Iterative Rounding Approximation Algorithms in Network Design

by

Marcus Shea

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Iterative rounding has been an increasingly popular approach to solving network design optimization problems ever since Jain introduced the concept in his revolutionary 2-approximation for the Survivable Network Design Problem (SNDP) [33]. This paper looks at several important iterative rounding approximation algorithms and makes improvements to some of their proofs. We generalize a matrix restatement of Nagarajan et al.'s token argument from [42] which we can use to simplify the proofs of Jain's 2-approximation for SNDP and Fleischer et al.'s 2-approximation for the Element Connectivity (ELC) problem. Lau et al. [38] show how one can construct a $(2, 2B + 3)$-approximation for the degree bounded ELC problem, and this thesis provides the proof. We provide some structural results for basic feasible solutions of the Prize-Collecting Steiner Tree problem, and introduce a new problem that arises, which we call the Prize-Collecting Generalized Steiner Tree problem.

Acknowledgements

I would like to thank my readers Chaitanya Swamy, Nicholas Harvey and Jochen Könemann for taking the time to read and provide feedback on this thesis. I would particularly like to thank Jochen Könemann for providing insightful discussion and support throughout the course of my Masters program.

Dedication

Long live the LCC.

Contents

List of Figures

Chapter 1

Introduction

The area of network design looks at designing graphs that satisfy certain desirable properties. Most network design problems arise from settings in the real world. The Steiner Tree problem, for example, has many applications including the problem of designing a network that most efficiently connects a collection of computers. Many of these network design problems are NP-hard, meaning that there are no known polynomial time algorithms to solve them. We are therefore interested in finding efficient algorithms that can find near-optimal solutions to these problems. One method of designing these approximation algorithms is called iterative rounding. This thesis will look at how iterative rounding can be used to construct approximation algorithms for several important NP-hard network design problems.

Iterative rounding has been an increasingly popular approach to solving network design optimization problems ever since Jain introduced the concept in his revolutionary 2-approximation for the Survivable Network Design Problem [33]. The idea is to model the network design problem as an integer program, and then iteratively use the linear programming relaxation to come up with an integer solution. The linear programming relaxation is solved, and then any variables above some certain threshold, say $\frac{1}{s}$, are rounded up. Provided that the removal of these variables maintains a similar linear program, and that we can prove that there is always a variable achieving this threshold value, this method allows one to obtain an integer solution of cost at most s times the optimal linear programming objective value. The iterative rounding approach has been used in both undirected network design [9, 18], as well as in directed network design [23, 40].

Every iterative rounding algorithm that we will examine in this thesis requires some form of uncrossing argument. This argument is used to obtain structure on a basic feasible solution x of the linear program (LP) being solved, and such an argument is frequently used in approximation literature [13, 14, 20]. The argument will give us a basis of particularly nice structure for the basic feasible solution x. Then, by assuming that every value of x is below a certain threshold, we can use a counting argument on our basis to derive a contradiction. There are two common counting arguments, ones based on counting endpoints of edges, and another based on counting what we call fractional tokens.

Endpoint based arguments, again first introduced by Jain [33], involve assigning the endpoints of every edge to appropriate constraints in our well structured basis, and then recounting them to show that if every edge has value below some threshold, we end up with more endpoints than we started with. The method of reassigning endpoints varies depending on the problem considered. This style of argument is used in the proof of Fleischer et al.'s 2-approximation for element connectivity [17] as well as in Lau et al.'s bicriteria approximation algorithms for degree bounded network design [38, 39]. One key attribute of such an argument is that endpoints are discrete - they are not to be split into fractions.

Instead of an endpoint based argument, some iterative rounding approximation algorithms are proven using a fractional token argument. First seen in a bicriteria approximation algorithm by Bansal et al. [5], fractional token arguments work by giving each variable a number of tokens which it can redistribute fractionally. As in the endpoint based argument, we then recount our number of tokens to show that if every edge is below some threshold, we end up with more tokens than when we started. The benefit of a fractional token based argument is that the analysis can be considerably more straight forward. This is demonstrated by Nagarajan et al. in [42] where the authors are able to greatly simplify both the proof of Jain's 2-approximation for Survivable Network Design [33], as well as the proof of Boyd and Pulleyblank's result that there is a 1-edge in any basic feasible solution to the LP defining the Symmetric Traveling Salesman problem [7].

In this thesis, we will look at various iterative rounding approximation algorithms for various approximation algorithms, making use of both endpoint and fractional token arguments, depending on the scenario. We now begin by describing the network design problems that we will be looking at in this thesis. We then mention our contributions that will be shown. We will review the work that has taken place in this area, and then we

will review the fundamental example of iterative rounding, Jain's 2-approximation for the Survivable Network Design problem.

1.1 Problems

Here we define the problems that we will be looking at in this thesis. In this thesis we will only be considering minimization problems. Consider a graph $G = (V, E)$ with costs c on edges.

Consider a special root node, and a set of terminal vertices. The (minimum cost) Steiner Tree problem (ST) is to find a minimum cost subgraph of G in which every terminal vertex is connected to the root. As mentioned earlier, this problem has many practical applications, such as the design of optical and wireless communication systems, as well as transportation and distribution networks [31]. This problem was one of the first few problems proven to be NP-complete by Karp [34] Because of its practical and long history, the Steiner Tree problem is considered a fundamental network design problem.

A common generalization of the Steiner Tree problem is the Steiner Forest problem (SF). In this setting we have terminal pairs instead of single vertices. The (minimum cost) Steiner Forest problem is to find a minimum cost subgraph that contains a path between every terminal pair. This problem has many applications as well, including the design of shipping networks between major cities [29]. Because of the fact that we only have connection requirements between pairs of vertices, the resulting solution may have multiple components, hence the name of the problem.

A further generalization is the Survivable Network Design problem. In this scenario, we allow integer connectivity requirements $r_{i,j}$ between pairs of vertices i, j. The Survivable Network Design problem is to find a minimum cost subgraph of G in which there are $r_{i,j}$ i, j-paths. In the edge connectivity version of the problem (SNDP), we wish for the paths to be edge-disjoint, and in the vertex connectivity version of the problem (VCSNDP) we would like them to be vertex-disjoint. A problem of intermediate difficulty is the Element Connectivity problem (ELC). It is defined by partitioning our vertices into terminals and non-terminals, and enforcing that connectivity requirements are only nonzero for pairs of terminals. Further, we define an *element* to be an edge or a non-terminal, and in this problem we require our paths to be element-disjoint. These

problems have applications in designing computer networks where we want computers i and j to be connected, even if $r_{i,j}$ connections fail [17], where the connections in question are determined by the variant of the problem chosen.

As these are all NP-hard problems, we should not expect to be able to find optimal solutions efficiently. We therefore are concerned with finding solutions that are provably close to optimal, in polynomial time.

1.1.1 Definition. *We say that $\mathcal{A}$ is an* α-approximation algorithm *for problem Γ if for any instance γ of Γ, $\mathcal{A}$ returns a feasible solution x to γ of objective value no more than α times the optimal objective value for γ.*

We also study *prize-collecting* variants of some these problems. Here we consider the scenario where we do not need to satisfy all connectivity requirements, but we are given a *penalty function* π for violating the requirements. The goal is to minimize the cost of the subgraph chosen plus the penalties paid for all connectivity requirements that are not satisfied. For example, in the Prize Collecting Steiner Tree (PC-ST), each terminal $v \in R$ has a penalty $\pi(v)$, and the cost of a chosen subgraph H is $\sum_{e \in E(H)} c(e) + \sum_{v \in R} \pi(v) z(v)$ where $z(v) = 0$ if v is connected to the root, and $z(v) = 1$ otherwise. We will refer to the prize-collecting versions of the other problems similarly.

We will also be looking at the *degree bounded* variants of these problems. In this case, we allow non-negative integral lower and upper degree bounds, L_v and B_v respectively, on any vertex $v \in V$. In the subgraph H chosen, we wish to have $L_v \leq |\delta_H(v)| \leq B_v$. We denote the Degree Bounded Steiner Tree problem by ST-B, and will refer to the degree bounded variants of the other problems similarly. In this case, we have *bicriteria* approximation algorithms, as we wish for solutions with both low cost, and minimal degree bound violations.

1.1.2 Definition. *We say that $\mathcal{A}$ is an* $(\alpha, \beta(B))$-approximation algorithm *for degree bounded problem Γ if for any instance γ of Γ, $\mathcal{A}$ returns a feasible solution x to γ of objective value no more than α times the optimal objective value for γ, and furthermore, the degree of any vertex v in our solution is $\geq L_v$, but no more than $\beta(B_v)$.*

1.2 Our Contributions and Results

In Chapter 3, we generalize a matrix restatement by Swamy [50] of Nagarajan et al.'s token argument from [42]. This allows one to easily show that a basic feasible solution for the LP of a given instance of a network design problem has an edge of value above a certain threshold, provided that it satisfies constraints of a certain form. We demonstrate how this fractional token argument simplifies the proof showing the existence of $\frac{1}{2}$-edge in any basic feasible solution to the standard LP for SNDP, in the same manner in which this is done in [42].

In Chapter 4, we present Fleischer et al.'s 2-approximation for ELC [17], and give a much simplified proof using a generalized fractional token argument, very similar to that of Nagarajan et al. [42]. We then present a $(2, 2B + 3)$-approximation for ELC-B, with degree bounds only allowed on terminal vertices. This algorithm is a generalization of a $(2, 2B + 3)$-approximation for the SNDP-B given by Lau et al. [38]. The authors claim that their algorithm can be generalized to one for ELC-B, and this thesis provides the proof.

In Chapter 5, we present Grandoni et al.'s 3-approximation for the PC-SF problem [26], with a simplified proof. We then provide some insights toward achieving a Lagrangian multiplier preserving 2-approximation for PC-ST. We show that performing certain operations on a basic feasible solution of the natural LP for this problem can result in a similar problem that we call the Prize Collecting Rooted Steiner Forest problem. We show that a fractional token argument will not be able to prove a $\frac{1}{2}$-approximation for this problem, and provide some insights into using an integer decomposition technique instead.

1.3 Related Work

The Steiner Tree problem is a classical problem in computer science research, which can be traced back as far as Fermat [31]. There have been many improving algorithms throughout the literature [51, 43, 35, 46, 45], eventually culminating with the current best algorithm of a $\ln(4) + \epsilon < 1.39$-approximation by Byrka et al. [8]. Their algorithm makes use of a novel iterative randomized rounding technique.

An early algorithm for the Steiner Forest problem by Agrawal et al. [1] gave a

$(2-\frac{2}{s})$-approximation for the problem, where s is the number of distinct terminal vertices. Later algorithms simplified the analysis and generalized the method [25], but no algorithms have been found with better approximation ratios.

Consider a Survivable Network Design problem on graph $G = (V, E)$ with costs c and connectivity requirements r_{ij} for pairs of vertices i, j. Let k be the maximum connectivity requirement. The edge-connectivity version of the problem (SNDP) was originally proposed by Agrawal et al. [1], and a $(2 - \frac{2}{s})\lceil \log_2(r_{max} + 1)\rceil$-approximation for SNDP was provided. In this case, r_{max} is the highest connectivity requirement, s is the number of vertices with some connectivity requirement, and the authors assume that you can use an edge multiple times. The current best known approximation algorithm for SNDP is a 2-factor iterative rounding algorithm provided by Jain [33], where edges are not repeated. We will review this algorithm at the end of this chapter.

The ELC problem was originally proposed by Jain et al. [32]. It can realistically model the problem of forming computer networks, as in such a scenario both links (edges) and routers (nonterminals) can fail, but typically network terminals (end hosts) are more robust. In Jain's original paper on the problem, he provides a $2H_k$-approximation, where $H_k = 1 + \frac{1}{2} + ... + \frac{1}{n}$. Fleischer et al. [17] generalized Jain's 2-approximation for SNDP to achieve a current best 2-approximation for the ELC problem. Cheriyan et al. [9] subsequently prove a similar but more general algorithm that gives a 2-approximation for ELC as well.

A general approximation algorithm for the vertex connectivity version of the problem, VCSNDP, has only recently been discovered. Chuzhoy et al. [12] present the first general approximation algorithm for this problem, obtaining a $O(k^3 \log n)$-approximation. Their algorithm achieves its goal through solving multiple carefully designed instances of the Element Connectivity Problem, ELC, a problem of intermediate difficulty to SNDP and VCSNDP.

1.3.1 Degree Constrained Problems

Network design subject to degree constraints has become an area of greater interest recently. A simpler setting is to attempt to minimize the maximum degree of a subgraph satisfying some network requirements. A well-known example is the Minimum Degree Spanning Tree problem, for which Fürer and Raghavachari provide a combinatorial algorithm that

returns a tree of maximum degree at most 1 more than the optimal tree, in the unweighted case [21]. This result is generalized to the weighted case by Singh and Lau [49], and to the case of unweighted Steiner trees by Fürer and Raghavachari [22].

A more general setting is one where we desire a minimum cost subgraph satisfying connectivity requirements in addition to degree bounds B_v for vertices $v \in V$. The first algorithms for such problems were $O(\log(n), \log(n) \cdot B)$-approximations by Ravi et al. [44] for each of ST-B, SF-B and SNDP-B. Both results have been significantly improved; in [38], Lau et al. provide a $(2, 2B + 3)$-approximation for SNDP-B and note that it can be extended to a $(2, 2B + 3)$-approximation algorithm for ELC-B, the proof of which is in Section 4.2. These results are further improved by Lau and Singh in [39], a current best $(2, B+3)$-approximation for SF-Band a $(2, B+6k+3)$-approximation for SNDP-B, where k is the maximum connectivity requirement. The authors claim that this result extends to the element connectivity setting as well. Note that the degree error in this case is only additive and not multiplicative.

1.3.2 Prize-Collecting Problems

The PC-ST problem was first proposed by Balas [3], but the first approximation algorithm came from Bienstock et al. [6]. The authors provided a 3-approximation. Goemans and Williamson improved this factor with a 2-approximate primal dual algorithm [25]. This was eventually improved to the current best 1.992-approximation due to Archer et al. [2].

The PC-SF problem was first formulated by Hajiaghayi and Jain [29] who provided a 3-approximate primal-dual algorithm. They also showed a 2.54-approximation for the problem that makes use of randomized LP rounding. The latter result was then shown for more general connectivity requirements and penalty functions by Sharma et al. [48]. Grandoni et al. [26] provided a 3-approximate iterative rounding algorithm, and Hajiaghayi et al. [30] provided an iterative rounding algorithm with the same approximation ratio, but extended to the more general PC-SNDP setting. In a recent paper by Hajiaghayi et al. [28] generalized the results by Sharma et al. [48]to include all-or-nothing penalty functions, giving 2.54-approximations for PC-SNDP and PC-ELC, and $O(k^3 \log n)$-approximation for PC-VCSNDP. By all-or-nothing, we mean that the full penalty is charged even if connectivity requirements are only slightly violated.

1.4 Jain's 2-approximation for SNDP

Jain's 2-approximation for the Survivable Network Design Problem [33] is a classic example of an iterative rounding approximation algorithm. This algorithm inspired many iterative rounding algorithms to follow, such as Fleischer et al.'s 2-approximation for the Element Connectivity Problem. We start this thesis with a review of this fundamental algorithm, as it motivates many of the later algorithms that we will encounter.

Consider a graph $G = (V, E)$ with edge costs $c \in {\mathbb{Q}_+}^E$ and symmetric connectivity requirements $r_{i,j} \in \mathbb{Z}^{V \times V}$ for any pair of vertices $i, j \in V$. The goal is to find a minimum cost subgraph H of G such that between any pair of vertices i, j we have at least $r_{i,j}$ edge-disjoint paths in H.

If we define a function $f : 2^V \to \mathbb{Z}$ by $f(A) = \max\{r_{i,j} : i \in A, j \notin A\}$ for all $\emptyset \neq A \subset V$, and set $f(V) = f(\emptyset) = 0$, then our problem is equivalent to solving the following integer program (IP):

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) && \text{(IP-}f\text{)} \\ \text{s.t.} \quad & x(\delta(A)) \geq f(\delta(A)) && \forall A \subseteq V \\ & x(e) \in \{0, 1\} && \forall e \in E \end{aligned}$$

We relax the last constraints to $0 \leq x(e) \leq 1$ for all $e \in E$ to get the linear relaxation (LP-f). Jain's algorithm is actually a 2-approximation for (IP-f) for any *weakly supermodular* function f, provided that there is a separation oracle for the linear relaxation of the respective integer program.

It is easy to see that the function f defined for the SNDP instance is a *proper function*. That is, $f(V) = 0$ and for any sets A and B, $f(A) = f(V - A)$ and $f(A \cup B) \leq \max\{f(A), f(B)\}$. Jain shows that this implies that f is *weakly supermodular* (see Lemma 2.4.5).

The algorithm is based on the fact that any basic solution to (LP-f) has an edge of value at least $\frac{1}{2}$. We can round up these edges, for which we pay at most an additional half of edge cost, and then solve a smaller residual problem in the same manner. The algorithm thus relies on the following theorem.

1.4.1 Theorem ([33])**.** *Let f be weakly supermodular and $f \neq 0$. In any basic solution x to (LP-f), there is an edge e where $x(e) \geq \frac{1}{2}$.*

We will provide a proof of this theorem in Chapter 3 using a fractional token argument. First, we present Jain's 2-approximation algorithm, Algorithm 1.

Algorithm 1 Iterative Rounding 2-Approximation for SNDP

1: $F \leftarrow \emptyset$
2: $f' \leftarrow f$
3: **while** F is not feasible for (IP-f) **do**
4: Solve (LP-f'), let x be an optimal basic solution
5: For any edge e where $x(e) \geq \frac{1}{2}$, add e to F, remove e from E
6: $f'(A) \leftarrow f(A) - |\delta_F(A)|$ for every $A \subset V$.
7: **end while**

It is well known that the cut function, $|\delta_G(\cdot)|$ is from the class of *submodular* functions. As the difference between a weakly supermodular function and a submodular function is weakly supermodular, we get that f' is weakly supermodular at every stage of our algorithm, justifying the recursive application of this algorithm. These facts are verified in Section 2.4.

One should argue that (LP-f) can be solved efficiently. However, this is not our focus for this algorithm, and refer the interested reader to Jain's paper [33] where he deals with these details.

Assuming that the key theorem holds, we justify that the output solution has cost within a factor of 2 of optimal. To do so, Jain provides an inductive proof which is common to many of the iterative rounding algorithms that followed.

