2}$.

2. The second case is when R_1 is the only child of S. Consider the three edge sets $E_1 = \delta(w) \cap \delta(S)$, $E_2 = \delta(w, R_1)$ and $E_3 = \delta(R_1) \cap \delta(S)$; see Figure 5(c). As we have assumed that S does not own any endpoint other than w (see Figure 5(a)), it follows that $\delta(w) = E_1 + E_2$, $\delta(R_1) = E_2 + E_3$ and $\delta(S) = E_1 + E_3$. Since $\chi_{\delta(S)}, \chi_{\delta(w)}, \chi_{\delta(R_1)}$ are linearly independent, this implies that $E_3 \neq \emptyset$, as otherwise $\chi_{\delta(S)} = \chi_{\delta(w)} - \chi_{\delta(R_1)}$. Note that $x(E_1) + x(E_2) = x(\delta(w)) = b_w$, and $x(E_2) + x(E_3) = x(\delta(R_1)) = f(R_1) = 1$, and $x(E_1) + x(E_3) = x(\delta(S)) = f(S) = 1$. As $b_w \geq 1$ is an integer and $x(E_3) > 0$, the only solution is $b_w = 1$ and $x(E_1) = x(E_2) = x(E_3) = 1/2$. This implies that R_1 cannot be of Class IIa; otherwise either E_2 and E_3 consists of a single heavy edge (since $|\delta(R_1)| = |E_2| + |E_3| = 3$ for a Class IIa child), contradicting that R_1 is of Class IIa. So, R_1 must be of Class IIb, and thus there is an out-heavy edge e in $\delta(R_1)$. If $e \in E_2$, then $b_w > x_e + x_f \geq 1$, as $d(w) \geq 5$ and other edges have positive values, contradicting that $b_w = 1$. Therefore, $e \in E_3$. Then, e is also an out-heavy edge of S. So, S is of Class IIb and only needs three tokens, and we are done.

We have considered all the cases when S has at least one Class II child, and Lemma 2.8 holds for S. □

The following claim solves the case when S has only Class I children, which has many cases in the proof.

Claim 2.14 *Suppose that S owns a vertex in W. Then Lemma 2.8 holds for S.*

Proof: By Claim 2.12 and Claim 2.13, if S has at least one Class II child or one Class III child, then Lemma 2.8 holds for S. So it remains to consider the case when S has only Class I children. Let $R_1, \ldots, R_l$ be the Class I children of S. By the definition of Class I, there are no heavy edges with one endpoint in one Class I child and another endpoint in another Class I child. Let h be the number of out-heavy edges of $R_1 \cup \ldots \cup R_l$ that are also in $\delta(S)$. Notice that it suffices to collect $4 - h$ tokens for S to prove the claim. In the following, we distinguish two cases, when $h \geq 1$ and when $h = 0$.

1. We consider the case when $h \geq 1$. See Figure 6 for an illustration. Let e_1 be an out-heavy edge in $\delta(S) \cap \delta(R_1)$. Then S is of Class IIb, and only needs three tokens. If w is assigned at least five tokens by the token assignment scheme, then S can collect three tokens from w, and we are done. So, by Claim 2.9, we assume that w has exactly four tokens. Then S can collect two tokens from w, and needs only one more token. As $f(S) = 1$ and there is already a heavy edge $e_1 \in \delta(S)$, Case (2) of Claim 2.9 cannot happen. So, by Claim 2.9, the only possibility left is that $b_w = 1$, $d(w) = 5$ and there is one heavy edge f in $\delta(w) \cap \delta(S)$. This implies that $\delta(S) = \{e_1, f\}$, as $f(S) = 1$ and e_1, f are heavy edges. Suppose that S has another Class I child R_2. Then R_2 has an out-heavy edge e_2. Since $e_2 \notin \delta(S)$, this implies that $e_2 \in \delta(w)$, as w is the only vertex in W owned by S. However, since $d(w) = 5$ and every edge has a positive value, this implies that $b_w = x(\delta(w)) > x_{e_2} + x_f \geq 1$, a contradiction to Claim 2.9. So R_2 does not exist. Hence R_1 is the only child of S. Since $d(w) = 5$, $d(R_1) = 2$, and $|\delta(w) \cap \delta(S)| = 1$, there must be an edge (w, x) with $x \in S - R_1$. So S can collect one token from x, as required.

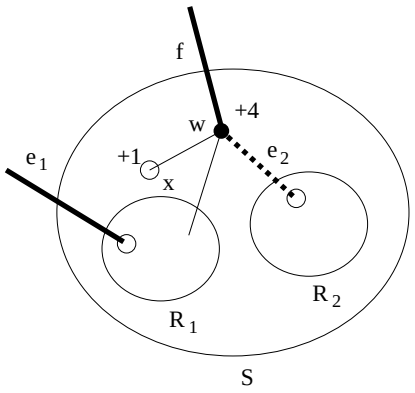

Figure 6: We illustrate the argument in the subcase when $h \geq 1$, which is in the case when S owns only one vertex w in W and all the children of S are of Class I.

2. Next we consider the case when $h = 0$. Then, every out-heavy edge of R_i is incident on w, as w is the only vertex in W that is owned by S. By the token assignment scheme, w receives one token from each of its incident edge, except when Rule (1) applies in which case e is a heavy edge in $\delta(S)$. Since $f(S) = 1$, there are at most two heavy edges in $\delta(S)$, and so w receives at least $d(w) - 2$ tokens by the token assignment scheme. Therefore, if $d(w) \geq 8$, then w has at least six tokens, and S can collect four tokens from w, and we are done. Henceforth, we assume that $7 \geq d(w) \geq 5$ (by Assumption 2.5(3)), and we further consider three subcases depending on the value of $d(w)$.

   In the following subcases, recall that we assumed that S owns only one vertex w in W, every child R_i of S is of Class I, and every out-heavy edge of R_i is incident on w.

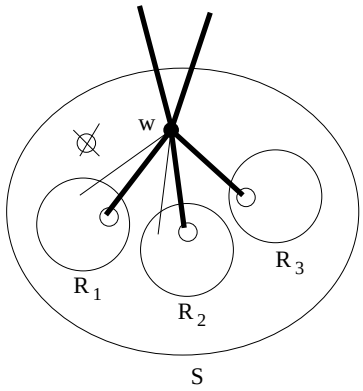

Figure 7: In this figure, we are in the case (2a) of Claim 2.14, in which S owns only one vertex w in W, every child R_i of S is of Class I, every out-heavy edge of R_i is incident on w, and furthermore $d(w) = 7$. We further restrict this subcase to the scenario when there are two heavy edges in $\delta(w) \cap \delta(S)$, S does not own any vertex other than w, and $\delta(w) \setminus \delta(S) = \delta(w, \cup_{i=1}^{l} R_i)$, and argue that it is not possible.

   (a) Suppose that $d(w) = 7$. If there is at most one heavy edge in $\delta(w) \cap \delta(S)$, then w receives at least six tokens by the token assignment scheme, and S can collect four tokens from w, and this is enough. So, assume that there are two heavy edges in $\delta(w) \cap \delta(S)$, and hence $|\delta(w) \cap \delta(S)| = |\delta(S)| = 2$. Then, w receives exactly five tokens by the token assignment scheme, and S can collect three tokens from w, and it needs one more. If S owns an endpoint other than w, then

S can collect one more token, and we are done. So, we further assume that S does not own an endpoint other than w. Therefore, $\delta(w) \setminus \delta(S) = \delta(w, \cup_{i=1}^{l} R_i)$. See Figure 7 for an illustration.

We argue that the remaining case is not possible. Note that $|\delta(w, \cup_{i=1}^{l} R_i)| = |\delta(w) \setminus \delta(S)| = |\delta(w)| - |\delta(w) \cap \delta(S)| = 7 - 2 = 5$. Since $d(R_i) = 2$ for each Class I child, this implies that $l \geq 3$. Each R_i has an out-heavy edge e_i incident on w. If $l \geq 4$, since $d(w) = 7$ and every edge has a positive value, it follows that $b_w = x(\delta(w)) > x(\delta(w) \cap \delta(S)) + x_{e_1} + x_{e_2} + x_{e_3} + x_{e_4} \geq 3$, a contradiction to $d(w) \geq b_w + 4$ by Assumption 2.5(3). Therefore, S must have exactly three children R_1, R_2, R_3. Since $|\delta(w, \cup_{i=1}^{l} R_i)| = 5$, there are exactly two children with $|\delta(R_i)| = |\delta(w, R_i)| = 2$, say R_1 and R_2. Again, $b_w = x(\delta(w)) = x(\delta(w) \cap \delta(S)) + x(\delta(R_1)) + x(\delta(R_2)) + x_{e_3} \geq 1 + 1 + 1 + \frac{1}{2} > 3$, a contradiction to $d(w) \geq b_w + 4$ by Assumption 2.5(3).

(b) Suppose $d(w) = 6$. This implies that $b_w \leq 2$ by Assumption 2.5(3). Also, since $d(w) = 6$ and every edge has a positive value, there are at most three heavy edges incident to w.

If there is no heavy edge in $\delta(w) \cap \delta(S)$, then w receives at least six tokens by the token assignment scheme, and S can collect four tokens from w, which is enough. So, assume that there is at least one heavy edge in $\delta(w) \cap \delta(S)$.

Suppose that there are two heavy edges in $\delta(w) \cap \delta(S)$. See Figure 8(a) for an illustration. Then w receives at least four tokens by the token assignment scheme, and S can collect two tokens from w, and it needs two more. Since $f(S) = 1$, $|\delta(w) \cap \delta(S)| = |\delta(S)| = 2$. Each child R_i is of Class I, and has one out-heavy edge incident on w. As there are at most three heavy edges incident to w, the only possibility is that S has exactly one child R_1. Since $d(w) = 6$, $|\delta(w) \cap \delta(S)| = 2$ and $|\delta(w) \cap \delta(R_1)| \leq 2$, there are at least two edges in $\delta(w)$ with the other endpoint in $S - R_1$, and so S can collect two tokens from these two endpoints, and we are done.