We assume that the algorithm provides a 2-approximate solution F_{res} to the residual problem, and then want to show that we can build this into a 2-approximate solution for the given instance. Let x^* be the optimal solution to (LP-f), and let F be all edges with value $x^*(e) \geq \frac{1}{2}$. Let OPT and OPT_{res} be the optimal solutions to the given instance and residual instance respectively. By definition,

$$\sum_{e \in F_{res}} c(e) \leq 2 \cdot \text{OPT}_{res}.$$

By the definition of the residual problem (LP-f'), we know that $F \cup F_{res}$ is feasible. Furthermore, we note that x^* restricted to the residual graph is a feasible solution for the

residual problem, and thus

$$\mathrm{OPT}_{res} \leq \sum_{e \in E-F} c(e)x^*(e).$$

Now the cost of $F \cup F_{res}$ is

$$\begin{aligned}
\sum_{e \in F_{res}} c(e) + \sum_{e \in F} c(e) &\leq 2 \cdot \mathrm{OPT}_{res} + \sum_{e \in F} c(e) \\
&\leq 2 \cdot \mathrm{OPT}_{res} + \sum_{e \in F} 2 \cdot x^*(e)c(e) \text{ since } x^*(e) \geq \frac{1}{2} \text{ for } e \in F \\
&\leq 2 \cdot \sum_{e \in E-F} c(e)x^*(e) + 2 \cdot \sum_{e \in F} c(e)x^*(e) \\
&= 2 \cdot \mathrm{OPT} \text{ as } x^* \text{ is an optimal solution for (LP-}f\text{)}
\end{aligned}$$

All that remains now is to show the key theorem. Jain's original argument would obtain a laminar basis $\mathcal{B}$ for the a basic solution x to (LP-f), and proceed with an endpoint counting argument, eventually showing that we have more edges in the support of x than sets in $\mathcal{B}$. This is a contradiction, as any basis for x has rank equal to the support of x.

Nagarajan et al. [42] construct a much simpler proof using a fractional token argument, but the general idea remains the same. Their paper also show how a fractional token argument can be used to prove the existence of an edge of value one in any extreme point of the natural LP relaxation for the traveling salesman problem. This allows for a simpler proof of Boyd and Pulleyblank's approximation algorithm for the traveling salesman problem.

In this thesis we extend the fractional token argument presented by Nagarajan et al. to a more complex system of constraints. In Chapter 3 we present this generalized fractional token argument. In Section 3.1 we show how this fractional token argument allows us to prove Theorem 1.4.1, completing the proof of Jain's 2-approximation for SNDP.

Chapter 2

Basics

In this chapter, we go over all the basic material needed for the rest of the thesis. The first section deals with some basic graph theory and notation. This is followed by a section with some standard results from linear programming theory. In section 3 we look at what is meant by a *laminar family*, and show how such a family induces a forest. In section 4 we look at proper, submodular, and weakly supermodular functions. We also prove an uncrossing argument for weakly supermodular functions, which is used in the proof of Jain's 2-approximation for SNDP. Chapters 5 and 6 generalize the definitions and results found in chapters 3 and 4 to functions that act on pairs of subsets of vertices. These results are used in Fleischer's 2-approximation for ELC, as well as in the $(2, 2B + 3)$-approximation for ELC-B shown in section 4.2.

2.1 Graph Basics and Notation

We define a *graph* $G = (V, E)$ to be a pair consisting of a vertex set V and an edge set $E \subseteq \{\{u, v\} : u, v \in V\}$. We call a $v \in V$ a *vertex* and a $e = \{u, v\} \in E$ an *edge*. For convenience, we may write $e = uv$. We define $E(G) = E$ and $V(G) = V$.

We define the *cut* for vertex set $A \subseteq V$ in graph G to be $\delta_G(A) = \{uv \in E : u \in A, v \notin A\}$. We will often write $\delta(A)$ when it is clear over which graph we are taking the cut. A cut defined for a disjoint pair of vertex sets (A, A') is given by $\delta_G(A, A') = \{uv \in E : u \in A, v \in A'\}$. For a subset of edges $F \subseteq E$, we define a cut over this subset by $\delta_F(A) = \delta_G(A) \cap F$. Similarly, we define $\delta_F(A, A') = \delta(A, A') \cap F$.

Define the characteristic vector of a set of edges $F \subseteq E$ to be a vector $[F] \in \{0,1\}^E$ where $[F](e) = 1$ if and only if $e \in F$. For the cut $\delta(A), A \subseteq V$ we write $[A] = [\delta(A)]$ and for a cut $\delta(A, A')$ over a disjoint pair of vertex subsets, we write $[A, A'] = [\delta(A, A')]$. For any family of sets or set pairs, $\mathcal{F}$, we define $\textsc{Span}(\mathcal{F})$ to be the vector space spanned by the characteristic vectors of the cuts of the members of $\mathcal{F}$.

For any vector or function x indexed over the set of edges E, we define for any subset $F \subseteq E$, $x(F) = \sum_{e \in F} x(e)$. Similarly, for any vertex subset $A \subseteq V$, define $x(A) = x(\delta(A))$ and for any disjoint pair of vertex subsets (A, A'), define $x(A, A') = x(\delta(A, A'))$.

A *directed graph* $D = (N, A)$ is defined as a pair consisting of a node set N and an arc set $A \subseteq N \times N$. A $n \in N$ is called a *node* and $a = (u, v) \in A$ is called an *arc* with *tail* u and *head* v. Let c be a vector of nonnegative capacities on edges.

A vector x defined on arcs A is a feasible *st-flow* for $s, t \in V$ if

- $\sum_{(u,v) \in A} x(u, v) = \sum_{(v,u) \in A} x(v, u)$ for every $v \in V - \{s, t\}$, and
- $x(u, v) \leq c(u, v)$ for each $(u, v) \in A$.

The value of an *st*-flow x is given by $(\sum_{(s,v) \in A} x(s, v) - \sum_{(v,s) \in A} x(v, s))$.

An *st-cut* A is a set $A \subset V$ such that $s \in A$, $t \notin A$. The capacity of such an *st*-cut is $c(A) = c(\delta(A))$.

The following theorem by Ford and Fulkerson [19] is fundamental to the construction of separation oracles for several algorithms considered in this thesis.

2.1.1 Theorem ([19])**.** *The maximum value of an st-flow in G is equal to the minimum capacity of an st-cut.*

Another fundamental theorem in graph theory is Menger's Theorem [41], which we will use in the formulation of some linear programs throughout this thesis.

2.1.2 Theorem (Menger's Theorem [41])**.** *Let s and t be non-adjacent vertices in a graph G. Then the minimum number of vertices separating s from t in G is equal to the maximum number of vertex disjoint paths from s to t in G.*

2.2 Linear Programming Basics

A *linear programming* (LP) problem is a problem of optimizing some linear function $c^T x$ over a finite number n of real variables x, subject to a finite number of linear equations and inequalities $Ax \leq b$. Note that we may write a linear equation as two linear inequalities, justifying the form we use here. Further, we may assume $x \geq 0$, as any free variable can be replaced by the difference of two nonnegative variables. In this thesis, we will only consider minimization problems.

If x satisfies $Ax \geq b$ and $x \geq 0$ then we say x is *feasible*. If some feasible solution x exists for an LP, then we say the LP is *feasible*, and otherwise, it is *infeasible*. We call the polytope $P = \{x : Ax \geq b, x \geq 0\}$ the *feasible region* for our LP. We say that a feasible solution x^* is *optimal* if $c^T x^* = \min\{c^T x : x \in P\}$.

2.2.1 Basic Solutions

Consider a polyhedron $P = \{Ax \geq b\} \subseteq \mathbb{R}^n$. Let a_i denote row i of A. Let c be a nonzero vector and $\beta = \min\{c^T x : Ax \geq b\}$. The affine hyperplane $H = \{x : c^T x = \beta\}$ is called a *supporting hyperplane* of P. A subset F of P is called a *face* if $F = P \cap H$ for some supporting hyperplane H.

Schrijver shows that the following is an equivalent characterization of a face:

2.2.1 Lemma ([47])**.** *F is a face of P if and only if $F = \{x \in F : A'x = b'\}$ for some subsystem $A'x \geq b'$ of $Ax \geq b$.*

A *minimal face* is an inclusion-wise minimal face of P. A *basic solution* (also called *vertex* in some sources) x is an element $x \in P$ such that $\{x\}$ is a minimal face.

2.2.2 Theorem ([47])**.** *If $x \geq 0$ for every $x \in P$, then every minimal face F is a set with a single basic solution.*

Proof. Suppose F is not a single basic solution. Let $\mathcal{E}$ be the set of inequalities which are tight for each $x \in F$. Since x^* is not a basic solution, the affine subspace defined by the equalities in $\mathcal{E}$ is at least one dimensional. Hence, it contains a line, L, passing through x^*. Because of constraints $x \geq 0$, the entire line L is not contained within P - there is

some point $\overline{x}$ on L that is on the boundary of P. But then adding this constraint to the subsystem that defines F (see Lemma 2.2.1) defines a smaller face, contradicting the definition of F. □

We say that a constraint $a_i^T x \geq b_i$ is *tight* for a solution x if $a_i^T x = b_i$. The following theorem is also proven by Schrijver, and will be used in the proofs of several approximation algorithms in this thesis.

2.2.3 Theorem ([47])**.** *A solution $x \in P$ is a basic solution if and only if the rank of all tight constraints for x is n.*

As many of our approximation algorithms will center on solving LPs, we must show that we can solve our LPs in polynomial time. To do so, we will require a *separation oracle* for our LP. A separation oracle is a polynomial time algorithm that, given a point x, can either confirm that x is a feasible solution, or provide an inequality that is satisfied by every feasible solution, but is not satisfied by x. Given a separation oracle, we then invoke the following theorem to solve our LP in polynomial time.

2.2.4 Theorem ([27])**.** *Given a separation oracle for an LP, the LP can be solved in polynomial time using the Ellipsoid Method.*

However, we often need an optimal basic solution. A standard result from linear programming theory (see [10] for example) is that we can find an optimal basic solution in polynomial time, given any optimal solution. Hence we get the corollary that we desire.

2.2.5 Corollary. *Given a separation oracle for an LP, we can find an optimal basic solution to the LP in polynomial time.*

2.3 Laminar Families

Laminar families of sets have a nice structure in the sense that the family can be easily described by a unique forest. In many iterative rounding approximations we use a laminar basis for a basic feasible solution to prove the existence of an edge of value above some threshold. In network design, laminar families are typically found on the set of vertices V of a graph.

2.3.1 Definition. *We say that two sets A and B* cross *if none of the sets $A - B$, $B - A$ and $B \cap A$ is empty.*

2.3.2 Definition. *We say that a family of sets is* laminar *if no two sets in it cross.*

2.3.3 Lemma ([33])**.** *Let $\mathcal{L}$ be a laminar family, and let $L \in \mathcal{L}$. Let A be some set that crosses L. Then if $L' \in \mathcal{L}$ crosses any set listed below, L' also crosses A.*

1. $(A - L)$
2. $(L - A)$
3. $(A \cap L)$
4. $(A \cup L)$

Proof. Case 1: L' crosses $A - L$.

If $L \cap L' = \emptyset$, it is clear that L' must cross A. If $L' \subseteq L$, then $(A - L) \cap L' = \emptyset$, contradicting that L' crosses $A - L$. Thus $L \subseteq L'$. Thus $\exists u \in L - A \subseteq L' - A$, $\exists v \in (A - L) - L' = A - L'$, $\exists w \in A \cap L \subseteq A \cap L'$, showing that L' crosses A.

Case 2: L' crosses $L - A$.

If $L \subseteq L'$, then $L' - (L - A) = \emptyset$, contradicting that L' crosses $L - A$. Thus $L' \subseteq L$. Thus $\exists u \in L' \cap (L - A) \subseteq L' - A$, $\exists v \in A - L \subseteq A - L'$, $\exists w \in L' - (L - A) \subseteq L' \cap A$, showing that L' crosses A.

Case 3: L' crosses $(A \cap L)$.

If $L \subseteq L'$, then $L' - A \cap L = \emptyset$, contradicting that L' crosses $A \cap L$. Thus $L' \subseteq L$. Thus $\exists u \in L' - (L \cap A) \subseteq L' - A$, $\exists v \in A - L \subseteq A - L'$, $\exists w \in L' \cap (L \cap A) \subseteq L' \cap A$, showing that L' crosses A.

Case 4: L' crosses $(A \cup L)$.

If $L \cap L' = \emptyset$, it is clear that L' must cross A. If $L' \subseteq L$, then $L' - (A \cup L) = \emptyset$, contradicting that L' crosses $A \cup L$. Thus $L \subseteq L'$. Thus $\exists u \in L' - (A \cup L) \subseteq L' - A$, $\exists v \in (A \cup L) - L' \subseteq A - L'$, $\exists w \in L \cap A \subseteq L' \cap A$, showing that L' crosses A.

□

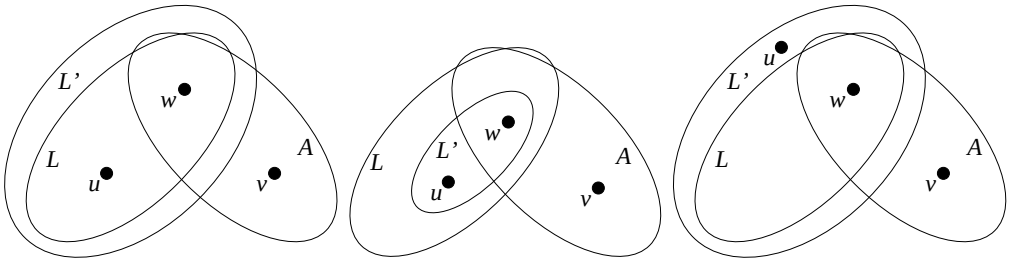

Figure 2.1: An illustration of our cases in Lemma 2.3.3. Left: Case 1, Centre: Cases 2 and 3, Right: Case 4.

2.3.1 Forest Induced by a Laminar Family

Any laminar family naturally induces a forest structure on its sets. To differentiate from the original graph, it is convenient to refer to vertices and edges in the original graph, and nodes and arcs in our forest.

2.3.4 Lemma. *Any laminar family of sets $\mathcal{L}$ induces a unique forest $F_{\mathcal{L}}$ where each $A \in \mathcal{L}$ is a node, and there is an arc between A to B if and only if $B \subset A$, and for any $C \in \mathcal{L} - \{B\}$ with $C \subset A$, we have $C \subseteq B$.*

Proof. As $\mathcal{L}$ is laminar, containment is a partial ordering, and as such every node has at most one parent. □

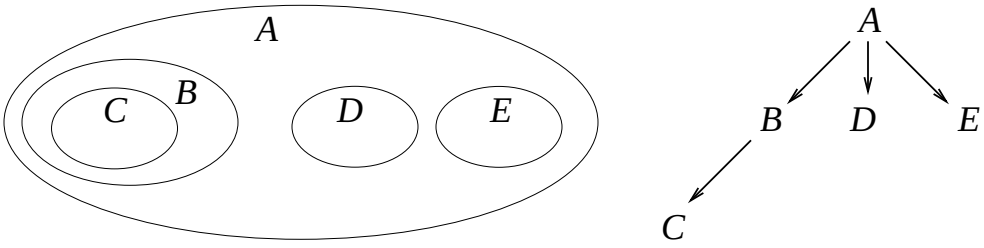

Figure 2.2: A laminar family $\mathcal{L}$ and its associated forest $F_{\mathcal{L}}$.

2.4 Proper, Submodular, and Weakly Supermodular Functions

In this section we define several important classes of functions, and prove an uncrossing argument for *weakly supermodular* functions. The uncrossing argument is necessary for the proof of Jain's 2-approximation for SNDP [33].

2.4.1 Definition. *A function* $f : 2^V \to \mathbb{Z}$ *is* submodular *if* $f(V) = 0$ *and for every* $A, B \subseteq V$, $f(A) + f(B) \geq \max\{f(A - B) + f(B - A), f(A \cap B) + f(A \cup B)\}$.

The above definition of *submodularity* is slightly non-standard, but is commonly used in literature on SNDP.

2.4.2 Lemma ([33])**.** *For any graph* G*, the function* $|\delta_G(\cdot)|$ *is submodular.*

Proof. Let $A, B \subseteq V$. We partition our vertices into four sets, $S_1 = A - B$, $S_2 = B - A$, $S_3 = A \cap B$, and $S_4 = V - (A \cup B)$. Note that the following equalities hold (see Figure 2.3):

1. $|\delta_G(A)| = |\delta(S_1, S_2)| + |\delta(S_1, S_4)| + |\delta(S_2, S_3)| + |\delta(S_3, S_4)|$
2. $|\delta_G(B)| = |\delta(S_1, S_2)| + |\delta(S_2, S_4)| + |\delta(S_1, S_3)| + |\delta(S_3, S_4)|$
3. $|\delta_G(A - B)| = |\delta(S_1, S_2)| + |\delta(S_1, S_3)| + |\delta(S_1, S_4)|$
4. $|\delta_G(B - A)| = |\delta(S_1, S_2)| + |\delta(S_2, S_3)| + |\delta(S_2, S_4)|$
5. $|\delta_G(A \cap B)| = |\delta(S_1, S_3)| + |\delta(S_2, S_3)| + |\delta(S_3, S_4)|$
6. $|\delta_G(A \cup B)| = |\delta(S_1, S_4)| + |\delta(S_2, S_4)| + |\delta(S_3, S_4)|$

These equalities make it easy to see that $|\delta_G(A)| + |\delta_G(B)| \geq \max\{|\delta_G(A - B)| + |\delta_G(B - A)|, |\delta_G(A \cap B)| + |\delta_G(A \cup B)|\}$.

□

2.4.3 Definition. *A function* $f : 2^V \to \mathbb{Z}$ *is* proper *if* $f(V) = 0$ *and the following two conditions hold:*

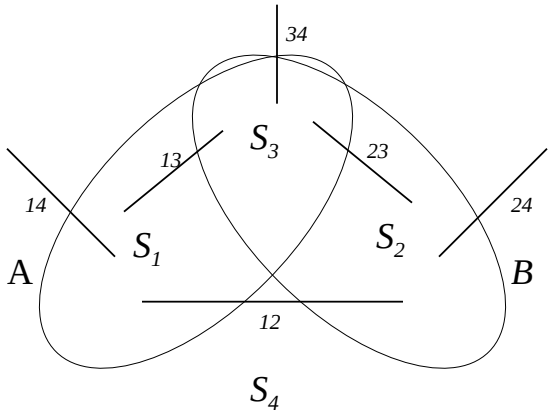

Figure 2.3: A line labelled ij represents edges $\delta(S_i, S_j)$.

1. *For every* $S \subseteq V$, $f(S) = f(\overline{S})$.