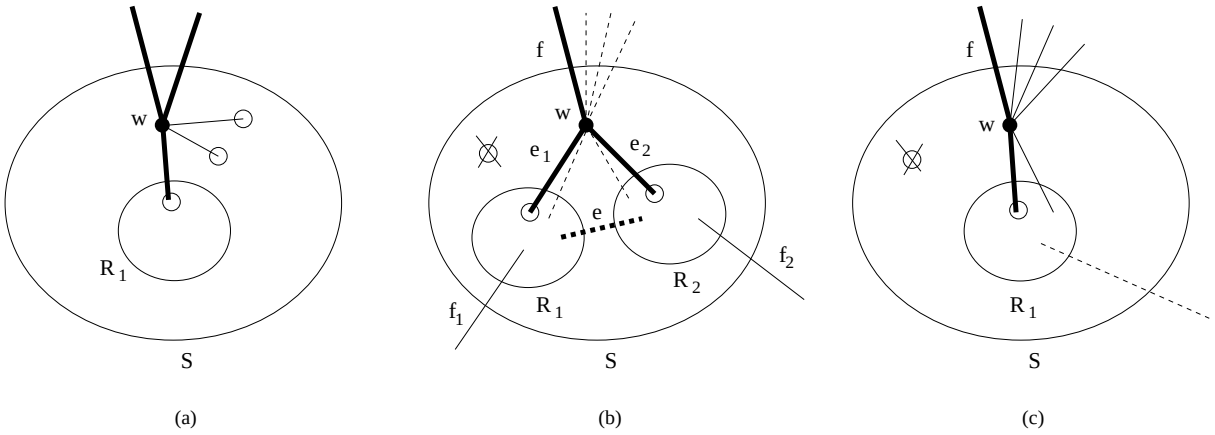

Figure 8: In this figure, we are in the case (2b) of Claim 2.14, in which S owns only one vertex w in W, every child R_i of S is of Class I, every out-heavy edge of R_i is incident on w, and furthermore $d(w) = 6$. We further argue that there are at most three heavy edges incident to w.
In (a), we consider the scenario when there are two heavy edges in $\delta(w) \cap \delta(S)$, in which S can collect two tokens from w and two tokens from two other endpoints that it owns.
In (b), we consider the scenario when there is exactly one heavy edge f in $\delta(w) \cap d(S)$ and S has exactly two children R_1 and R_2. We further assume that S does not own any vertex other than w. Then, no matter how we place the other edge of R_i, we reach a contradiction.
In (c), we consider the scenario when there is exactly one heavy edge f in $\delta(w) \cap d(S)$ and S has exactly one child R_1. We further assume that S does not own any vertex other than w, but then it implies that the characteristic vectors are linearly dependent.

Henceforth, we assume that there is exactly one heavy edge f in $\delta(w) \cap \delta(S)$. By the token assignment scheme, w receives at least five tokens. So, S can collect three tokens from w, and it needs only one more. If S owns another endpoint other than w, then S can collect one more token, as required. So, we further assume that S does not own an endpoint. We assumed that each R_i has an out-heavy edge e_i incident on w (as $h = 0$ in Case (2); see the caption in Figure 8). As there are at most three heavy edges incident on w (as $b_w \leq 2$ in Case (2b); see the caption in Figure 8) and one of these heavy edges is in $\delta(S)$, S has at most two children. In the following two paragraphs, we will consider the two scenarios when S has exactly two children (Figure 8(b)) and S has exactly one child (Figure 8(c)). Recall that S only needs to collect one more token.

Suppose S has exactly two children R_1 and R_2. We will show in this paragraph that this case is not possible; see Figure 8(b). Since each R_i has an out-heavy edge e_i incident on w, there are three heavy edges incident on w, Thus $b_w > 1$, and so $b_w = 2$ (as we assumed $b_w \leq 2$ in Case (2b)). If $\delta(w, R_i) = \delta(R_i)$, then $b_w > x_f + x(\delta(R_1)) + x_{e_2} \geq 1/2 + 1 + 1/2 = 2$, as $d(w) = 6$ and every edge has a positive value, a contradiction. Hence, we assume that $\delta(w, R_1) = \{e_1\}$ and $\delta(w, R_2) = \{e_2\}$. Suppose $\delta(R_1, R_2) \neq \emptyset$. Let $\delta(R_1, R_2) = \{e\}$. Then, $x_e + x_{e_1} = x(\delta(R_1)) = f(R_1) = 1$, and $x_e + x_{e_2} = x(\delta(R_2)) = f(R_2) = 1$, and $x_{e_1} + x_{e_2} = x(\delta(w)) - x(\delta(w) \cap \delta(S)) = b_w - f(S) = 1$. Therefore, $x_e = x_{e_1} = x_{e_2} = 1/2$, but then e is a heavy edge between two Class I children, a contradiction to the definition of Class I. So, we must have $\delta(R_1, R_2) = \emptyset$. Let $R_1 = \{e_1, f_1\}$ and $R_2 = \{e_2, f_2\}$. The only possibility left is $f_1 \in \delta(S)$ and $f_2 \in \delta(S)$. Note that $x_{e_1} + x_{f_1} = x(\delta(R_1)) = f(R_1) = 1$, and $x_{e_2} + x_{f_2} = x(\delta(R_2)) = f(R_2) = 1$, and $x_{e_1} + x_{e_2} + x(\delta(w) \cap \delta(S)) = x(\delta(w)) = b_w = 2$, and $x_{f_1} + x_{f_2} + x(\delta(w) \cap \delta(S)) = x(\delta(S)) = f(S) = 1$. Solving these equations, we have $x(\delta(w) \cap \delta(S)) = 1/2$. Since $f \in \delta(w) \cap \delta(S)$ is a heavy edge, this implies that f is the only edge in $\delta(w) \cap \delta(S)$. But then $\delta(w) = \{e_1, e_2, f\}$, and hence $d(w) = 3$, a contradiction.

The final remaining case is when S has exactly one child R_1, and there is exactly one heavy edge f in $\delta(w) \cap \delta(S)$. See Figure 8(c) for an illustration. Recall that S only needs to collect one more token, when there is exactly one heavy edge f in $\delta(w) \cap \delta(S)$. If S owns an endpoint other than w, then S can collect one more token, as required. So, we further assume that S does not own an endpoint other than w. We prove that this case would not happen. Note that $x(\delta(w) \cap \delta(S)) + x(\delta(w, R_1)) = x(\delta(w)) = b_w = 2$, and $x(\delta(w, R_1)) + x(\delta(R_1) \cap \delta(S)) = x(\delta(R_1)) = f(R_1) = 1$, and $x(\delta(w) \cap \delta(S)) + x(\delta(R_1) \cap \delta(S)) = x(\delta(S)) = f(S) = 1$. Solving these equations, we have $x(\delta(R_1) \cap \delta(S)) = 0$. Therefore, $\delta(w, R_1) = \delta(R_1)$, and hence $\chi_{\delta(w)} = \chi_{\delta(R_1)} + \chi_{\delta(S)}$, contradicting the linear independence of these characteristic vectors.

(c) Suppose $d(w) = 5$. See Figure 9 for an illustration. This implies that $b_w \leq 1$ by Assumption 2.5(3), and hence $b_w = 1$. Also, since $d(w) = 5$ and every edge has a positive value, there is at most one heavy edge incident on w. Each child R_i has a heavy edge e_i incident on w. So, S has exactly one child R_1, and there is no heavy edge in $\delta(w) \cap \delta(S)$. By the token assignment scheme, w receives five tokens, and S can collect three tokens from w, and it needs only one more. If S owns an endpoint, then S can collect one more token, as required. So, we further assume that S does not own an endpoint.

We prove that this remaining case is not possible. Note that $x(\delta(w) \cap \delta(S)) + x(\delta(w, R_1)) = x(\delta(w)) = b_w = 1$, and $x(\delta(w, R_1)) + x(\delta(R_1) \cap \delta(S)) = x(\delta(R_1)) = f(R_1) = 1$, and $x(\delta(w) \cap \delta(S)) + x(\delta(R_1) \cap \delta(S)) = x(\delta(S)) = f(S) = 1$. Solving these equations, we have $x(\delta(R_1) \cap \delta(S)) = x(\delta(w, R_1)) = x(\delta(w) \cap \delta(S)) = 1/2$. Since $d(R_1) = 2$, there is only one edge $e \in \delta(R_1) \cap \delta(S)$, having $x_e = 1/2$. By the definition of Class I, e must be an out-heavy edge of R_1 in $\delta(S)$, contradicting $h = 0$ assumed in Case (2).

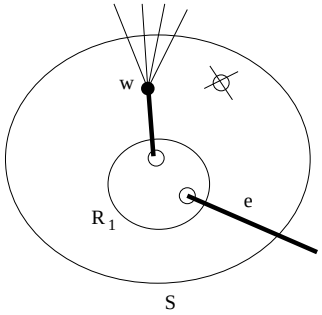

Figure 9: In this figure, we are in the case (2c) of this claim, in which S owns only one vertex w in W, every child R_i of S is of Class I, every out-heavy edge of R_i is incident on w, and furthermore $d(w) = 5$. We further restrict this to the scenario when S has only one (Class I) child R_1, there is no heavy edge in $\delta(w) \cap \delta(S)$, and S does not own any vertex other than w.

We have considered all the cases and completed the proof of Claim 2.14. □

Proof of Lemma 2.8: We now complete the proof of Lemma 2.8 by solving the case when S does not own any vertex in W. Let h be the number of out-heavy edges in S, and let t be the number of tokens that S can collect (from the vertex that it owns, from the excess tokens that its children have, or from the token that it receives from Rule (1) of the token assignment scheme). In the following, we say that a node R is of *Type A* if R is of Class Ia or of Class IIa. To prove Lemma 2.8, it suffices to have $h + t \geq 4$ if S is not of Type A, and $h + t \geq 3$ if S is of Type A.

Claim 2.15 *Suppose that S does not own a vertex in W. 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 two excess tokens, and so it contributes two to t. And if R is of Class IIa, then it has one excess token, and so it contributes one to t.

If R is of Class IIb, then it has one excess token which contributes one to t, and one out-heavy edge $e \in \delta(R)$. If $e \in \delta(S)$, then e contributes one to h. Otherwise, since S does not own any vertex in W, the other endpoint of e is in some other child R' of S. Then, e contributes one to t by Rule (1) of the token assignment scheme. So, e contributes one to $h + t$, and thus R contributes two to $h + t$.

Finally, we consider Class I children. By the same argument as above, each out-heavy edge contributes one to $h + t$, 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 two out-heavy edges, and so it contributes two to $h + t$. If R is of Class Ia, then it has one out-heavy edge, and it contributes one to $h + t$. □

We are ready to finish the proof of Lemma 2.8 by considering the number of children of S. Recall that it suffices to have $h + t \geq 4$ if S is not of Type A, and $h + t \geq 3$ if S is of Type A.

1. S has at least four children. By Claim 2.15, each child can contribute at least one 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 2.15, and we are done. So, we assume that 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, we further assume that S does not own an endpoint. See Figure 10 for an illustration. We divide this case into two subcases, depending on whether S has a Class Ia child.

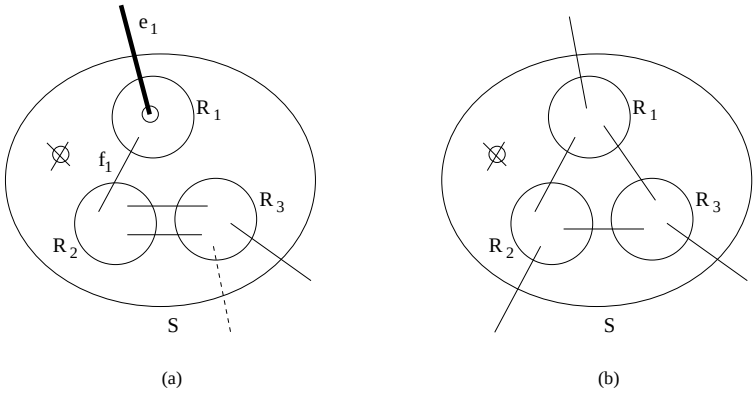

Figure 10: In this figure, we are in the case when S has exactly three children and S does not own any vertex in W. We assume that all children are of Type A, and S does not own any vertex.
In (a), we consider the subcase when S has a Class Ia child R_1. We argue that S is of Class Ia, and only require two tokens, and can collect two tokens from R_2 and R_3, which must be of Class IIa.
In (b), we consider the other subcase when S has three Class IIa children R_1, R_2, R_3. Jain's proof implies that S is also of Class IIa, and thus it only requires three tokens, and so it can collect one token from each R_i.