2. *For every* $A, B \subseteq V$, $f(A \cup B) \leq \max\{f(A), f(B)\}$.

2.4.4 Definition. *A function* $f : 2^V \to \mathbb{Z}$ *is weakly supermodular if* $f(V) = 0$ *and for every* $A, B \subseteq V$, $f(A) + f(B) \leq \max\{f(A - B) + f(B - A), f(A \cap B) + f(A \cup B)\}$.

2.4.5 Lemma ([24])**.** *Every proper function* f *is weakly supermodular.*

Proof. By the definition of proper functions, for any $A, B \subseteq V$, we have

1. $f(A) = f((A - B) \cup (A \cap B)) \leq \max\{f(A - B), f(A \cap B)\}$
2. $f(A) = f(\overline{A}) \leq \max\{f(B - A), f(\overline{A \cup B})\} = \max\{f(B - A), f(A \cup B)\}$
3. $f(B) = f((B - A) \cup (A \cap B)) \leq \max\{f(B - A), f(A \cap B)\}$
4. $f(B) = f(\overline{B}) \leq \max\{f(A - B), f(\overline{A \cup B})\} = \max\{f(A - B), f(A \cup B)\}$

By summing the two inequalities involving the minimum of $f(A-B), f(B-A), f(A \cap B), f(A \cup B)$ we see that f is weakly supermodular. □

2.4.6 Lemma ([33])**.** *If* f *is weakly supermodular and* g *is submodular, then* $f - g$ *is weakly supermodular.*

Proof. Let $A, B \subseteq V$. We have that $f(A)+f(B) \leq \max\{f(A-B)+f(B-A), f(A \cap B)+f(A \cup B)\}$. Suppose $f(A-B)+f(B-A) = \max\{f(A-B)+f(B-A), f(A \cap B)+f(A \cup B)\}$, the other case proceed similarly. Then,

$$\begin{aligned}(f-g)(A)+(f-g)(B) &= (f(A)+f(B))-(g(A)+g(B)) \\ &\leq (f(A-B)+f(B-A))-(g(A-B)+g(B-A)) \\ &= (f-g)(A-B)+(f-g)(B-A)\end{aligned}$$

Hence

$$\begin{aligned}(f-g)(A)+(f-g)(B) \leq \max\{&(f-g)(A-B)+(f-g)(B-A), \\ &(f-g)(A \cap B)+(f-g)(A \cup B)\}\end{aligned}$$

as desired.

□

2.4.1 Uncrossing Argument for a Weakly Supermodular Function

Consider the linear program (LP-f) presented in section 1.4, for some weakly supermodular function f on a graph $G = (V, E)$. Let x be a basic solution to (LP-f) with $0 < x(e) < 1$ for every $e \in E$. The following uncrossing arguments, first presented in Jain's 2-approximation for SNDP, are used to obtain a laminar basis for x that can be useful in proving that some edge must have an x value above a certain threshold.

We say that a set A is tight if the constraint corresponding to A in (LP-f) is tight.

2.4.7 Lemma ([33])**.** *If two sets A and B are tight, then at least one of the following must hold:*

1. *$A-B$ and $B-A$ are also tight, and $[A]+[B] = [A-B]+[B-A]$*
2. *$A \cap B$ and $B \cup A$ are also tight, and $[A]+[B] = [A \cap B]+[B \cup A]$*

Proof. Since f is weakly supermodular, either $f(A) + f(B) \leq f(A - B) + f(B - A)$ or $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$. We show the proof for the first case, the second case is proven similarly.

As A and B are tight, we get the following two equalities:

$$f(A) = x(A - B, B - A) + x(A - B, \overline{A \cup B}) + x(B - A, A \cap B) + x(A \cap B, \overline{A \cup B}) \tag{2.1}$$

$$f(B) = x(B - A, A - B) + x(B - A, \overline{A \cup B}) + x(A - B, A \cap B) + x(A \cap B, \overline{A \cup B}) \tag{2.2}$$

As x is a feasible solution, we also have

$$f(A - B) \leq x(A - B, B - A) + x(A - B, \overline{A \cup B}) + x(A - B, A \cap B) \tag{2.3}$$

$$f(B - A) \leq x(B - A, A - B) + x(B - A, \overline{A \cup B}) + x(B - A, A \cap B) \tag{2.4}$$

Then (2.1) + (2.2) $\leq$ (2.3) + (2.4) implies that $x(A \cap B, \overline{A \cup B}) = 0$, and that the two inequalities are in fact equalities. Thus $A - B$ and $B - A$ are both tight. Further there is no edge going between $A - B$ and $A \cap B$, so in this case we have that $[A] + [B] = [A - B] + [B - A]$. □

Let $\mathcal{T}$ denote the family of all tight sets. For a family of sets $\mathcal{F}$, let $\textsc{Span}(\mathcal{F})$ denote the vector space spanned by $[A]$ for all $A \in \mathcal{F}$. Lemma 2.4.7 allows us to prove the following:

2.4.8 Lemma ([33])**.** *For any maximal laminar family, $\mathcal{L}$, of tight sets,* $\textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T})$.

Proof. Since $\mathcal{L} \subseteq \mathcal{T}$, $\textsc{Span}(\mathcal{L}) \subseteq \textsc{Span}(\mathcal{T})$. Suppose $\textsc{Span}(\mathcal{L}) \neq \textsc{Span}(\mathcal{T})$. Then there exists a set $A \in \mathcal{T}$ for which $[A] \notin \textsc{Span}(\mathcal{L})$. Choose such an A that crosses the minimum number of sets in $\mathcal{L}$. Let $L \in \mathcal{L}$ be a set that crosses A. By Lemma 2.4.7, we have one of the following occuring:

1. $A - L$ and $L - A$ are tight, and $[A] + [L] = [A - L] + [L - A]$
2. $A \cap L$ and $L \cup A$ are tight, and $[A] + [L] = [A \cap L] + [L \cup A]$

We suppose the first case occurs, proof of the second case follows similarly. Since $[A] \notin \textsc{Span}(\mathcal{L})$, either $[A - L] \notin \textsc{Span}(\mathcal{L})$ or $[L - A] \notin \textsc{Span}(\mathcal{L})$. Again, we prove only the first case, and the second case follows similarly. Thus $[A - L] \notin \textsc{Span}(\mathcal{L})$. Note that $A - L$ does not cross L. By Lemma 2.3.3, any set $L' \in \mathcal{L}$ that crosses $A - L$ must also cross A, contradicting that A crosses a minimal number of sets from $\mathcal{L}$. □

We can now find a laminar family of tight constraints that define our basic feasible solution x, as shown in the following lemma.

2.4.9 Lemma ([33])**.** *There exists a laminar family, $\mathcal{B}$, of tight sets satisfying the following:*

1. $|\mathcal{B}| = |E|$,
2. *The vectors $[A]$ for all $A \in \mathcal{B}$ are linearly independent,*
3. *For every $A \in \mathcal{B}$, $f(A) \geq 1$.*

Proof. Let $\mathcal{L}$ be a maximal laminar family of tight sets. Continually remove any set in this family that can be written as a linear combination of the other members; call the resulting family $\mathcal{B}$. Thus $\mathcal{B}$ satisfies condition 2, and as the span is unchanged, $\textsc{Span}(\mathcal{B}) = \textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T})$, by Lemma 2.4.8. As x is basic, the dimension of $\textsc{Span}(\mathcal{T})$ is $|E|$, showing that $\mathcal{B}$ satisfies condition 1. Finally, note that any tight set must satisfy condition 3 as no edge has zero value. □

2.5 Pair-Laminar Families

The groundwork for this and the following section was done by Fleischer et al. [17] for the proof of their 2-approximation for the Element Connectivity Problem (ELC). The goal is to be able to construct an uncrossing lemma for ELC, similar to the one used in Jain's 2-approximation.

Consider a family of disjoint pairs of subsets of V, $\mathcal{F} \subseteq 2^V \times 2^V$. We refer to a pair $(A, A') \in \mathcal{F}$ as a *set pair*, and we call A the *left-set* and A' the *right-set*.

We define a relation $\leq$ on set pairs by $(A, A') \leq (B, B')$ if $A \subseteq B$ and $B' \subseteq A'$. It can be easily verified that this relation is transitive, reflexive and antisymmetric, and hence

defines a partial order. If $(A, A') \leq (B, B')$ or $(B, B') \leq (A, A')$, we say (A, A') and (B, B') are *comparable*. Note that $(A, A') \leq (B, B')$ if and only if $(B', B) \leq (A', A)$.

2.5.1 Definition. *The pairs* (A, A') *and* (B, B') do not pair-cross *if:*

1. (A, A') *and* (B, B') *are comparable, or*
2. $A \subseteq B'$ *and* $B \subseteq A'$ *(that is,* $(A', A) \leq (B', B)$*).*

Otherwise, we say (A, A') *and* (B, B') pair-cross.

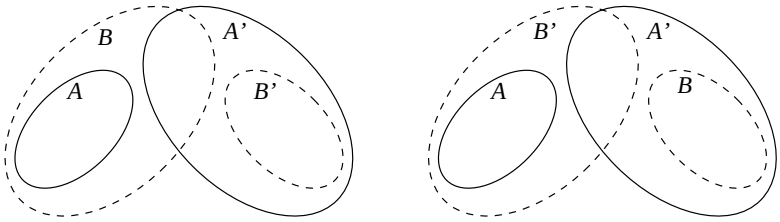

Figure 2.4: Left: Set pairs satisfy $(A, A') \leq (B, B')$. Right: Set pairs (A, A') and (B, B') not comparable, but do not pair-cross. However, (A', A) and (B', B) pair-cross, assuming all sets are non-empty.

Firstly, note that the two conditions in Definition 2.5.1 cannot hold simultaneously if the sets involved are non-empty. Further, note that this definition is not symmetric in the following sense: it is possible that (A, A') and (B, B') do not pair-cross while (A', A) and (B', B) do pair-cross. This asymmetry is necessary to ensure that we can formulate an uncrossing lemma in which one does not get the same set pairs output as were input. We would like our uncrossing lemma applied to crossing pairs (A', A) and (B', B) to return (A, A') and (B, B').

The following lemma is straightforward, but will be useful for reference later.

2.5.2 Lemma. *If* (A, A') *and* (B, B') *do not pair-cross, and* $A \cap B \neq \emptyset$*, then* (A, A') *and* (B, B') *are comparable.*

Proof. If they were not comparable, then $\emptyset \neq A \cap B \subseteq A \subseteq B'$, implying that B and B' are not disjoint. □

2.5.3 Definition. *A collection $\mathcal{L}$ of pairs (A, A') is called* pair-laminar *if no two pairs in $\mathcal{L}$ pair-cross.*

We provide an example of pair-laminar set pairs $\mathcal{L} = \{(A, A'), (B, B'), (C, C')\}$ in Figure 2.5. Notice that $(A, A') \leq (B, B')$, $(C, C') \leq (B, B')$, and (A, A') and (C, C') are not comparable with $\leq$, but they satisfy $A \subseteq C'$, $C \subseteq A'$.

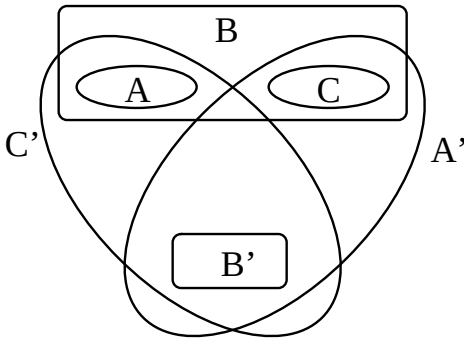

Figure 2.5: A pair-laminar collection $\mathcal{L} = \{(A, A'), (B, B'), (C, C')\}$ of set pairs

The following technical lemma is parallel to Lemma 2.3.3 in the case of laminar families, and is needed for the uncrossing lemma that will soon follow. Its proof, consisting of exhaustive case analysis, is omitted, but can be found in [17].

2.5.4 Lemma ([17])**.** *Let (A, A') and (B, B') be set pairs that pair-cross. Let (X, X') be a set pair that does not pair-cross (B, B'). If (X, X') pair-crosses any one of $(A \cap B, A' \cup B')$, $(A \cup B, A' \cap B')$, $(A \cap B', A' \cup B)$, or $(A' \cap B, A \cup B')$, then (X, X') pair-crosses (A, A') as well.*

2.5.5 Lemma ([17])**.** *Given pairs (A, A') and (B, B'), neither (A, A') nor (B, B') pair-cross any of the four pairs $(A \cap B, A' \cup B')$, $(A \cup B, A' \cap B')$, $(A \cap B', A' \cup B)$, or $(A' \cap B, A \cup B')$.*

Proof. We consider (A, A'), the proof is analagous for (B, B'). We see that:

- $(A \cap B, A' \cup B') \leq (A, A')$
- $(A, A') \leq (A \cup B, A' \cap B')$
- $(A \cap B', A' \cup B) \leq (A, A')$

- $A \subseteq A \cup B'$ and $A' \cap B \subseteq A'$

Thus (A, A') does not pair-cross any of the four sets. □

2.5.1 Forest Induced by a Pair-Laminar Family

We begin by showing the unique forest associated with the pair-laminar family $\mathcal{L}$ from Figure 2.5. Recall that $(A, A') \leq (B, B')$, $(C, C') \leq (B, B')$, and (A, A'),(C, C') are not comparable using $\leq$, but they do not pair-cross. The forest is depicted in Figure 2.6.

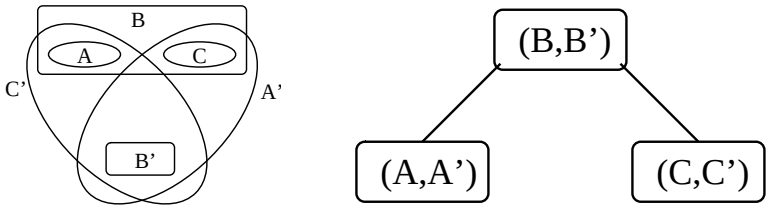

Figure 2.6: A pair-laminar collection $\mathcal{L} = \{(A, A'), (B, B'), (C, C')\}$ of set pairs and its associated forest

2.5.6 Lemma ([17])**.** *If $\mathcal{B}$ is a collection of pair-laminar set pairs, then the poset defined by $\leq$ on the set pairs in $\mathcal{B}$ is described by a unique forest $F_{\mathcal{B}}$.*

Proof. Let (A, A'), (B, B'), $(C, C') \in \mathcal{B}$, with $(B, B') \neq (C, C')$. To show that the partial order defined by $\leq$ gives a forest, we only need to show that if $(A, A') \leq (B, B')$ and $(A, A') \leq (C, C')$, then (B, B') and (C, C') must be comparable in the partial order. This would ensure that every node has at most one parent. Thus it suffices to establish that if

- $(A, A') < (B, B')$
- $(A, A') < (C, C')$
- $(C, C') \not< (B, B')$,

then $(B, B') < (C, C')$. Since $\mathcal{B}$ is pair-laminar, (B, B') and (C, C') do not pair-cross. If $(B, B') \not< (C, C')$, then $B \subseteq C'$ and $C \subseteq B'$, as otherwise they would pair-cross. Then $A \subseteq B \subseteq C'$ and since $C \cap C' = \emptyset$, this implies $A \not\subseteq C$, contradicting $(A, A') \leq (C, C')$. Thus it must be that $(B, B') \leq (C, C')$. □

2.6 Two-Set Functions

In this section we generalize many of the definitions and results found in Section 2.4 to *two-set functions*, that is, functions that are defined over pairs of subsets of some vertex set V. We first define upon exactly what family of subsets we require our two-set functions to be defined.

2.6.1 Definition. *We say a family of pairs of subsets of V, $\mathcal{F} \subseteq 2^V \times 2^V$, is* two-set consistent *if every pair has disjoint subsets, and if whenever $(A, A'), (B, B') \in \mathcal{F}$, we have $(A \cap B, A' \cup B'), (A \cup B, A' \cap B'), (A \cap B', A' \cup B), (A' \cap B, A \cup B')$ all in $\mathcal{F}$ as well.*

Figure 2.7 provides an illustration of the set pairs involved in Definition 2.6.1.

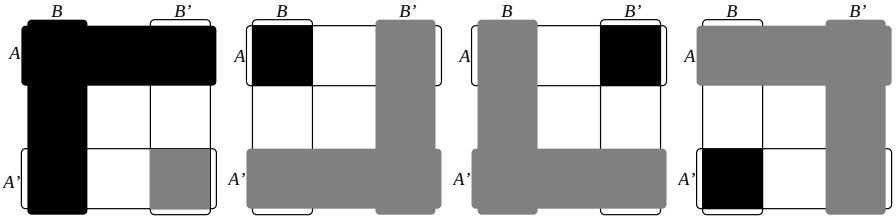

Figure 2.7: From left to right: $(A \cup B, A' \cap B')$, $(A \cap B, A' \cup B')$, $(A \cap B', A' \cup B)$, $(A' \cap B, A \cup B')$

The following definitions are defined in such a way that will allow us to eventually formulate a generalized uncrossing argument.

2.6.2 Definition. *We say that a two-set function f defined on a two-set consistent family on V is* two-submodular *if for all pairs (A, A'), (B, B') we have*

$$f(A, A') + f(B, B') \geq \max\{f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B'), \\ f(A \cap B', A' \cup B) + f(A' \cap B, A \cup B')\} \tag{2.5}$$

2.6.3 Definition. *We say that a two-set function f defined on a two-set consistent family on V is* weakly two-supermodular *if for all pairs (A, A'), (B, B') we have*

$$f(A, A') + f(B, B') \leq \max\{f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B'), \tag{2.6}$$
$$f(A \cap B', A' \cup B) + f(A' \cap B, A \cup B')\} \tag{2.7}$$

When we say a function f is weakly two-supermodular, it is assumed that f is a two-set function defined on some two-set consistent family on V.

As the functions $|\delta_F(\cdot)|$ and $x(\cdot)$ are submodular in the traditional setting, we see that we get a parallel result for two-set functions.

2.6.4 Lemma ([17]). *The two-set functions $|\delta_F(A, A')|$ and $x(A, A')$ are two-submodular.*

Proof. Follows by simple counting arguments that show that any edge counted on the right hand side of (2.5) also appears on the left-hand side. We present one half of the proof, we will show that

$$|\delta_F(A, A')| + |\delta_F(B, B')| \geq |\delta_F(A \cup B, A' \cap B')| + |\delta_F(A \cap B, A' \cup B')| \tag{2.8}$$

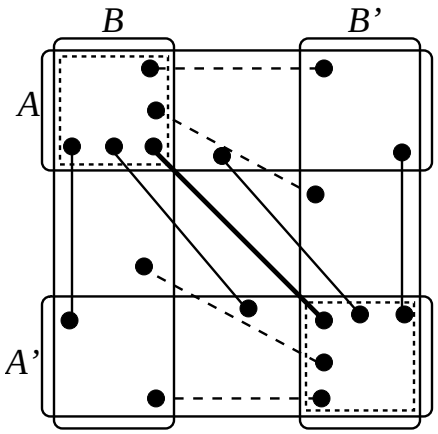

Figure 2.8: Used for proof of Lemma 2.6.4

Figure 2.8 shows all possible edges that can be present in the right hand side of (2.8), with the thick edges appearing twice. Note that any solid thin edge is present on the left hand side in term $|\delta_F(A, A')|$, any dashed edge is present in term $|\delta_F(B, B')|$, and the thick edge is present in both terms. Thus we get that the inequality holds.

□

2.6.5 Lemma ([17]). *If f is weakly two-supermodular and g is two-submodular, then $f - g$ is weakly two-supermodular.*

Proof. Whichever term of the definition of weak two-supermodularity (2.6) or (2.7) achieves the maximum, $-g(A, A') - g(B, B')$ satisfies the same inequality, by (2.5). Hence $(f - g)(A, A') + (f - g)(B, B')$ satisfies this inequality. □

2.6.1 Uncrossing Argument for a Weakly Two-Supermodular Function

This argument is the two-set parallel to the uncrossing argument for supermodular functions, presented in Section 2.4.1.

Consider a basic feasible solution x to the following LP, defined for a graph G and weakly two-supermodular function f:

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) && && \text{(LP2-}f, G, \mathcal{F}\text{)} \\ \text{s.t.} \quad & x(A, A') \geq f(A, A') && \forall (A, A') \in \mathcal{F} \\ & 0 \leq x(e) \leq 1 && \forall e \in E \end{aligned}$$

Suppose further that $0 < x(e) < 1$ for every edge $e \in E$.

We say a set pair (A, A') is *tight* if the corresponding constraint is tight, $x(A, A') = f(A, A')$.

2.6.6 Lemma (Uncrossing Lemma [17])**.** *If (A, A') and (B, B') are tight for a weakly two-supermodular function f with respect to x, then one of the following holds:*

1. *$(A \cup B, A' \cap B')$ and $(A \cap B, A' \cup B')$ are tight, and*
   $[A, A'] + [B, B'] = [A \cup B, A' \cap B'] + [A \cap B, A' \cup B']$
2. *$(A \cap B', A' \cup B)$ and $(A' \cap B, A \cup B')$ are tight and*
   $[A, A'] + [B, B'] = [A \cap B', A' \cup B] + [A' \cap B, A \cup B']$

Proof. As f is weakly two-supermodular, either (2.6) or (2.7) hold. Suppose (2.6) holds, the other case follows similarly. Recall (2.6):

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

We know that $x(A, A')$ is two-submodular by Lemma 2.6.4. Define two-set function h by $h(Z, Z') = f(Z, Z') - x(Z, Z')$ which is weakly two-supermodular by Lemma 2.6.5. As x is feasible, $x \geq f \implies h \leq 0$. As (A, A') and (B, B') are tight, we have $h(A, A') = 0 = h(B, B')$. Thus, by two-supermodularity,

$$0 = h(A, A') + h(B, B') \leq h(A \cup B, A' \cap B') + h(A \cap B, A' \cup B') \leq 0$$

implying that $h(A \cup B, A' \cap B') = 0 = h(A \cap B, A' \cup B')$. Thus $(A \cap B, A' \cup B')$ and $(A \cup B, A' \cap B')$ are tight.

We get that

$$[A, A'] + [B, B'] \geq [A \cup B, A' \cap B'] + [A \cap B, A' \cup B'] \tag{2.9}$$

follows by a counting argument, as every edge that appears in the right-hand side of the inequality must appear in the left-hand side. We wish to show equality, and will do so by contradiction. Suppose the inequality is strict for some edge e. As the x-value of an edge e is the same in all sets that contain e, (2.9) gives that $x(A, A') + x(B, B') \geq x(A \cup B, A' \cap B') + x(A \cap B, A' \cup B')$, and furthermore, as (2.9) is strict for some edge e, we also get $x(A, A') + x(B, B') > x(A \cup B, A' \cap B') + x(A \cap B, A' \cup B')$. But then

$$\begin{aligned}
0 &= h(A, A') + h(B, B') \\
&= f(A, A') + f(B, B') - x(A, A') - x(B, B') \\
&\leq f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B') - x(A, A') - x(B, B') \quad \text{by defition of } f \\
&< f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B') - x(A \cup B, A' \cap B') - x(A \cap B, A' \cup B') \\
&= h(A \cup B, A' \cap B') + h(A \cap B, A' \cup B') = 0,
\end{aligned}$$

a contradiction.

□

2.6.7 Lemma ([17])**.** *Let $\mathcal{T}$ be the collection of all tight set pairs with respect to x. Then any maximal pair-laminar collection $\mathcal{L}$ of tight set pairs satisfies* $\textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T})$.

Proof. Suppose not. Then $\textsc{Span}(\mathcal{L}) \subset \textsc{Span}(\mathcal{T})$ as $\mathcal{L} \subset \mathcal{T}$. Then there must exist a pair $(A, A') \in \mathcal{T}$ with $[A, A'] \notin \textsc{Span}(\mathcal{L})$ that pair-crosses a minimum number of set pairs in $\mathcal{L}$. Let $(B, B') \in \mathcal{L}$ be one of the set pairs in $\mathcal{L}$ that pair-crosses (A, A'). We use the

Uncrossing Lemma (Lemma 2.6.6) which states that we can rewrite $[A, A']$ as a linear combination of characteristic vectors of pair-laminar tight set pairs. The proof proceeds based on which condition of the Uncrossing Lemma holds. Suppose condition 1 holds, and condition 2 holding would proceed similarly.

The lemma gives that $(A \cap B, A' \cup B')$ and $(A \cup B, A' \cap B')$ are tight (that is, they are in $\mathcal{T}$), and

$$[A, A'] = [A \cup B, A' \cap B'] + [A \cap B, A' \cup B'] - [B, B']$$

Since $[A, A'] \notin \textsc{Span}(\mathcal{L})$, at least one of two vectors $[A \cup B, A' \cap B']$ or $[A \cap B, A' \cup B']$ is not in $\textsc{Span}(\mathcal{L})$. We now show that whichever one is not in the span pair-crosses strictly fewer sets in $\mathcal{L}$ than (A, A'), contradicting the definition of (A, A'). Suppose $[A \cup B, A' \cap B'] \notin \textsc{Span}(\mathcal{L})$, the other case would proceed similarly. Lemma 2.5.4 gives that any set pair $(X, X') \in \mathcal{L}$ pair-crossing either of $(A \cup B, A' \cap B')$ must also have pair-cross (A, A'), since (X, X') does not pair-cross (B, B') by pair-laminarity of $\mathcal{L}$. By Lemma 2.5.5 the set pair $(A \cup B, A' \cap B')$ does not pair-cross (B, B'), implying that it pair-crosses fewer set pairs than (A, A'), a contradiction. □

We now get the pair-laminar basis that we were hoping for. This pair-laminar basis is fundamental to the proof of Fleischer et al.'s 2-approximation for ELC.

2.6.8 Corollary ([17])**.** *For any basic solution x, with $0 < x(e) < 1$ for every edge $e \in E$, there exists a pair-laminar set $\mathcal{B}$ of tight set pairs satisfying:*

1. $|\mathcal{B}| = |E|$,
2. *the vectors $[A, A']$ for $(A, A') \in \mathcal{B}$ are linearly independent,*
3. $f(A, A') \geq 1$ *for all* $(A, A') \in \mathcal{B}$

Proof. Let $\mathcal{L}$ be a maximal pair-laminar collection of tight set pairs, $\textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T})$, by Lemma 2.6.7. Continually remove any set pair in this set that can be written as a linear combination of other members of the set; call the resulting set $\mathcal{B}$. Clearly the vectors in $\mathcal{B}$ are now linearly independent, and this operation has not changed the span. Further, as any vector in $\mathcal{B}$ represents a tight constraint, and no $x(e) > 0$ for every edge e, we must have $f(A, A') \geq 1$ for all $(A, A') \in \mathcal{B}$. As the tight constraints in $\mathcal{B}$ are linearly independent and span all tight constraints for basic solution x, the constraints in $\mathcal{B}$ form a basis for x, and hence $|\mathcal{B}| = |E|$. □

Chapter 3

Fractional Tokens

In this section we generalize the fractional token argument presented by Nagarajan et al. [42] in their simplified proof of Jain's 2-approximation for EC-SNDP. They construct a n-by-n linear system that is satisfied by the basic feasible solution x. By giving fractional tokens to each edge, which are then redistributed and counted appropriately using a laminar system of tight constraints, they are able to contradict the assumption that $x(e) < \frac{1}{2}$ for every edge e.

Swamy [50] reformulated the fractional token argument in [42] using matrix notation. This allowed one to show that for any problem of similar structure, there must be some variable of value at least half. We extend this matrix argument, and now allow for a more complex system of constraints.

We show that any solution x to a n-by-n linear system satisfying the properties similar to those used in [42] is guaranteed to have some $x(j)$ above a certain threshold, depending on where column j appears in our system. In the following section, we show that we can define such a system for our basic feasible solution x for SNDP, which will imply that there is some edge e where $x(e) \geq \frac{1}{2}$.

Let $k \in \mathbb{Z}$ be at least 1. Let $M = [A_k, A_{k-1}, ..., A_1]$ be an $n \times n$ matrix. Here each A_i is a matrix with n rows and some number of columns. Suppose every row of M has a non-zero entry, and the sum of the entries of some column does not equal zero. Furthermore, the following properties hold for a column j from A_s:

[M_s1] Every entry in column j of A_s is in $\{+1, 0, -1, -2, ..., -s\}$,

[M_s2] There is at most one negative entry in column j of A_s, call it $-r$ if it exists,

[M_s3] If there is such a negative entry, then there are at most r entries of $+1$ in column j. If not, then there are at most s entries of $+1$.

3.0.9 Lemma. *Let $b \in \mathbb{Z}^n$ be a vector of integers, and let $x = (x_k, x_{k-1}, ..., x_1)^T$ be the solution of the system $[A_k, A_{k-1}, ..., A_1](x_k, x_{k-1}, ..., x_1)^T = b$, with $x > 0$. Then there is a column j from some submatrix A_s for which $x(j) \geq 1/s$.*

Proof. Assume that $x_j < 1/s$ for all columns j in submatrix A_s. We will use a fractional token argument to show a contradiction. Each variable j will receive one token, and we will distribute these over the rows of M such that

- each row receives at least one distinct token, and
- there are some unused tokens among the columns.

Such an argument would clearly show the number of columns of A to be larger than the number of rows, a contradiction.

Let m_{ij} be the entry in row i and column j of matrix M. We use the following token distribution rule. Each column j gives $m_{ij}x(j) \geq 0$ tokens to row i if $m_{ij} \geq 0$, and it gives $1 - m_{ij}x(j) > 0$ tokens to row i otherwise.

Note, that conditions [M_s1]-[M_s3] imply that every column redistributes at most one token. Also note that if $m_{ij} \neq 0$, then column j gives a non-zero amount of tokens to row i.

We now show that each row receives one token. Consider a row i. Let $\boldsymbol{I}^s_+$ be the set of columns j of A_s with $m_{ij} = +1$. Similarly, for $-r \in \{-1, -2, ..., -s\}$ define $\boldsymbol{I}^s_{-r}$ as the set of columns j of A_s with $m_{ij} = -r$.

Row i then receives t_i tokens where

$$\begin{aligned} t_i &:= \sum_{s=1}^{k}(x(\boldsymbol{I}_+^s) + \sum_{r=1}^{s}(|\boldsymbol{I}_{-r}^s| - rx(\boldsymbol{I}_{-r}^s))) \\ &= (\sum_{s=1}^{k}\sum_{r=1}^{s}|\boldsymbol{I}_{-r}^s|) + \sum_{s=1}^{k}(x(\boldsymbol{I}_+^s) - \sum_{r=1}^{s} rx(\boldsymbol{I}_{-r}^s)) \\ &= (\sum_{s=1}^{k}\sum_{r=1}^{s}|\boldsymbol{I}_{-r}^s|) + \sum_{j=1}^{n} m_{ij}x(j) \\ &= (\sum_{s=1}^{k}\sum_{r=1}^{s}|\boldsymbol{I}_{-r}^s|) + b_i \end{aligned}$$

Since b is a vector of integers, t_i is an integer as well. As every row has a positive entry, each row i receives a nonzero amount of tokens. Thus $t_i > 0 \implies t_i \geq 1$.

Finally, consider a column j whose entries do not sum to zero. By conditions $[M_s2]$ and $[M_s3]$, as well as the token redistribution scheme, we see that this column did not redistribute all its tokens, and hence there are some tokens that have not been distributed, a contradiction. □

In the original fractional token argument and the matrix reformulation by Swamy, the matrix used assumed $A_s = \emptyset$ for all $s \neq 2$. This generalized argument can be used in additional settings, and we will show some applications in the sections that follow.

3.1 Application to Jain's 2-Approximation for SNDP

In this section we wish to apply the fractional token argument to complete the proof of Jain's key theorem, Theorem 1.4.1. We let x be a basic feasible solution to (LP-f) (see Section 1.4). We may assume $x(e) < 1$ for every edge e, because if not the algorithm is able to make progress. We also assume $0 < x(e)$ for every edge e, because if there is a zero edge, we can remove it from the problem, and x remains a basic feasible solution of the reduced problem.

We want to show that there must be some edge e with value $x(e) \geq \frac{1}{2}$. Let $\mathcal{B}$ be our laminar basis of tight constraints defining x, as given by 2.4.9. Let $F_{\mathcal{B}}$ be the unique forest induced by $\mathcal{B}$, given by Lemma 2.3.4.

We define a matrix M using $F_{\mathcal{B}}$ where rows are indexed by constraints from $\mathcal{B}$. Columns are indexed by edges $e \in E$. The row for $A \in \mathcal{L}$ is given by $[A] - \sum_{C \text{ child of } A}[C]$. Note that M is non-singular as $\mathcal{B}$ is a basis for x, and so there is some column with entries that do not sum to zero, and certainly every row has a nonzero entry. Define $b \in \mathbb{Z}^{\mathcal{B}}$ by $b_A = f(A) - \sum_{C \text{ child of } A} f(C)$. Note that x is the unique solution to $My = b$. If we can show properties $[M_2 1]$-$[M_2 3]$ apply to M, then Lemma 3.0.9 applies, and we get that $x(e) \geq \frac{1}{2}$ for some edge $e \in E$. Note that we are using the lemma with $k = 2$, $M = [A_2]$. We reproduce properties $[M_2 1]$-$[M_2 3]$ below for our matrix M with entry A, e represented by m_{Ae}.

$[M_2 1]$ $m_{Ae} \in \{0, 1, -1, -2\}$ for every $m_{Ae} \in M$,

$[M_2 2]$ There is at most one negative entry in column e of M, call it $-r$ if it exists,

$[M_2 3]$ if there is such a negative entry, then there are at most r entries of $+1$ in column j. If not, then there are at most 2 entries of $+1$.

We now argue that properties $[M_2 1]$-$[M_2 3]$ apply for this system. Consider a node A and edge $e = uv$. Note that at most two of A's children can have the same edge in their cut, as otherwise the children would not be disjoint, violating laminarity of $\mathcal{L}$. For an edge $e = uv$, the lowest node containing u and the lowest node containing v are the only nodes that can have a positive entry, and these are the only nodes possible to have e in their cut, and yet not have any children with e in their cut.

If A has entry -1 for edge $e = uv$, then there is a unique child, say C with one endpoint, say u in C (see Figure 3.1). We must have $v \in A$, as otherwise e would be in $\delta(A)$. Further, v cannot be in a separate child, as then e would be in that child's cut, resulting in an entry of -2 for A. The lowest node containing u will not contain v as it is a descendent of C, and hence gets entry of 1 for column e. As A is the lowest node containing v, but e is not in $\delta(A)$, there is no other positive entry for this column.

Finally, if A has entry -2 for edge e, then there are two children of A, one containing u, the other containing v, with e in their cuts (see Figure 3.2). The lowest node containing u will thus have e in its cut, and hence an entry of 1, similarly the lowest node containing v will have an entry of 1 as well.

Hence Lemma 3.0.9 applies, and there must be some edge e with $x(e) \geq \frac{1}{2}$, proving Theorem 1.4.1.

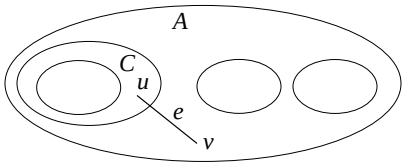

Figure 3.1: A set A with entry -1 for edge e.

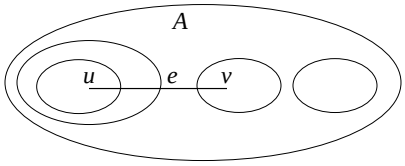

Figure 3.2: A set A with entry -2 for edge e.

Chapter 4

Element Connectivity

In this chapter, we look at the Element Connectivity Problem (ELC). The problem, originally formulated by Jain et al. [32], is motivated by its application in real computer networks. In such networks, both links and routers may fail, but typically network terminals are more robust [17]. The failure of a network terminal is uncommon, and less vital to the connectivity of the network as a whole. As such, the designer wants to build a network protecting against the failure of both links and routers.

We begin by looking at a 2-approximation algorithm by Fleischer et al. [17]. Their algorithm is inspired by Jain's 2-approximation for SNDP. Its proof follows the same outline as Jain's, except that all the results must be generalized to functions that operate on pairs of subsets of vertices. Some basic material on pair-laminar families and two-set functions is available in Sections 2.5 and 2.6 respectively, and this material will be assumed knowledge. Fleischer et al. finish their proof with an endpoint based argument in [17], but here we use the generalized fractional token argument from Chapter 3 to give a much simplified version of the proof.

We also prove a $(2, 2B + 3)$-approximation algorithm for the Degree Bounded Element Connectivity problem (ELC-B). This algorithm, due to the work of Lau et al. [37], is modelled off of the authors $(2, 2B + 3)$-approximation for SNDP-B. In Section 4.2 we generalize the framework to two-set functions and pair-laminar families to prove the corresponding result for ELC-B.