The first subcase is when S has a Class Ia child. Each Class Ia child R_i has an out-heavy edge e_i with $x_{e_i} > 1/2$. Note that the other endpoint of e_i cannot be in another child R_j of S for $i \neq j$, by the definition of Type A children. Also, since S does not own any vertex in W by Claim 2.14, this implies that e_i must be in $\delta(S)$. Hence, since $f(S) = 1$, S can have at most one Class I 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 $f(S) = x(\delta(S)) = x_{e_1} + x(\delta(R_3) \cap \delta(S)) > 1/2 + (1 - 1/2) = 1$, since $|\delta(R_3)| = 3$ and each edge e in $\delta(R_3)$ has $x_e < 1/2$ by the definition of Class IIa. Since $|\delta(R_2, R_3)| = 2$, this implies that $d(S) = 2$, and hence S is of Class Ia and only requires two tokens. In fact, S can collect two tokens, one token from R_2 and one token from R_3, and we are done.

The second subcase is when R_1, R_2, R_3 are all Class IIa children. We use the following lemma in Jain's proof.

Lemma 2.16 (Lemma 23.18 of [22]) *The* corequirement *of a set S is defined as $d(S)/2 - f(S)$. Suppose S has α children and owns β endpoints, where $\alpha + \beta = 3$. Furthermore, each child of S, if any, has a corequirement of $1/2$. Then, S also has a corequirement of $1/2$.*

In our case, when $f(S) = 1$, S has a corequirement $1/2$ if and only if $d(S) = 3$; in particular, a Class IIa node has a corequirement $1/2$. Therefore, Lemma 2.16 implies that S is also of Class IIa, and thus it only requires three tokens. So, S can collect three tokens, one from each R_i, as required.

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 two to $h + t$ by Claim 2.15, then we are done. We divide this case into two subcases, depending on the number of Type A children.

   The first subcase is when R_1 is of Type A and R_2 is not of Type A. See Figure 11(a) for an illustration. So, by Claim 2.15, R_1 can contribute one to $h + t$ and R_2 can contribute two to $h + t$, and so S needs only one more token. If S owns an endpoint, then we are done. So, we further assume that S does not own an endpoint. We shall prove that this would not happen. In this case, $x(\delta(R_1) \cap$

$\delta(S)) + x(\delta(R_1, R_2)) = x(\delta(R_1)) = f(R_1) = 1$, $x(\delta(R_1) \cap \delta(S)) + x(\delta(R_2) \cap \delta(S)) = x(\delta(S)) = f(S) = 1$, and $x(\delta(R_2) \cap \delta(S)) + x(\delta(R_1, R_2)) = x(\delta(R_2)) = f(R_2) = 1$, and hence we must have $x(\delta(R_1) \cap \delta(S)) = x(\delta(R_1, R_2)) = x(\delta(R_2) \cap \delta(S)) = 1/2$. Therefore, R_1 cannot be of Class Ia, since otherwise it has an edge with $x_e > 1/2$. Also, R_1 cannot be of Class IIa, since $d(R_1) = 3$, either $\delta(R_1, R_2)$ or $\delta(R_2) \cap \delta(S)$ is a single edge e with $x_e = 1/2$, contradicting that R_2 is of Class IIa.

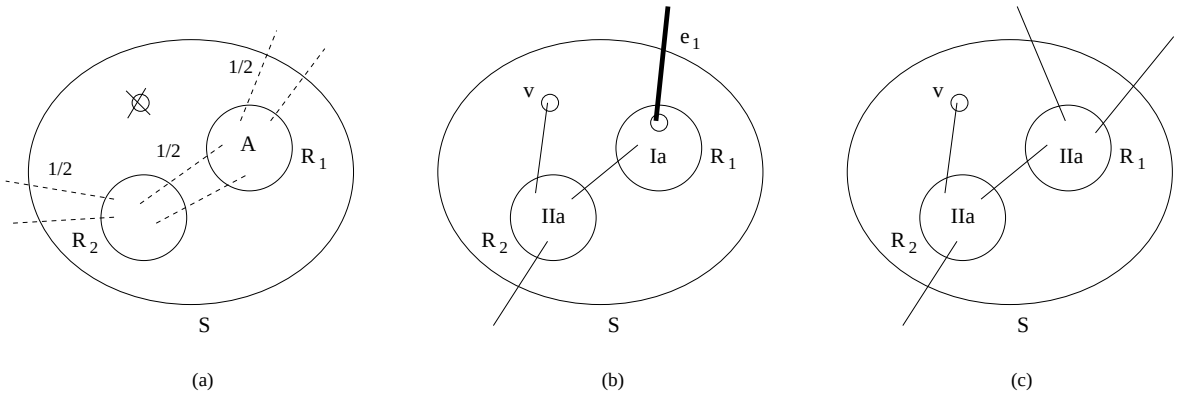

Figure 11: In this figure, we are in the case when S has exactly two children R_1, R_2 and S does not own any vertex in W.
In (a), we consider the subcase when R_1 is of Type A and R_2 is not of Type A. We further assume that S does not own a vertex. Then we argue that this cannot happen, as R_1 cannot be of Type A.
In (b), we consider the scenario when both R_1 and R_2 are of Type A and R_1 is of Class Ia. We further assume that S owns exactly one endpoint v. Then we show that S is also of Class Ia, and only requires two tokens, which can be collected from R_2 and v.
In (c), we consider the scenario when both R_1 and R_2 are of Type A and R_1 is of Class IIa. We further assume that S owns exactly one endpoint v. Then we show that S is also of Class IIa, and only requires three tokens, which can be collected from R_1, R_2 and v.