4.1 A 2-Approximation Algorithm for ELC by Fleischer et al.

In [17], Fleischer et al. proves a generalization of Jain's theorem [33], which allows them to prove a 2-approximation for ELC through iterative rounding using a very similar approach as Jain's for EC-SNDP. The difficulty is in formulating an appropriate generalization and proving the analagous technical results, including an uncrossing lemma. We present Fleischer's algorithm here, and will use our generalized fractional token argument to complete the proof. We will then use many of the results proven in this section to prove an approximation algorithm for the degree bounded version of this problem.

4.1.1 Preliminaries

Let $G = (V, E)$ be an undirected graph, let $n = |E|$. Let $c(e)$ be a cost for each edge $e \in E$. Partition vertices into *terminals* R and *nonterminals* Q. Let r_{uv} define our symmetric connectivity requirements between every pair of vertices u, v, but allowing nonzero connectivity only between pairs of terminals. We define the set of *elements* to be all edges and nonterminal vertices. The ELC problem is to find a minimum-cost subgraph H of G such that for every pair u, v of vertices there are at least r_{uv} element-disjoint u, v-paths in H.

We will use Menger's Theorem to formulate an appropriate integer program to solve the ELC instance. Menger's Theorem (see Theorem 2.1.2) implies the following corollary, by using an auxiliary graph that adds a vertex in the middle of each edge, and a clique on all neighbours of any required vertex $r \in R - \{s, t\}$.

4.1.1 Corollary ([17])**.** *The minimum number of elements in $E \cup Q$ separating s from t in G is equal to the maximum number of element disjoint paths from s to t in G.*

A subgraph H of G satisfies the connectivity requirements if, for every pair of terminals $u, v \in R$, for every subset $X \subseteq E \cup Q$ with $|X| < r_{uv}$, we have that u and v are in the same component of $G - X$.

4.1.2 Definition. *We define an* ELC set pair *to be a pair (A, A') where $A, A' \subseteq V$ are disjoint and cover all terminals, that is $R \subseteq A \cup A'$. Note that any vertex in $\overline{(A \cup A')}$ must be a nonterminal.*

Define the two-set function f_{elt} over all ELC set pairs (A, A') by $f_{elt}(A, A') = max\{r_{uv} : u \in A \cap R, v \in A' \cap R\}$, and set $f_{elt}(A, A') = 0$ if either A or A' is empty.

For any set-pair (A, A'), there can be at most one element-disjoint path from A to A' through each vertex in $\overline{(A \cup A')}$, as any such vertex is a nonterminal. Thus, for any feasible subgraph H, the number of edges in $\delta_H(A, A')$ must be at least $f_{elt}(A, A') - |Q - A \cup A'|$ $(= f_{elt}(A, A') - |\overline{(A \cup A')}|)$. Thus we define the two-set function g_{elt} by $g_{elt}(A, A') = f_{elt}(A, A') - |Q - A \cup A'|$. This motivates the following linear program and lemma.

Given a a family of pairs of subsets of V, $\mathcal{F} \subseteq 2^V \times 2^V$ and a function $f : \mathcal{F} \to \mathbb{N}$, consider the following linear program:

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) && && \text{(LP2-}f, G, \mathcal{F}\text{)} \\ \text{s.t.} \quad & x(A, A') \geq f(A, A') && \forall (A, A') \in \mathcal{F} \\ & 0 \leq x(e) \leq 1 && \forall e \in E \end{aligned}$$

4.1.3 Lemma. *Let $\mathcal{F}$ be the family of all ELC set pairs. The set of integral solutions to the linear program (LP2-$g_{elt}, G, \mathcal{F}$) equals the set of solutions to the corresponding element connectivity problem.*

Proof. Certainly any solution to the element connectivity problem must have, for any set-pair (A, A'), at least $g_{elt}(A, A')$ edges between A and A', by the preceeding discussion. Let $H = (V, E')$ be the graph of an integral solution to (LP2-$g_{elt}, G, \mathcal{F}$). Suppose there is some set of elements $X \subseteq E \cup Q$ of size $|X| < r_{uv}$ such that u and v are not connected in $H - X$. Let A be all vertices in the component of $H - X$ containing u, and let $A' = (V - X \cap Q) - A$. Hence there is no edge between A and A' in $H - X$, by definition, and $R \subseteq A \cup A'$, thus

(A, A') is a set-pair. We have

$$\begin{aligned}
0 &= x_{(H-X)}(A, A') \\
&= x_H(A, A') - x_X(A, A') \\
&\geq x(A, A') - |X \cap E| \\
&\geq g_{elt}(A, A') - |X \cap E| \text{ by feasibility} \\
&= f_{elt}(A, A') - |Q - A \cup A'| - |X \cap E| \\
&\geq r_{uv} - |X \cap Q| - |X \cap E| \text{ by choice of } (A, A') \\
&= r_{uv} - |X| > 0 \text{ as } |X| < r_{uv}
\end{aligned}$$

As we have an integral solution, this implies that there must be an edge between A and A' in $H - X$, a contradiction. Hence any integral solution to (LP2-$g_{elt}, G, \mathcal{F}$) is a feasible solution to the ELC problem. □

Just as Jain's 2-approximation for SNDP is in fact a 2-approximation for (IP-f), we are going to create an approximation algorithm that finds a 2 approximate integer solution to (LP2-$f, G, \mathcal{F}$) for any *weakly two-supermodular* function f (see Definition 2.6.3). Note that we require such a function to be required over a *two-consistent* family of set pairs (see Definition 2.6.1).

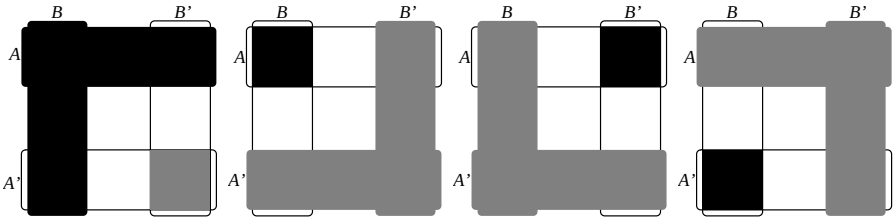

Figure 4.1: From left to right: $(A \cup B, A' \cap B')$, $(A \cap B, A' \cup B')$, $(A \cap B', A' \cup B)$, $(A' \cap B, A \cup B')$

4.1.4 Lemma. *The family $\mathcal{F}$ of all ELC set pairs is two-set consistent.*

Proof. Let $(A, A'), (B, B')$ be two ELC set pairs. Clearly each of $(A \cap B, A' \cup B'), (A \cup B, A' \cap B'), (A \cap B', A' \cup B), (A' \cap B, A \cup B')$ are disjoint pairs of sets, as can be seen in Figure 4.1.

Secondly, let U denote the union of one of the listed set pairs. Note that $R \subseteq A \cup A'$ and $R \subseteq B \cup B'$ thus

$$\begin{aligned} R &\subseteq (A \cap B) \cup (A \cap B') \cup (A' \cap B) \cup (A' \cap B') \\ &\subseteq U \qquad \text{as seen in Figure 4.1} \end{aligned}$$

Hence each pair is an ELC set-pair. □

The approximation algorithm will be based on the following theorem:

4.1.5 Theorem. *[[17]] For any weakly two-supermodular function $f \neq 0$, any basic solution to (LP2-$f, G, \mathcal{F}$) has at least one variable e such that $x(e) \geq 1/2$.*

We will prove this theorem later in Section 4.1.4 using the generalized fractional token argument presented in Chapter 3.

4.1.2 Presentation of the Algorithm

Given a polynomial-time separation oracle for (LP2-$f, G, \mathcal{F}$) we describe the 2-approximation algorithm for solving the associated integer program.

Algorithm 2 Iterative Rounding 2-approximation

1: $F \leftarrow \emptyset$
2: $G' \leftarrow G = (V, E)$
3: $f' \leftarrow f$
4: **while** F is not feasible for (LP2-$f, G, \mathcal{F}$) **do**
5: Solve (LP2-$f', G', \mathcal{F}$), let x^* be an optimal basic solution
6: $F \leftarrow F \cup \{e : x^*(e) \geq \frac{1}{2}\}$
7: $G' \leftarrow (V, E - F)$
8: $f'(A, A') \leftarrow f(A, A') - |\delta_F(A, A')|$
9: **end while**

Note that our residual problem is formed by subtracting of a subset of rounded x values from edges that are chosen for the final solution, and any such x value is originally at least $\frac{1}{2}$. Thus we are able to prove that the output solution has cost no more than a factor

of 2 times that of the optimal solution in an identical fashion to the proof of the parallel theorem in Jain's 2-approximation for SNDP (see Section 1.4).

Lemmas 2.6.4 and 2.6.5 then give us the following corollary, which justify that our algorithm can be applied recursively.

4.1.6 Corollary ([17]). *If f is weakly two-supermodular, then $f(A, A') - |\delta_F(A, A')|$ is also weakly two-supermodular for any set of edges $F \subseteq E$.*

Theorem 4.1.5 and the above corollary then imply the following theorem.

4.1.7 Theorem ([17]). *Given a polynomial-time separation oracle for (LP-$f, G, \mathcal{F}$) and any subsequent LP created by the algorithm, Algorithm 2 is a polynomial-time 2-approximation algorithm for finding the minimum cost integer solution to (LP-$f, G, \mathcal{F}$).*

To obtain a 2-approximation for ELC, we thus we need to show that g_{elt} is weakly two-supermodular. We start by showing that f_{elt} is weakly two-supermodular. First, we require some additional notation. Given (A, A'), there is a pair $i \in A \cap R$, $j \in A' \cap R$ that determines the value of $f_{elt}(A, A')$. Let $i_{(A,A')}$ denote one such i and $j_{(A,A')}$ denote a corresponding j, and call these witnesses for the set pair (A, A').

4.1.8 Lemma ([17]). *The two-set function f_{elt} is weakly two-supermodular.*

Proof. Given ELC set pairs (A, A'), (B, B'), we need to show one of the following:

- $f(A, A') + f(B, B') \leq f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B')$ (2.6)
- $f(A, A') + f(B, B') \leq f(A \cap B', A' \cup B) + f(A' \cap B, A \cup B')$ (2.7)

Note that for any pair (B, B'), $R \subseteq B \cup B'$, thus $i_{(A,A')} \in (A \cap B) \cup (A \cap B')$ and $j_{(A,A')} \in (A' \cap B) \cup (A' \cap B')$. Figure 4.2 depicts all possible locations for each of $i_{(A,A')}$, $i_{(B,B')}$, $j_{(A,A')}$, $j_{(B,B')}$.

Call sets $A \cap B$ and $A' \cap B'$ complements, and sets $A \cap B'$ and $A' \cap B$ complements. Note that

$$I = \{i_{(A,A')}, i_{(B,B')}, j_{(A,A')}, j_{(B,B')}\}$$

intersects two, three, or all four of the sets $A \cap B$, $A' \cap B'$, $A \cap B'$ and $A' \cap B$.

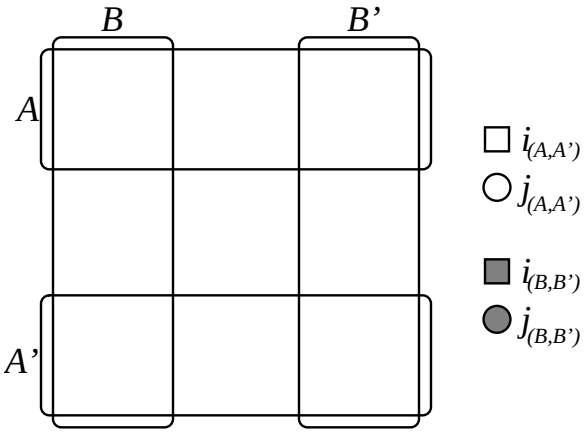

Figure 4.2: Showing witness legend

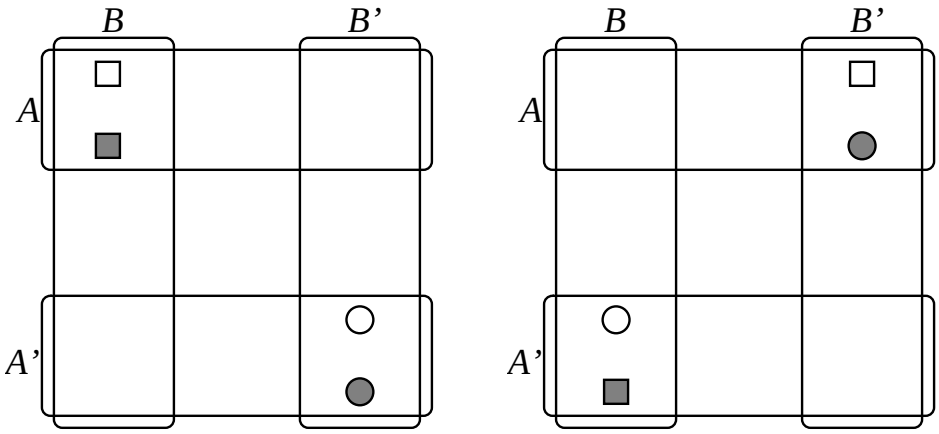

Figure 4.3: I intersects two sets

If only two, then these sets are complements, and we are in one of the two cases depicted in Figure 4.3. In the left case we get that (2.6) holds, and in the right case (2.7) holds. Note that $f_{elt}(A, A') = f_{elt}(B, B')$.

If I intersects three sets, then two of these are complements with the property that a witness for (A, A') is in one set, and a witness for (B, B') is in the other. We are thus in a situation symmetric to one of those depicted in Figure 4.4. The inequality of corresponding to that complementary pair holds. For example, in the left figure, we are looking at complementary pair $(A \cap B', A' \cup B)$ and $(A' \cap B, A \cup B')$, and we see that inequality (2.7) must hold.

Finally, we have the case where I intersects all four sets, as in Figure 4.5. In this

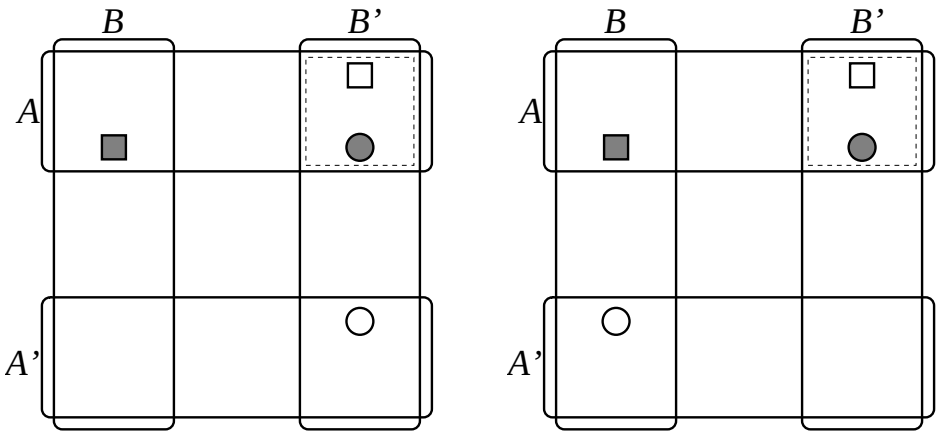

Figure 4.4: I intersects three sets

case, select the complementary pair that contains a the witness pair (i, j) with $r_{ij} = \max\{f_{elt}(A, A'), f_{elt}(B, B')\}$. Then the inequality that corresponds to this complementary pair holds. Say, for example, $r_{i_{(A,A')},j_{(B,B')}} = \max\{f_{elt}(A, A'), f_{elt}(B, B')\}$. In our figure, these are located in complementary pair $(A \cap B', A' \cup B)$ and $(A' \cap B, A \cup B')$, and we see that the corresponding inequality (2.7) must hold.

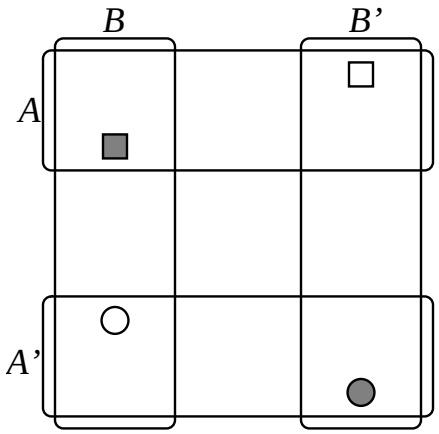

Figure 4.5: I intersects four sets

□

4.1.9 Corollary ([17])**.** *The two-set function $g_{elt}(A, A')$ is weakly two-supermodular.*

Proof. Notice that the function $h(A, A') = |Q - (A \cup A')|$ satisfies (2.5) with equality as

all arguments are disjoint pairs. Thus h is two-submodular. Applying Lemma 2.6.5 gives that $g_{elt} = f_{elt} - h$ is weakly two-supermodular.

□

4.1.3 Polynomial Time Separation Oracle

Recall that we require a polynomial time separation oracle for (LP-$g_{elt}, G, \mathcal{F}$) to as well as any subsequent LP generated in order to be able to run our 2-approximation algorithm. To do this, we interpret x-values as capacities and transform the graph induced by the current fractional solution x and the fixed edges F into a directed graph by replacing every edge by oppositely oriented edges with the same capacity as the original undirected edge, with fixed edges getting capacity of 1. We then perform a procedure of splitting nonterminal vertices to model the fact that at most one path can pass through any nonterminal, see Figure 4.6. Then in the resulting graph, the maximum flow value between u and v is the vertex connectivity between u and v. If this is less than r_{uv}, the minimum cut reveals a violated inequality. This gives us a polynomial-time separation oracle for (LP-$g_{elt}, G, \mathcal{F}$) and thus we can obtain a basic solution in polynomial time using the ellipsoid algorithm [27].

4.1.4 Application of the Fractional Token Argument

All that remains to show is the existence of a $\frac{1}{2}$-edge in any basic solution. Let x be a basic feasible solution to (LP-$f, G, \mathcal{F}$) for a weakly two-supermodular function f. If $x(e) = 0$ for some $e \in E$, we can remove edge e from the graph G and variable $x(e)$ from (LP-$f, G, \mathcal{F}$). The residual solution x remains a basic feasible solution to the modified LP. Thus we assume without loss of generality that $x(e) > 0$ for all $e \in E$. Thus henceforth, E is the support of x. We may assume $x(e) < 1$ for all $e \in E$, as otherwise Theorem 4.1.5 holds trivially.

We say a set pair (A, A') is *tight* if $x(A, A') = f(A, A')$. Through a standard uncrossing argument for weakly two-supermodular functions (see Section 2.6.1), we can obtain a laminar basis $\mathcal{B}$ that defines x, given by Lemma 2.6.8. Let $F_{\mathcal{B}}$ be the unique forest defined by $\mathcal{B}$. We will use this forest to show that there must be some edge in x of value at least $\frac{1}{2}$.