The second subcase is when both R_1 and R_2 are of Type A. By Claim 2.15, each R_i can contribute one to $h + t$, and S needs at most two more tokens. If S owns two endpoints, then we are done. By the same argument as in the above paragraph, it cannot be the case that S does not own any vertex, and so S must own at least one endpoint. So, we assume that S owns exactly one endpoint v. Suppose R_1 is of Class Ia, then its out-heavy edge e_1 must be in $\delta(S)$. See Figure 11(b) for an illustration. So, R_2 cannot be of Class Ia; otherwise $x(\delta(S)) > 1$. If R_2 is of Class IIa, then we cannot have $|\delta(R_1, R_2)| \geq 2$ since $d(R_1) = 2$, and also cannot have $|\delta(R_2) \cap \delta(S)| \geq 2$ since $f(S) = 1$. Therefore, the only possibility is $|\delta(R_1, R_2)| = |\delta(v, R_2)| = |\delta(R_2) \cap \delta(S)| = 1$. Hence, $d(S) = 2$ and so S is of Class Ia and only requires two tokens. In fact, S can collect two tokens, one from R_2 and one from v, as required. Finally, suppose R_1 and R_2 are of Class IIa. See Figure 11(c) for an illustration. Then, S is of Class IIa by Lemma 2.16, and only requires three tokens. And S can collect three tokens, one from R_1, one from R_2 also one from v, as required.

4. S has exactly one child R. By linear independence of $\chi_{\delta(S)}$ and $\chi_{\delta(R)}$ and $f(S) = f(R) = 1$, there must be one edge $f \in \delta(S) - \delta(R)$ and one edge $g \in \delta(R) - \delta(S)$. Hence, S must own at least two endpoints, and thus can collect two tokens. If R is not of Type A, then R contributes two to $h + t$ by Claim 2.15, and S has at least four tokens, and we are done. So, we assume that R is of Type A. We divide this into two subcases, depending on whether R is of Class Ia or Class IIa.

   The first subcase is when R is of Class Ia. See Figure 12(a) for an illustration. Since S does not own any vertex in W, the out-heavy edge of R must be in $\delta(S)$. Then, S is of Class IIb and only require

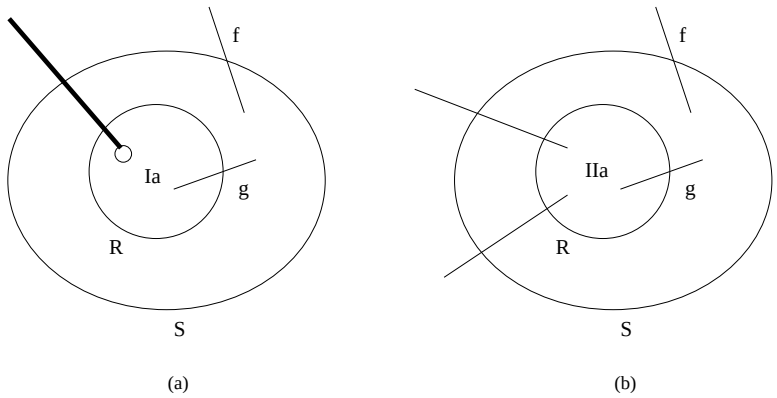

Figure 12: In this figure, we are in the case when S has exactly one child R and S does not own any vertex in W.
In (a), we consider the subcase when R is of Class Ia.
In (b), we consider the subcase when R is of Class IIa.

three tokens. If S owns at least three endpoints, then S can collect three tokens, and we are done. So, we assume that S owns exactly two endpoints. Then, we must have $x_f = x_g$ and $d(S) = 2$, and thus S is also of Class Ia and only requires two tokens, and we are also done.

The second subcase is when R is of Class IIa. See Figure 12(b) for an illustration. If S owns at least three endpoints, then S can collect four tokens, three tokens from the endpoints and one token from R, and we are done. So, we assume that S owns exactly two endpoints. Then, we must have $x_f = x_g$ and $d(S) = 3$. Since R is of Class IIa, this implies that S of also of Class IIa and only requires three tokens. Note that S can collect two tokens from the endpoints and one token from R, and we are also done.

We have considered all the cases and completed the proof of Lemma 2.8. □

As shown in the end of Section 2.3, Lemma 2.8 implies Lemma 2.3, which in turns implies Theorem 1.1.

3 Minimum Bounded Degree Steiner Networks

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

Our approximation algorithm for the minimum bounded degree Steiner network problem is also an iterative algorithm. It 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* $W_h = \{v \in W \mid x(\delta(v)) \geq 6f_{\max}\}$, where $f_{\max} := \max_{S \subseteq V} f(S)$. This set of vertices plays the same role as the set of vertices with no 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 step ensures that the degree bounds are only violated by an additive term.

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

Minimum Bounded Degree Steiner Network

1. Initialization $F \leftarrow \emptyset$, $f'(S) \leftarrow f(S)$ for all $S \subseteq V$.
2. While F is not a Steiner network do
   (a) *Computing an optimal extreme point solution:*
   Find an optimal extreme point solution x satisfying f' and remove every edge e with $x_e = 0$.
   Set $W_h = \{v \in W \mid x(\delta(v)) \geq 6f_{\max}\}$ and $b_v = x(\delta(v))$ for $v \in W$.
   (b) *Removing a degree constraint:*
   For every $v \in W$ with degree at most four, 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 one.
   (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 $S \subseteq V$, set $f'(S) \leftarrow f(S) - d_F(S)$.
3. Return F.