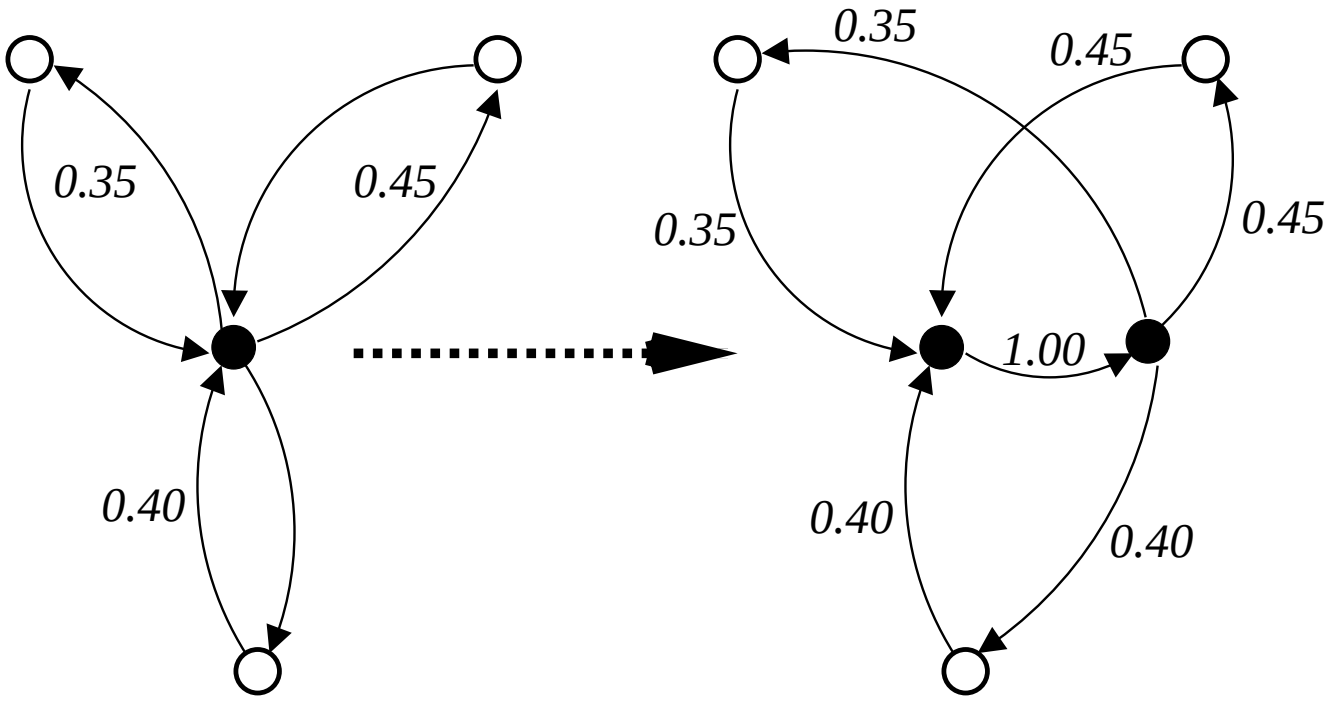

Figure 4.6: Splitting procedure - labels are capacities.

To apply the generalized fractional token argument, we use our forest $F_{\mathcal{B}}$ to define a matrix M with rows indexed by set pairs $(A, A') \in \mathcal{B}$ and columns indexed by edges $e \in E$. The row for (A, A') is given by $[A, A'] - \sum_{(B,B') \text{ child of } (A,A')} [B, B']$. Note that M is non-singular as the vectors $[A, A']$ for $(A, A') \in \mathcal{B}$ are linearly independent. Define $b \in \mathbb{Z}^{\mathcal{B}}$ by $b_{(A,A')} = f(A, A') - \sum_{(B,B') \text{ child of } (A,A')} f(B, B')$.

Note that x is the unique solution to $My = b$. We want to apply Lemma 3.0.9, to show that we get that $x(e) \geq \frac{1}{2}$ for some edge $e \in E$. Note that we are applying the lemma with $M = A_2$, so we need to satisfy conditions $[M_2 1]$-$[M_2 3]$ for the matrix M. As M is non-singular, we have some column whose entries do not sum to zero, and every row has a nonzero entry. Here we reproduce the conditions for convenience, and then present lemmas to show that they do indeed hold for our system.

$[M_s 1]$ Every entry in column j of A_s is in $\{+1, 0, -1, -2, ..., -s\}$,

$[M_s 2]$ There is at most one negative entry in column j of A_s, call it $-r$ if it exists,

$[M_s 3]$ If there is such a negative entry, then there are at most r entries of $+1$ in column j.

If not, then there are at most s entries of $+1$.

[$M_2$1] $m_{(A,A'),e} \in \{0, 1, -1, -2\}$ for every $m_{(A,A'),e} \in M$,

[$M_2$2] There is at most one negative entry in column e of M, call it $-r$ if it exists,

[$M_2$3] If there is such a negative entry, then there are at most r entries of $+1$ in column e. If not, there there are at most 2 entries of $+1$.

See Figures 4.7 and 4.8 for an illustration of edges that would create a negative value for a set pair (A, A'). We first prove a few quick lemmas that will help us show the desired properties hold.

4.1.10 Lemma. *If $(A, A'), (B, B') \in \mathcal{B}$ and $A \cap B \neq \emptyset$, then (A, A') and (B, B') are comparable via $\leq$.*

Proof. Suppose not. Then $A \subseteq B' \implies B \cap B' \neq \emptyset$, a contradiction as set pairs are disjoint. □

4.1.11 Lemma. *If $(C, C') \leq (A, A')$ are set pairs in $\mathcal{B}$, and some edge $e = uv$ is in both $\delta(A, A')$ and $\delta(C, C')$, then for any (B, B') satisfying $(C, C') \leq (B, B') \leq (A, A')$, $e \in \delta(B, B')$.*

Proof. Say, without loss of generality, $u \in A, v \in A'$. Suppose $v \in C$, thus $u \in C'$. But $v \in A' \subseteq C'$ implies that $C \cap C' \neq \emptyset$, a contradiction.

Thus $u \in C, v \in C'$. Then $u \in C \subseteq B$ and $v \in A' \subseteq B'$, and $e \in \delta(B, B')$. □

4.1.12 Lemma. *Consider node $(A, A') \in \mathcal{B}$ with children (B_i, B_i') for $i = 1...k$. For any edge $e = uv$, there are at most two children with e in their cut. Furthermore, if there are two, then $e \notin \delta(A, A')$.*

Proof. Suppose $k \geq 3$ and e is in the cut of 3 distinct children. Without loss of generality, say $e \in \delta(B_1, B_1') \cap \delta(B_2, B_2') \cap \delta(B_3, B_3')$. Since $e = uv$ is in the three cuts, either u or v is in two of B_1, B_2, B_3. Say $u \in B_1 \cap B_2$. But (B_1, B_1') and (B_2, B_2') are siblings, and hence not comparable via $\leq$, hence $u \in B_1 \subseteq B_2'$ implies $B_2 \cap B_2' \neq \emptyset$, a contradiction, thus e can be in at most two of the children's cuts.

Suppose $k = 2$ and $e \in \delta(B_1, B_1') \cap \delta(B_2, B_2')$. Say $u \in B_1, v \in B_1'$ and $v \in B_2, u \in B_2'$. Then $\{u, v\} \subseteq B_1 \cup B_2 \subseteq A \implies e \notin \delta(A, A')$. (see Figure 4.8 (right figure) for an illustration). □

Note that if three set pairs contain an edge e in their cut, then two of them must share a vertex in their left component, and hence are comparable by Lemma 4.1.10. Thus we cannot have three set pairs that are pair-wise not comparable, and all having e in their cuts.

We now argue that properties $[M_2 1]$-$[M_2 3]$ apply for this system. Lemma 4.1.12 immediately gives that $[M_2 1]$ holds.

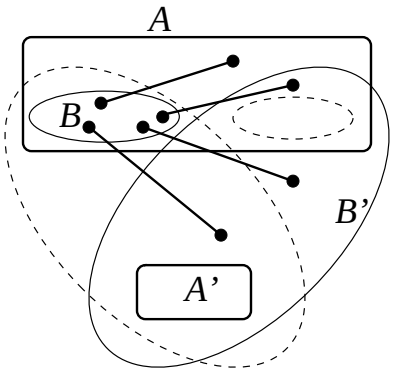

Figure 4.7: A set pair (A, A') with entry -1 for shown edges.

Consider a node (A, A') and edge $e = uv$. Note that at most two of (A, A')'s children can have the same edge in their cut by Lemma 4.1.12. For $e = uv$, the lowest node containing u in the left set and the lowest node containing v in the left set are the only nodes that can have a positive entry, and these are the only nodes possible to have e in their cut, and yet not have any children with e in their cut. This is due to the fact that for $(B, B') \leq (C, C')$, $B' \subseteq C'$.

If (A, A') has entry -1 for edge $e = uv$, then there is a unique child, say (C, C') with one endpoint, say u in C (see Figure 4.7). No ancestor to (A, A') has e in its cut, as otherwise so would (A, A') by Lemma 4.1.11, contradicting that (A, A') has entry -1 for e. We must have $v \notin A'$, as otherwise e would be in $\delta(A)$, and then the entry for (A, A') would be 0, not -1. Further, v cannot be in a separate child, as then e would be in that child's cut, resulting in an entry of -2 for (A, A'). The lowest node containing u will have v in its right set, as it is a descendent of (C, C'), and hence gets entry of 1 for column e.

Any other node (B, B') not comparable to (A, A') with e in its cut must have $v \in B, u \in A'$, but then $B \subseteq A'$, a contradiction as $v \notin A'$. Hence there is no other node with e in its cut, so we have one node with entry -1 and another with 1 for edge e.

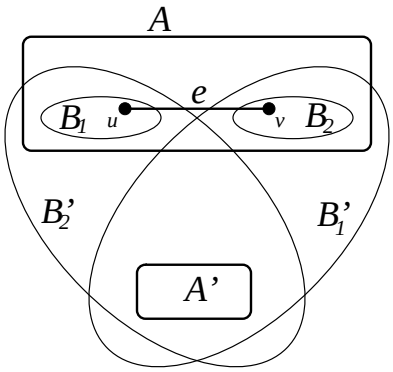

Figure 4.8: A set A with entry -2 for edge e.

Finally, if (A, A') has entry -2 for edge e, then there are two children of (A, A'), one containing u, the other containing v, with e in their cuts (see Figure 4.8). The lowest node containing u will thus have e in its cut, and hence an entry of 1, similarly the lowest node containing v will have an entry of 1 as well.

Hence Lemma 3.0.9 applies, and there must be some edge e with $x(e) \geq \frac{1}{2}$, proving Theorem 1.4.1.

4.2 A $(2, 2B + 3)$-approximation for degree bounded ELC

In this section we look at the degree bounded version of the element connectivity problem (ELC-B). We will be using a lot of the machinery developped in the proof of Fleischer's 2-approximation for ELC. We will model our framework on Lau et al's $(2, 2B + 3)$-approximation for EC-SNDP in their extended abstract [37]. They claim that their methodology can be extended to get a $(2, 2B + 3)$-approximation for the element connectivity version of the problem with degree bounds on terminals only. This thesis will provide the proof of this algorithm.

4.2.1 Preliminaries

Let $G = (V, E)$ be our graph, with vertices partitioned into terminals R and non-terminals Q. Let $c(e)$ be our cost for edge $e \in E$, and let r_{uv} be our connectivity requirement for vertices u, v, with $r_{uv} > 0 \iff \{u, v\} \subseteq R$. Let $W \subseteq R$ be our set of terminals with degree constraints. For a vertex $v \in W$ let L_v and B_v be our lower and upper bounds on v respectively.

We have $f_{elt}(A, A') = max\{r_{uv} : u \in A \cap R, v \in A' \cap R\}$ and $g_{elt}(A, A') = f_{elt} - |Q - (A \cup A')|$, as defined in Fleischer's ELC algorithm. We let $\mathcal{F}$ be the set of all ELC set pairs. Then, with the same reasoning as before, integer points of the following LP are exactly the solutions to the ELC-B instance when $f = g_{elt}$:

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) & & \\ \text{s.t.} \quad & x(A, A') \geq f(A, A') & & \forall (A, A') \in \mathcal{F} \\ & x(\delta(v)) \leq B_v & & \forall v \in W \\ & x(\delta(v)) \geq L_v & & \forall v \in W \qquad (4.1) \\ & 0 \leq x(e) \leq 1 & & \forall e \in E \end{aligned}$$

First we show that we may assume that there are no constraints of form (4.1). For any lower bound $L_v > 0$ for a vertex $v \in W$, degree constraints of the form $x(\delta(v)) \geq L_v$ can be incorporated as connectivity constraints by setting $f(\{v\}, \overline{\{v\}}) = \max\{g_{elt}(\{v\}, \overline{\{v\}}), L_v\}$, and $f = g_{elt}$ otherwise. We show that f is still weakly two-supermodular.

4.2.1 Lemma. *f as defined above is weakly two-supermodular.*

Proof. Let (A, A'), (B, B') be two ELC set pairs. Note that if neither one of (A, A') or (B, B') has the form $(\{v\}, \overline{\{v\}})$ for some $v \in W : L_v > g_{elt}(\{v\}, \overline{\{v\}})$, then

$$f(A, A') + f(B, B') = g_{elt}(A, A') + g_{elt}(B, B'),$$

and the result follows, as g_{elt} is weakly two-supermodular and $g_{elt} \leq f$.

Suppose, without loss of generality, $(A, A') = (\{v\}, \overline{\{v\}})$ for some $v \in W$ with $L_v > g_{elt}(\{v\}, \overline{\{v\}})$. As $v \in R$, either $v \in B$ or $v \in B'$.

If $v \in B$ then it is easy to see that

$$f(A, A') + f(B, B') = f(A \cup B, A' \cap B') + f(A \cap B, A' \cup B'),$$

and if $v \in B'$ then

$$f(A, A') + f(B, B') = f(A \cap B, A' \cup B')' + f(A' \cap B, A \cup B').$$

□

Thus we have shown that we may assume, without loss of generality, that there are no lower bound constraints, as these can be incorporated into a function f that is still weakly two-supermodular. Hence we now want to find integer solutions to the following LP:

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) && && \text{(LP3-}f, G, \mathcal{F}, B\text{)} \\ \text{s.t.} \quad & x(A, A') \geq f(A, A') && \forall (A, A') \in \mathcal{F} \\ & x(\delta(v)) \leq B_v && \forall v \in W \\ & 0 \leq x(e) \leq 1 && \forall e \in E \end{aligned}$$

4.2.2 Presentation of the Algorithm

The $(2, 2B + 3)$-approximation algorithm is as follows:

4.2.2 Theorem. *Provided that the algorithm terminates, F is a $(2, 2B+3)$-approximation.*

Proof. The proof that this algorithm is a 2-approximation on the cost is identical to that of the proof in Jain's 2-approximation for SNDP, see Section 1.4.

We prove that for every vertex $v \in W$, $\deg_F(v) \leq 2B_v + 3$. Consider a degree constraint, say for vertex v, and focus on the last iteration in which B'_v changes or the constraint gets deleted. Suppose we have added α edges incident to v prior to this iteration. Since each edge added had value $\geq \frac{1}{2}$, $B_v \geq \alpha/2 + B'_v$, or restated, $\alpha \leq 2B_v - 2B'_v$.

If in this is the last iteration in which B'_v is changed (constraint not deleted), say we add $\beta \geq 1$ more edges incident to v. Thus the final degree of v is $\alpha + \beta$. We must have had $B'_v \geq \beta/2$ as $x(e) \geq \frac{1}{2}$. Therefore we have $\alpha + \beta \leq 2B_v - 2B'_v + 2B'_v = 2B_v$.

Algorithm 3 Iterative Rounding $(2, 2B + 3)$-approximation

$F \leftarrow \emptyset$
$G' \leftarrow G = (V, E)$
$f' \leftarrow f$
$B' \leftarrow B$
while F is not feasible for (LP3-$f, G, \mathcal{F}, B$) **do**
    Solve (LP3-$f', G', \mathcal{F}, B'$), let x^* be an optimal basic solution
    If there exists a $v \in W$ such that $|\delta(v)| \leq 4$, remove v from W, return to while
    For any edge $e = uv$ where $x^*(e) \geq \frac{1}{2}$, add e to F, decrease B'_u and B'_v by $x^*(e)$
    $G' \leftarrow (V, E - F)$
    $f'(A, A') \leftarrow f(A, A') - |\delta_F(A, A')|$ for all $(A, A') \in \mathcal{F}$
end while

If we delete this constraint in this iteration and solve the problem for the relaxed version, in the worst case the final solution contains all 4 remaining edges incident with v. Thus, in the final solution, the degree of v is at most $\alpha + 4 \leq 2B_v - 2B'_v + 4$. Since $\alpha + 4$ is an integer, and $B'_v > 0$, this gives that $\alpha + 4 \leq 2B_v + 3$, as desired. □

4.2.3 Polynomial Time Separation Oracle

The separation oracle for (LP3-$f, G, \mathcal{F}, B$) is nearly identical to the separation oracle used in Fleischer's 2-approximation for ELC, see Section 4.1.3. The only difference is that for every $v \in W$, we check that $x(\delta(v)) \leq B_v$, and return this constraint if the proposed solution x does not satisfy the degree bound.

4.2.4 Basic Feasible Solution Structure Lemma

Our goal is now to show that this algorithm does indeed terminate. We wish to show that in every iteration there is some progress made. Suppose that this is not the case. Let x be our basic optimal solution to (LP3-$f, G, \mathcal{F}, B$). Suppose that $|\delta(v)| \geq 5$ for every $v \in W$, and $0 < x(e) < \frac{1}{2}$ for every $e \in E$. We use a counting argument to arrive at a contradiction. We start by getting some structure on our basic feasible solution x.

4.2.3 Theorem. *Let f be a weakly two-supermodular function, x a basic feasible solution of (LP) such that $0 < x(e) < 1$ for all $e \in E$. Then there exists a pair-laminar basis $\mathcal{B}$ that partitions into a set of singletons $\mathcal{D}$ for degree constraints and the remaining set $\mathcal{C}$ for connectivity constraints where:*

1. *Every set $(\{v\}, V - \{v\}) \in \mathcal{D}$ has $x(\delta(v)) = B_v > 0$,*
2. *Every set $(A, A') \in \mathcal{C}$ has $x(A, A') = f(A, A') \geq 1$,*
3. *$|\mathcal{B}| = |\mathcal{D}| + |\mathcal{C}| = |E|$,*
4. *Characteristic vectors $[A, A']$ for all $(A, A') \in \mathcal{B}$ are linearly independent,*
5. *x is the unique solution to the equations*

$$\{x(\delta(v)) = B_v : (\{v\}, V - \{v\}) \in \mathcal{D}\} \bigcup \{x(A, A') = f(A, A') : (A, A') \in \mathcal{C}\}$$

Proof. Let $\mathcal{T}_\mathcal{D}$ and $\mathcal{T}_\mathcal{C}$ be the set of all tight degree constraints and the set of all tight connectivity constraints respectively. By Lemma 2.6.7, we can let $\mathcal{L}$ be a maximal pair-laminar collection of tight set pairs for connectivity constraints such that $\textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T}_\mathcal{C})$. Continually remove any constraint from $\mathcal{L} \cup \mathcal{T}_\mathcal{D}$ that can be written as a linear combination of others. Let the resulting constraints be $\mathcal{D} \subseteq \mathcal{T}_\mathcal{D}$ and $\mathcal{C} \subseteq \mathcal{L} \subseteq \mathcal{T}_\mathcal{C}$. As $\mathcal{L}$ is pair-laminar, so is $\mathcal{C}$. We have that $\textsc{Span}(\mathcal{D} \cup \mathcal{C}) = \textsc{Span}(\mathcal{T}_\mathcal{D} \cup \mathcal{L}) = \textsc{Span}(\mathcal{T}_\mathcal{D} \cup \mathcal{T}_\mathcal{C}) = |E|$. Thus $|\mathcal{D} \cup \mathcal{C}| = |E|$. Conditions 1 and 2 are clear as $x(e) > 0$ for any edge e. Conditions 4 and 5 are clear by construction.

We show that $\mathcal{B} := \mathcal{D} \cup \mathcal{C}$ is pair-laminar. Any two sets $(\{v_1\}, V - \{v_1\}), (\{v_2\}, V - \{v_2\}) \in \mathcal{D}$ do not pair-cross as $\{v_1\} \subseteq V - \{v_2\}$ and $\{v_2\} \subseteq V - \{v_1\}$. Consider $(A, A') \in \mathcal{C}$ and $(\{v\}, V - \{v\}) \in \mathcal{D}$. Either $v \in A$ in which case $(\{v\}, V - \{v\}) \leq (A, A')$ or $v \notin A$ in which case $v \in A'$ as $R \subseteq A \cup A'$, thus $\{v\} \subseteq A'$ and $A \subseteq V - \{v\}$. Thus the set pairs do not pair cross, and so $\mathcal{B}$ is pair-laminar.

□

Note that any set pair is only included once in $\mathcal{B}$ due to linear independence, and hence will correspond to either a degree or connectivity constraint.

4.2.5 Endpoint Based Counting Argument

In this section we present the endpoint based counting argument that will contradict that $0 < x(e) < \frac{1}{2}$ for every edge e, and $|\delta(v)| \geq 5$ for every $v \in W$.

Every edge $e = uv$ has two endpoints, u and v. There are a total of $2|E|$ endpoints in G. If we can assign endpoints to set pairs in $\mathcal{B}$ in such a way that root nodes get at least 3 endpoints, and any other node gets at least 2, then we get that $2|E| > 2|\mathcal{B}|$, contradicting Theorem 4.2.3.

Construct the unique pair-laminar forest $F_{\mathcal{B}}$ on $\mathcal{B}$ given by Lemma 2.5.6. We begin by defining how a set pair can own an endpoint.

4.2.4 Definition. *We say that a set pair $(A, A') \in \mathcal{B}$* left-owns *endpoint u of edge $e = uv$ if (A, A') is the lowest node in $F_{\mathcal{B}}$ with $u \in A$ and $v \in A'$ (implying $e \in \delta(A, A')$).*

4.2.5 Definition. *We say that a set pair $(A, A') \in \mathcal{B}$* right-owns *endpoint u of edge $e = uv$ if (A, A') is the parent of the highest node (C, C') in $F_{\mathcal{B}}$ with $v \in C$ and $u \in C'$ (implying $e \in \delta(C, C')$).*

Note that any two set pairs with a common element in their left-set are comparable by Lemma 2.5.2, and hence both of the above notions of owning an endpoint are well defined. We would like at most one set pair to own an endpoint, and this is accomplished with the following definition.

4.2.6 Definition. *We say that a set pair $(A, A') \in \mathcal{B}$* owns *endpoint u of edge $e = uv$ if it is the lowest node that either left-owns or right-owns u.*

We must show that this notion of owning an endpoint is well defined.

4.2.7 Lemma. *The definition of a set pair owning an endpoint is well defined.*

Proof. Consider an endpoint u of an edge $e = uv$. Since the notions of left-owning and right-owning an endpoint are well defined, we just need to show that if one set pair (A, A') left-owns u and another (B, B') right-owns u, then (A, A') and (B, B') are comparable, and hence one is lower.

If (B, B') right-owns u, then (B, B') is parent to the highest set pair (C, C') with $v \in C$, $u \in C'$. Thus $v \in C \subseteq B$. Now if (A, A') and (B, B') were not comparable, we have $u \in A \subseteq B'$, but this contradicts the definition of (C, C'). □

Figure 4.9 illustrates what a set pair (A, A') looks like when it owns endpoint u of edge $e = uv$.

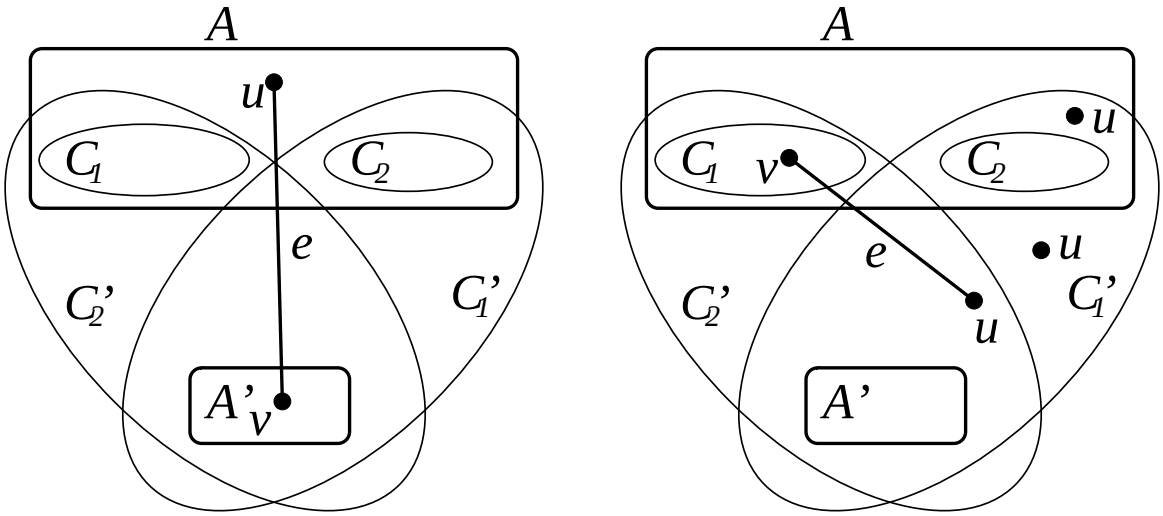

Figure 4.9: (A, A') owns endpoint u of e. Left: (A, A') left-owns u. Right: (A, A') right-owns u, also shows 2 other possible locations for u.

We will need a few technical lemmas before we can complete the counting argument.

4.2.8 Lemma. *Suppose (A, A') owns endpoint u of $e = uv$. Then*

1. *(A, A') does not own endpoint v of e,*
2. *$u \notin A'$,*
3. *For any child (C, C') of (A, A'), $u \notin C$.*

Proof. 1. Suppose (A, A') owns endpoint v of e. Clearly it cannot left-own both or right-own both u and v, so say without loss of generality that it left-owns u and right owns v. Thus $u \in A, v \in A'$, and (A, A') is parent to some (C, C') with $v \in C, u \in C'$. But $v \in C \subseteq A$, a contradiction.

2. Suppose $u \in A'$. Then (A, A') must right-own u, so (A, A') is parent to the highest node (C, C') with $v \in C, u \in C'$. But $v \in C \subseteq A$, contradicting the definition of (C, C').

3. Suppose $u \in C$. If (A, A') left-owns u then $v \in A' \subseteq C'$, contradicting that (A, A') left-owns u.

   If (A, A') right-owns u then (A, A') has some child (B, B') with $v \in B, u \in B'$. Thus $(C, C') \neq (B, B')$. But then we have $u \in C$ and $v \in B \subseteq C'$, again implying that there is a lower set that left-owns u.

□

4.2.9 Lemma. *Let $(A, A') \in \mathcal{B}$ have k children $(C_i, C_i'), i = 1...k$. Set pair (A, A') owns an endpoint of e if and only if e is in exactly one of the cuts $\delta(A, A')$, $\delta(C_i, C_i')$ for $i = 1...k$.*

Proof. ($\Rightarrow$) Say (A, A') owns endpoint u of edge $e = uv$.

Case 1: $e \in \delta(C_i, C_i')$ for some i.

Note that $u \notin C_i$ by Lemma 4.2.8 fact 3. Thus $v \in C_i, u \in C_i'$, see Figure 4.10.

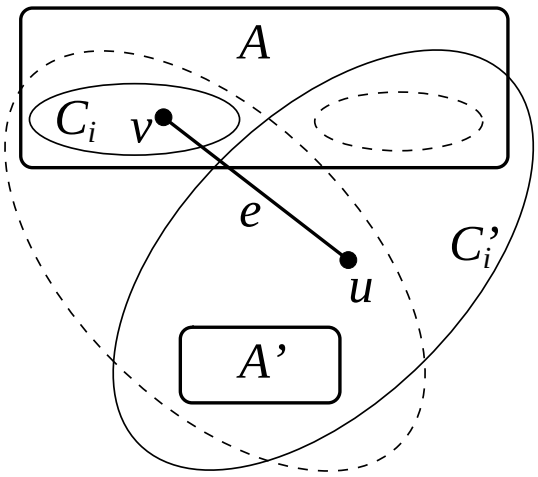

Figure 4.10: Edge e from proof of Lemma 4.2.9

We cannot have $e \in \delta(C_j, C_j')$ for some other child (C_j, C_j') of (A, A'), as then we would have $u \in C_j'$, violating Lemma 4.2.8 fact 3. Further, $v \in C_i \implies v \in A$, and $u \notin A' \implies e \notin \delta(A, A')$, by Lemma 4.2.8 fact 2.

Case 2: $e \notin \delta(C_i, C_i')$ for any i.

Then a must left-own u, implying that $e \in \delta(A, A')$.

($\Leftarrow$) Suppose e is in exactly one of the cuts $\delta(A, A')$, $\delta(C_i, C_i')$ for $i = 1...k$.

Case 1: $e \in \delta(A, A')$

If $u \in A, v \in A'$, we immediately get that $v \in C_i'$ for each child. As $e \notin \delta(C_i, C_i')$ for any i, we have $u \notin C_i$ for any child, thus (A, A') owns endpoint u of e.

Case 2: $e \in \delta(C_i, C_i')$ for some i.

Say without loss of generality that $v \in C_i$, $u \in C_i'$. For every other child (C_j, C_j'), $v \in C_i \subseteq C_j'$. Thus $u \notin C_j$ for any child (C_j, C_j') as this would imply $e \in \delta(C_j, C_j')$ (see Figure 4.10).

Note that $v \in C_i \subseteq A$. As $e \notin (A, A')$, $u \notin A'$, and thus (A, A') is parent to the highest node (C_i, C_i') with $v \in C_i, u \in C_i'$, and no lower node than (A, A') has u in its left-set. Hence (A, A') owns endpoint u.

□

We now use an idea that comes from the original proof of Jain's 2-approximation for SNDP [33]. We define a notion of *corequirement* for a set, and a will look at sets that have low corequirement because these set pairs will have the fewest endpoints available to them. We will have to treat these set pairs more carefully than the others.

4.2.10 Definition. *We define the* corequirement *of an edge e to be* $\text{COREQ}(e) = \frac{1}{2} - x(e)$, *and the corequirement of a set pair to be* $\text{COREQ}(A, A') = \sum_{e \in \delta(A,A')} \text{COREQ}(e) = \frac{1}{2}|\delta(A, A')| - x(A, A')$. *Note that if* $(A, A') \in \mathcal{C}$ *then* $\text{COREQ}(A, A') = \frac{1}{2}|\delta(A, A')| - f(A, A')$.

Note that for $(A, A') \in \mathcal{C}$, $\text{COREQ}(A, A')$ is an integer if and only if $|\delta(A, A')|$ is even, and $\text{COREQ}(A, A') \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, ...\}$ if and only if $|\delta(A, A')|$ is odd. In the latter case, we say $\text{COREQ}(A, A')$ is *half-integral*.

We can now state the lemma that will give us our final contradiction.

4.2.11 Lemma. *Let T be a subtree of our forest rooted at node (A, A'). The endpoints owned by T can be redistributed in such a way that (A, A') gets at least 3, and each descendent gets at least 2. Further, if* $\text{COREQ}(A, A') \neq \frac{1}{2}$ *then (A, A') gets at least 4, and if $(A, A') \in \mathcal{D}$ then (A, A') gets at least 5.*

We will prove this lemma by induction. But before we can prove the lemma, we need to build some facts about set pairs with low corequirement. As mentioned, these sets have to be treated more carefully because they have fewer available endpoints. Note that any set pair corresponding to a degree requirement is a leaf in our pair laminar family.

The following three lemmas will help us prove the inductive case of our lemma. The only difficult cases are when we are at a non-leaf node with only connectivity constraints as children. In particular, the more sets with low corequirement, the more difficult it will be to accumulate endpoints for the root node. As we are only dealing with tight connectivity constraints, we know that the f-value of any set pairs involved is an integer.

The following lemma is quite lengthy, but does involve some nice counting arguments. It is used to cover the case of when we have a node with at most three children, all with low corequirement, and when that node owns three minus the number of children in endpoints.

4.2.12 Lemma. *Suppose $(A, A') \in \mathcal{C}$ has $\alpha \geq 1$ children, all in $\mathcal{C}$, each with corequirement of $\frac{1}{2}$, and (A, A') owns β endpoints. If $\alpha + \beta = 3$, then* $\text{COREQ}(A, A') = \frac{1}{2}$.

Proof. We begin by showing that $|\delta(A, A')|$ is odd, which implies that $\text{COREQ}(A, A') \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, ...\}$. We then complete the proof by showing that $\text{COREQ}(A, A') < \frac{3}{2}$.

Let $(C_1, C_1'), ..., (C_\alpha, C_\alpha')$ be the children of (A, A'). Note that as each child (C_i, C_i') has $\text{COREQ}(C_i, C_i') = \frac{1}{2}$, we know $|\delta(C_i, C_i')|$ is odd.

Case $\alpha = 1, \beta = 2$:

For convenience, we write $(C, C') = (C_1, C_1')$. As these are tight connectivity constraints, we have

$$f(A, A') = x(A, A') = x(\delta(A, A') - \delta(C, C')) + x(\delta(A, A') \cap \delta(C, C')) \tag{4.2}$$
$$f(C, C') = x(C, C') = x(\delta(C, C') - \delta(A, A')) + x(\delta(A, A') \cap \delta(C, C')) \tag{4.3}$$

As both $f(A, A')$ and $f(C, C')$ are integers, by subtracting (4.2) − (4.3) we get that

$$x(\delta(A, A') - \delta(C, C')) - x(\delta(C, C') - \delta(A, A')) \in \mathbb{Z} \tag{4.4}$$

If this value is nonzero then one of $\delta(A, A') - \delta(C, C')$ or $\delta(C, C') - \delta(A, A')$ has at least 3 edges, since each edge has value less than $\frac{1}{2}$. But these are the exact edges that

give (A, A') an endpoint by Lemma 4.2.9, and (A, A') owns $\beta = 2$ endpoints, hence (4.4) must be zero. Since (A, A') owns exactly 2 endpoints, there must be one edge in each set $\delta(A, A') - \delta(C, C')$ and $\delta(C, C') - \delta(A, A')$. Thus

$$\begin{aligned}|\delta(A, A')| &= |\delta(A, A') - \delta(C, C')| + |\delta(A, A') \cap \delta(C, C')| \\ &= |\delta(C, C') - \delta(A, A')| + |\delta(A, A') \cap \delta(C, C')| \\ &= |\delta(C, C')|, \text{ which is odd.}\end{aligned}$$

Case $\alpha = 2, \beta = 1$:

Note that any edge in $\delta(C_1, C_1') \cap \delta(C_2, C_2')$ has one endpoint in C_1 and the other in C_2, both contained within S. Thus $\delta(A, A') \cap \delta(C_1, C_1') \cap \delta(C_2, C_2') = \emptyset$.