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 the degree bound of each vertex is violated by at most* $6r_{\max} + 3$.

Proof: The proof is by a standard inductive argument. We provide a short explanation. Note that f' in each iteration remains a skew supermodular function, and so Lemma 3.2 continues to hold. Since we always pick an edge with $x_e \geq 1/2$ and the remaining fractional solution is a feasible solution for the residual problem, the cost of the solution returned is at most twice the cost of the linear programming solution as claimed in Theorem 1.2.

Next, we bound the degree violation of a vertex v. First, observe that while $v \in W_h$, we pick at most $b_v - 6f_{\max}$ edges incident on v, as we only do that in Step 2c and the degree bound of v is reduced by one whenever such an edge is picked. When $v \in W - W_h$, we pick at most $12f_{\max} - 1$ edges incident on v, since we only do that in Step 2d or Step 2c and the degree bound is reduced by at least $1/2$ whenever we include such an edge. Finally, when $v \notin W$, we pick at most four edges incident on v, since the degree of v is at most four by Step 2b. Hence, the number of edges picked that are incident on v is at most $(b_v - 6f_{\max}) + (12f_{\max} - 1) + 4 = b_v + 6f_{\max} + 3$. □

For the correctness of the algorithm, we shall prove the following lemma in Section 3.2, which will ensure that the algorithm terminates with a feasible solution, completing 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 an extreme point solution to (LP), W be the set of vertices with degree constraints, and $W_h = \{v \in W \mid x(\delta(v)) \geq 6f_{\max}\}$. Then at least one of the following is true.*

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 $v \in W$ with $d(v) \leq 4$.

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 give two tokens for each edge (the initially 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 there are some extra tokens left. This will contradict $|E| = |\mathcal{L}| + |T|$ of Lemma 2.1, and thus completes the proof.

Our analysis is similar to Jain's analysis, the main difference being 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 some 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 at least three tokens as in Jain's proof. To overcome this, we use a different token assignment scheme so that a similar induction hypothesis as Jain's works.

Initial 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:

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

It is useful to note that this definition reduces to Jain's definition of co-requirement if every heavy edge e with $x_e \geq 1/2$ is thought of as two parallel edges, each aiming to achieve a value of $1/2$ but sharing the current x_e value equally (i.e. each gets $x_e/2$), so that summing $1/2 - x_e/2$ 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 its degree. In particular, 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 on 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. Finally, by the initial token assignment scheme, an endpoint $v \notin W$ can get two tokens from a heavy edge incident on it, because the other endpoint of the heavy edge must be in W_h.

We are ready to prove the following lemma which shows that the tokens can be reassigned as discussed previously.

Lemma 3.3 *For any subtree of L rooted at node S, we can reassign tokens initially assigned to vertices in S such that each vertex in $T \cap S$ gets at least two tokens, each node in the subtree gets at least two tokens, and the root S gets at least three tokens. Moreover, the root S gets exactly three tokens only if* $\text{coreq}(S) = 1/2$.

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 handles 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.

Proof of base case of Lemma 3.3: Let $S \in \mathcal{L}$ be a leaf node. First, consider the case when $S \cap W = \emptyset$. 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 at least 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 two

tokens by the token assignment scheme. If it does not contain a heavy edge, then S receives three tokens and coreq$(S) = 1/2$. If $|\delta(S)| = 2$, then at least one edge is a heavy edge. If both edges are heavy, then S can get four tokens; otherwise if only one edge is heavy then it gets three tokens and coreq$(S) = 1/2$.

Next, we consider the case when S owns a vertex $v \in S \cap (W \setminus W_h)$ but S does not own a vertex in W_h. By the token assignment scheme, v receives 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 the characteristic vectors as stated in Lemma 2.1.

Finally, we consider the case when 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 also when S is not a leaf in the laminar family, and will be used in the induction step.

Claim 3.4 *Suppose that the induction hypothesis holds for each child of S and that 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 O_h denote the vertices in W_h owned by S. Let $B := \sum_{v \in O_h} x(\delta(v)) = \sum_{v \in O_h} b_v \geq \sum_{v \in O_h} 6 f_{\max} = 6r f_{\max}$. Informally, vertices in W_h owned by S would have collected a total of at least B tokens if the two tokens at each edge were distributed evenly. However, by the initial token assignment scheme, some vertices in O_h may not get any token for the 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, $x(\delta(R)) = f(R) \leq f_{\max}$ and similarly $x(\delta(S)) \leq f_{\max}$. Thus

$$x((\cup_{v \in O_h} \delta(v)) \cap (\cup_R \delta(R) \cup \delta(S))) \leq x(\cup_R \delta(R) \cup \delta(S)) \leq (c+1) f_{max}.$$

Therefore,

$$\sum_{v \in O_h} \sum_{u : u \text{ owned by S}} x_{uv} \geq B - (c+1) f_{max} \geq f_{max}(6r - c - 1).$$

Since there is no 1-edge, there are at least $f_{\max}(6r - c - 1) + 1$ endpoints in the above sum where both the endpoints are owned by S. Let $e = \{u, v\}$ be such an edge with $v \in O_h$. 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 O_h 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, S can collect $2r + 4$ tokens by Claim 3.4. So, these tokens can be reassigned so that S has four tokens and each vertex in W_h owned by S has two tokens, which is enough for the induction hypothesis. □

Proof of the induction step: The presence of heavy edges with $x_e \geq 1/2$ introduces some difficulties in carrying out the inductive argument in [9]. 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 $\mathrm{wdeg}(\delta(S))$

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

be the *weighted degree* of S. This definition is keeping with the idea of thinking each edge with $x_e \geq 1/2$ as two parallel edges. Observe that for any $v \notin W$, v receives exactly $\mathrm{wdeg}(v)$ tokens in the initial assignment as it gets one token for each edge and two tokens for all heavy edges incident on 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 $x_{e'} = x_{e''} = x_e/2$. Observe that

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

and $\mathrm{wdeg}(\delta(S)) = |\delta'(S)|$ 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 Jain [9] 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 $1/2$. Then the co-requirement of S is $1/2$.*

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 [22] can be used to show that $|\delta'(S)|$ is odd. Hence, the co-requirement of S is semi-integral. Now, we show that coreq$(S) < 3/2$, proving the claim. Clearly,

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

where the first sum is over all children R of S and the 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 $1/2$ and in the second sum is strictly less than $1/2$, if $\alpha > 0$ then we have coreq$(S) < 3/2$ which proves the claim. So, assume $\alpha = 0$, i.e. S does not own any token. 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. Therefore, we have coreq$(S) < \sum_R \text{coreq}(R) = 3/2$, 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 in the initial token assignment. 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.

Therefore, for the rest of the cases, we assume that S does not own any vertex of W. We note that the remaining case analysis is very similar to that of Jain, with the refined 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. Otherwise, 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 $1/2$ and it only needs three tokens.

- S has exactly two children R_1 and R_2. If both of them have two excess tokens from the induction hypothesis then we are done. Otherwise, let R_1 have exactly one token and hence it has co-requirement of $1/2$ by the induction hypothesis. We claim that S must own an endpoint. Suppose to the contrary that S does not own any endpoint. Then, if there are α edges 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, $\chi_{\delta'(S)} \neq \chi_{\delta'(R_1)} + \chi_{\delta'(R_2)}$, there must be an edge between R_1 and R_2 and therefore $\text{coreq}(S) < \text{coreq}(R_2) + 1/2$. 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 $\text{coreq}(R_2) < \text{coreq}(S) + 1/2$. 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 a 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 $1/2$. Then, by Lemma 3.5, S has a co-requirement of $1/2$ and only needs three tokens.

This completes the proof of the induction step. □

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. There is at least one extra token in any root of $\mathcal{L}$. 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.

3.3 Integrality Gap Example

In Figure 13 we show that the linear program (LP) has an integrality gap of $B + \Omega(r_{\max})$.

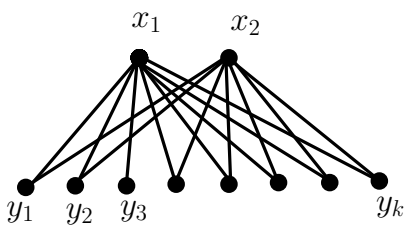

Figure 13: An integrality example of (LP).

In this example, we have a complete bipartite graph $(X, Y; E)$ 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 $k/2$, and 0 otherwise. The fractional solution where all edges have fractional value $1/2$ is the optimal solution, in which the degree of x_1 and x_2 is equal to $k/2 = \Delta_f^*$. 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 $(2, b_v + \frac{1}{2}r_{\max})$.

Concluding Remarks

An interesting open question is to close the gap for the minimum bounded degree Steiner forest problem, i.e. to determine whether the integrality gap of the linear programming relaxation is $B_v + 2$ or $B_v + 3$. Another interesting open question is to determine whether it is hard to approximate the minimum bounded degree Steiner network problem with degree violation $o(r_{\max})$.

The iterative relaxation method has been successfully applied to network design problems with degree constraints [14, 21, 2], and recently it has also been applied to other combinatorial optimization problems and also give simple proofs of existing results [10]. We hope that this method will find further applications.

Acknowledgement

We are very grateful to two anonymous reviewers for giving many useful suggestions that greatly improved the presentation of this paper.

References

[1] A. Agrawal, P. Klein and R. Ravi, *When trees collide: an approximation algorithm for the generalized Steiner problem on networks*, Proceedings of the twenty-third annual ACM symposium on Theory of computing, pages: 134–144, 1991.

[2] N. Bansal, R. Khandekar and V. Nagarajan, *Additive guarantees for degree bounded directed network design*, SIAM Journal on Computing, 29, 1413-1431, 2009.

[3] F. Bauer and A. Varma, *Degree-constrained multicasting in point-to-point networks*, 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 APPROX 2005, pp. 26-39.

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

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

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

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

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

[10] T. Király, L.C. Lau, M. Singh, *Degree Bounded Matroids and Submodular Flows*, Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 259-272, 2008.

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

[12] 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.

[13] 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.

[14] L.C. Lau, S. Naor, M. Salavatipour and M. Singh, *Survivable network design with degree or order constraints*, SIAM Journal on Computing 39(3), 1062–1087, 2009.

[15] L.C. Lau, R. Ravi, M. Singh, *Iterative Methods in Combinatorial Optimization*, Cambridge University Press, 2011.

[16] A. Louis, N.K. Vishnoi, *Improved algorithm for degree bounded survivable network design problem*, in Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory, 408-419, 2010.

[17] Z. Nutov, *Degree-constrained node-connectivity*, in Proceedings of the 10th Latin American Symposium on Theoretical Informatics (LATIN), 582–593, 2012.

[18] B. Raghavachari, *Algorithms for Finding Low Degree Structures*, in Approximation algorithms, Dorit Hochbaum (ed.), PWS Publishers Inc., pages 266-295, 1996.

[19] 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.

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

[21] 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, 2007.

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