We define the following sets to partition our edges:

$$\begin{aligned}\mathbf{A} &:= \delta(A, A') - (\delta(C_1, C_1') \cup \delta(C_2, C_2')) \\ \mathbf{B} &:= \delta(C_1, C_1') \cap \delta(A, A') \\ \mathbf{C} &:= \delta(C_2, C_2') \cap \delta(A, A') \\ \mathbf{D} &:= \delta(C_1, C_1') - (\delta(A, A') \cup \delta(C_2, C_2')) \\ \mathbf{E} &:= \delta(C_2, C_2') - (\delta(A, A') \cup \delta(C_1, C_1')) \\ \mathbf{F} &:= \delta(C_1, C_1') \cap \delta(C_2, C_2')\end{aligned}$$

Notice that

$$\begin{aligned}\delta(A, A') &= \mathbf{A} \cup \mathbf{B} \cup \mathbf{C} \\ \delta(C_1, C_1') &= \mathbf{B} \cup \mathbf{D} \cup \mathbf{F} \\ \delta(C_2, C_2') &= \mathbf{C} \cup \mathbf{E} \cup \mathbf{F}\end{aligned}$$

Lemma 4.2.9 gives that $\mathbf{A} \cup \mathbf{D} \cup \mathbf{E}$ are the exact edges that contribute endpoints to (A, A'). Thus $|\mathbf{A} \cup \mathbf{D} \cup \mathbf{E}| = |\mathbf{A}| + |\mathbf{D}| + |\mathbf{E}| = \beta = 1$. As $|\delta(C_1, C_1')|$ and $|\delta(C_2, C_2')|$ are both odd, we have that

$$\begin{aligned}|\delta(C_1, C_1')| + |\delta(C_2, C_2')| + 1 \qquad &\text{is odd} \\ (|\mathbf{B}| + |\mathbf{D}| + |\mathbf{F}|) + (|\mathbf{C}| + |\mathbf{E}| + |\mathbf{F}|) + (|\mathbf{A}| + |\mathbf{D}| + |\mathbf{E}|) \qquad &\text{is odd} \\ |\mathbf{B}| + |\mathbf{C}| + |\mathbf{A}| = |\mathbf{A} \cup \mathbf{B} \cup \mathbf{C}| = |\delta(A, A')| \qquad &\text{is odd}\end{aligned}$$

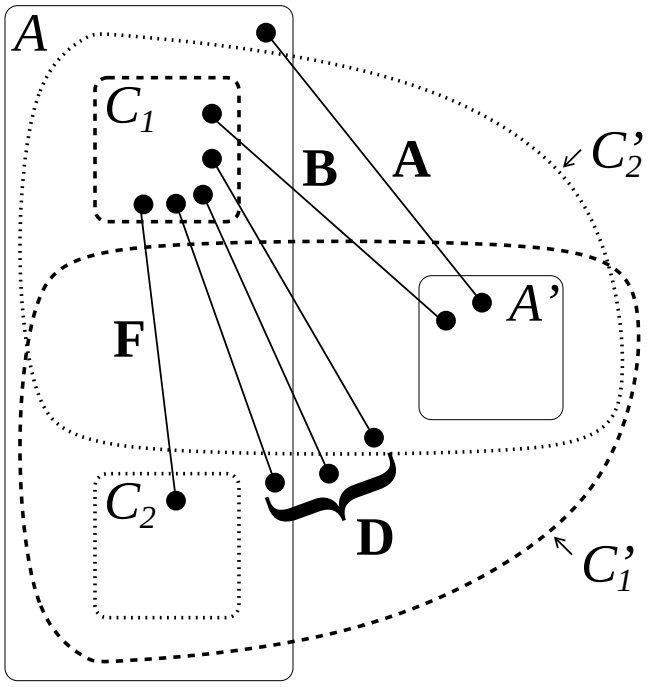

Figure 4.11: Illustration of edge partition when $\alpha = 2$. Note that **C** and **E** are omitted, but they are similar to **B** and **D** respectively.

Case $\alpha = 3, \beta = 0$:

As before, $\delta(A, A') \cap \delta(C_i, C_i') \cap \delta(C_j, C_j') = \emptyset$ for any $i \neq j$. Similarly, $\delta(C_1, C_1') \cap \delta(C_2, C_2') \cap \delta(C_3, C_3') = \emptyset$. As (A, A') owns no endpoints, Lemma 4.2.9 implies that any edge $e \in \delta(A, A') \bigcup_{i=1}^{3} \delta(C_i, C_i')$ is a member of exactly two of the cuts $\delta(A, A')$, $\delta(C_1, C_1')$, $\delta(C_2, C_2')$, $\delta(C_3, C_3')$.

We define the following sets to partition our edges:

$$\begin{aligned}
\mathbf{A} &:= \delta(A, A') \cap \delta(C_1, C_1') \\
\mathbf{B} &:= \delta(A, A') \cap \delta(C_2, C_2') \\
\mathbf{C} &:= \delta(A, A') \cap \delta(C_3, C_3') \\
\mathbf{D} &:= \delta(C_1, C_1') \cap \delta(C_2, C_2') \\
\mathbf{E} &:= \delta(C_1, C_1') \cap \delta(C_3, C_3') \\
\mathbf{F} &:= \delta(C_2, C_2') \cap \delta(C_3, C_3')
\end{aligned}$$

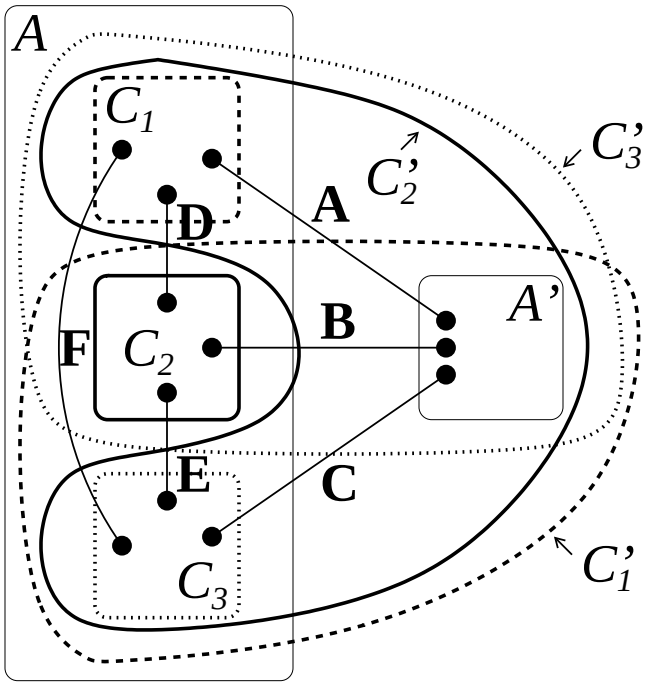

Figure 4.12: Illustration of edge partition when $\alpha = 3$.

Notice that

$$\begin{aligned}
\delta(A, A') &= \mathbf{A} \cup \mathbf{B} \cup \mathbf{C} \\
\delta(C_1, C_1') &= \mathbf{A} \cup \mathbf{D} \cup \mathbf{E} \\
\delta(C_2, C_2') &= \mathbf{B} \cup \mathbf{D} \cup \mathbf{F} \\
\delta(C_3, C_3') &= \mathbf{C} \cup \mathbf{E} \cup \mathbf{F}
\end{aligned}$$

As $|\delta(C_i, C_i')|$ is odd for each $i = 1, 2, 3$, we get that

$$\begin{aligned}
|\delta(C_1, C_1')| + |\delta(C_2, C_2')| + |\delta(C_3, C_3')| &\quad \text{is odd} \\
(|\mathbf{A}| + |\mathbf{D}| + |\mathbf{E}|) + (|\mathbf{B}| + |\mathbf{D}| + |\mathbf{F}|) + (|\mathbf{C}| + |\mathbf{E} + |\mathbf{F}|) &\quad \text{is odd} \\
|\mathbf{A}| + |\mathbf{B}| + |\mathbf{C}| = |\mathbf{A} \cup \mathbf{B} \cup \mathbf{C}| = |\delta(A, A')| &\quad \text{is odd}
\end{aligned}$$

We now complete the proof by showing that $\textsc{coreq}(A, A') < \frac{3}{2}$, which implies that $\textsc{coreq}(A, A') = \frac{1}{2}$ as the corequirement of a set pair (A, A') is half integral if $|\delta(A, A')|$ is

odd. We have that

$$\begin{aligned}
\textsc{coreq}(A, A') &= \sum_{e \in \delta(A,A')} \textsc{coreq}(e) \\
&\leq \sum_{i=1}^{\alpha} \sum_{e \in \delta(C_i, C_i')} \textsc{coreq}(e) + \sum_{e \in \delta(A,A') - [\cup_{i=1}^{\alpha} \delta(C_i, C_i')]} \textsc{coreq}(e) \qquad (4.5) \\
&\leq \sum_{i=1}^{\alpha} \textsc{coreq}(C_i, C_i') + \sum_{e \text{ has endpoint owned by } (A,A')} \textsc{coreq}(e) \\
&= \frac{\alpha}{2} + \sum_{e \text{ has endpoint owned by } (A,A')} \textsc{coreq}(e) \qquad (4.6)
\end{aligned}$$

Recall that $\alpha + \beta \leq 3$. As the sum in (4.6) has β terms and $\textsc{coreq}(e) < \frac{1}{2}$, if $\beta > 0$ then $\textsc{coreq}(A, A') < \frac{3}{2}$.

If $\beta = 0$, then refer to the sets defined above in Case $\alpha = 3$, $\beta = 0$, and see Figure 4.12. Note that there must be some edge in one of **D**,**E**, or **F**, as otherwise $[A, A'] = [C_1, C_1'] + [C_2, C_2'] + [C_3, C_3']$, contradicting linear independence of $\mathcal{C}$, given by Theorem 4.2.3. This implies that the inequality on (4.5) is strict, thus $\textsc{coreq}(A, A') < \frac{3}{2}$.

□

The following lemma is to ensure that a node with a single low corequirement child can acquire at least three endpoints in total. It will be able to take one endpoint from its child, and will own at least two itself.

4.2.13 Lemma. *If a node $(A, A') \in \mathcal{C}$ has exactly one child $(C, C') \in \mathcal{C}$, then (A, A') owns at least 2 endpoints.*

Proof. By the same reasoning as Case $\alpha = 1, \beta = 2$ of Lemma 4.2.12, we have that line (4.4) holds:

$$x(\delta(A, A') - \delta(C, C')) - x(\delta(C, C') - \delta(A, A')) \in \mathbb{Z}$$

We cannot have that both the sets $\delta(A, A') - \delta(C, C')$ and $\delta(C, C') - \delta(A, A')$ are empty, otherwise $[A, A'] = [C, C']$, a contradiction to Theorem 4.2.3. If one set is empty, then in order for the above quantity to be an integer, the other set must have at least 3 edges, as every edge has value $\leq \frac{1}{2}$. If both are non-empty, then there is at least one edge in each set. Every one of these edges provides an endpoint to (A, A') by Lemma 4.2.9, and thus (A, A') owns at least 2 endpoints. □

This final lemma is similar to the previous one, but in the case that a node has two children, at least one of which has low corequirement. If we can show this node owns one endpoint, then it will be able to accumulate enough endpoints in the inductive case.

4.2.14 Lemma. *If a node $(A, A') \in \mathcal{C}$ has two children (C_1, C_1') and (C_2, C_2') in $\mathcal{C}$, and $\textsc{coreq}(C_1, C_1') = \frac{1}{2}$, then (A, A') owns at least one endpoint.*

Proof. Suppose (A, A') does not own any endpoints. Then the sets $\delta(A, A') - (\delta(C_1, C_1') \cup \delta(C_2, C_2'))$, $\delta(A, A') - (\delta(C_1, C_1') \cup \delta(C_2, C_2'))$, and $\delta(A, A') - (\delta(C_1, C_1') \cup \delta(C_2, C_2'))$ are all empty, by Lemma 4.2.9 (see Figure 4.13).

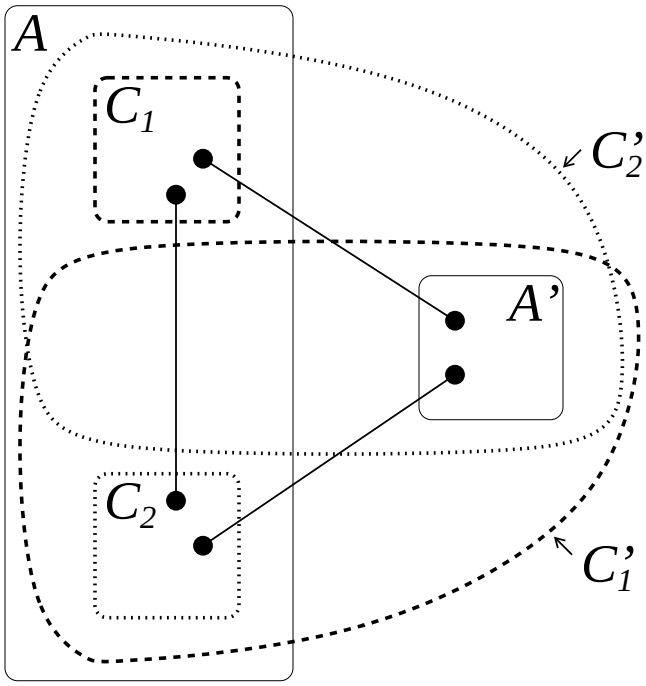

Figure 4.13: Every edge is in two cuts.

Note that

$$|\delta(A, A')| = |\delta(C_1, C_1')| + |\delta(C_2, C_2')| - 2|\delta(C_1, C_1') \cap \delta(C_2, C_2')| \tag{4.7}$$

As $\textsc{coreq}(C_1, C_1') = \frac{1}{2}$, $|\delta(C_1, C_1')|$ is odd, thus by (4.7) the parity of $|\delta(A, A')|$ and

$|\delta(C_2, C_2')|$ are not the same. Further,

$$\begin{aligned}
\textsc{coreq}(A, A') - \textsc{coreq}(C_2, C_2') &= \sum_{e \in \delta(A,A')} \textsc{coreq}(e) - \sum_{e \in \delta(C_2,C_2')} \textsc{coreq}(e) \\
&= \sum_{e \in \delta(A,A') \cap \delta(C_1,C_1')} \textsc{coreq}(e) - \sum_{e \in \delta(C_2,C_2') \cap \delta(C_1,C_1')} \textsc{coreq}(e) \\
&\leq \sum_{e \in \delta(C_1,C_1')} \textsc{coreq}(e) \qquad (4.8) \\
&= \textsc{coreq}(C_1, C_1') = \frac{1}{2}
\end{aligned}$$

Similarly,

$$\begin{aligned}
\textsc{coreq}(C_2, C_2') - \textsc{coreq}(A, A') &= \sum_{e \in \delta(C_2,C_2') \cap \delta(C_1,C_1')} \textsc{coreq}(e) - \sum_{e \in \delta(A,A') \cap \delta(C_1,C_1')} \textsc{coreq}(e) \\
&\leq \sum_{e \in \delta(C_1,C_1')} \textsc{coreq}(e) \qquad (4.9) \\
&= \textsc{coreq}(C_1, C_1') = \frac{1}{2}
\end{aligned}$$

As $[A, A'] \neq [C_1, C_1'] + [C_2, C_2']$, we must have $\delta(C_2, C_2') \cap \delta(C_1, C_1') \neq \emptyset$, which implies that the inequality (4.8) is strict. Similarly, as $[C_2, C_2'] \neq [A, A'] + [C_1, C_1']$, we must have $\delta(A, A') \cap \delta(C_1, C_1') \neq \emptyset$ implying that the inequality (4.9) is strict as well. Thus the corequirements of (A, A') and (C_2, C_2') differ by strictly less than $\frac{1}{2}$, which implies that they are equal. But this contradicts that $|\delta(A, A')|$ and $|\delta(C_2, C_2')|$ have different parity. □

Finally, we are ready to prove our counting argument. The following proves Lemma 4.2.11. Recall that we have a subtree T of our pair-laminar forest, and we want to redistribute endpoints in T in such a way that the root (A, A') gets at least 3, and each descendent gets at least 2. Furthermore, if $\textsc{coreq}(A, A') \not/ \frac{1}{2}$ then (A, A') gets at least 4, and if (A, A') is a degree constraint, then (A, A') gets at least 5.

Proof. Note that any set (A, A') of fractional value $x(A, A')$ must be a degree constraint. For any degree constraint $(\{v\}, V - \{v\}) \in \mathcal{D}$, this set pair is a leaf and owns 5 endpoints as $|\delta(v)| \geq 5$. Consider $(A, A') \in \mathcal{C}$ that is a leaf. As $x(A, A') = f(A, A') \geq 1$ and every edge has value $< \frac{1}{2}$, $|\delta(A, A')| \geq 3$ so (A, A') owns at least 3 endpoints. If $|\delta(A, A')| = 3$

then $\textsc{coreq}(A, A') = \frac{1}{2}$ and only 3 endpoints are required, and otherwise (A, A') collects at least 4 endpoints.

We say that a set pair has a *surplus* of p if $p + 2$ endpoints have been assigned to it inductively. Consider a non-leaf $(A, A') \in \mathcal{C}$. Apply this endpoint redistribution procedure to each subtree of (A, A') inductively.

If (A, A') has two or more children, one of which is a degree constraint, then it can collect 3 surplus endpoints from the degree constraint, and 1 surplus endpoint from any other child.

If (A, A') has one child (C, C') that is a degree constraint, then $\delta(A, A') \neq \delta(C, C')$ by linear independence. Thus there is some edge in $\delta(A, A') - \delta(C, C')$ or $\delta(C, C') - \delta(A, A')$, which will give an endpoint to (A, A') by Lemma 4.2.9. Thus (A, A') gets that one endpoint, and 3 surplus endpoints from (C, C').

We have now dealt with all cases involving degree constraints, so all children of our root can be assumed to be connectivity constraints. Suppose non-leaf $(A, A') \in \mathcal{C}$ has $\alpha \geq 1$ children and owns $\beta \geq 0$ endpoints. If $\alpha + \beta \geq 4$ then (A, A') can collect 4 endpoints. Suppose $\alpha + \beta \leq 3$.

Case α =1: Lemma 4.2.13 implies that $\beta \geq 2$. If $\beta > 2$ then (A, A') can collect 4 endpoints. If $\beta = 2$ and the child has corequirement $\neq \frac{1}{2}$, then (A, A') collects 2 surplus from the child and owns $\beta = 2$ endpoints, thus can collect 4 endpoints. If the child has corequirement $= \frac{1}{2}$, then $\alpha + \beta = 3$, and Lemma 4.2.12 implies that $\textsc{coreq}(A, A') = \frac{1}{2}$, and (A, A') can collect 3 endpoints, 1 surplus from the child, and the 2 that it owns.

Case α =2: If both children have corequirement $\neq \frac{1}{2}$, (A, A') gets 2 surplus endpoints from each to collect 4 endpoints. Otherwise, Lemma 4.2.14 implies that $\beta \geq 1$. If $\beta \geq 2$ then (A, A') can get 1 surplus endpoint from each child and owns at least 2 itself to collect 4 endpoints, so suppose $\beta = 1$. If one child has corequirement $\neq \frac{1}{2}$ then (A, A') collects 2 surplus endpoints from that child, 1 from the other child, and owns $\beta = 1$ to collect 4 endpoints. Otherwise, each child has corequirement of $\frac{1}{2}$. But then $\alpha + \beta = 3$ and Lemma 4.2.12 implies that $\textsc{coreq}(A, A') = \frac{1}{2}$, and (A, A') can collect 3 endpoints: 1 from each child and the $\beta = 1$ it already owns.

Case α =3: If any child has corequirement $\neq \frac{1}{2}$, (A, A') collects 2 surplus endpoints from that child and 1 from every other to get at 4 endpoints. Otherwise Lemma 4.2.12 implies that $\text{COREQ}(A, A') = \frac{1}{2}$ and (A, A') can collect 3 endpoints, taking 1 surplus endpoint from each child.

□

4.2.6 Experimental Results

We had hoped that this $(2, 2B+3)$-approximation for degree bounded element connectivity would also work with degree bounds on nonterminals. However, some new insight would be required to be able to show that this current algorithm would work with bounds on nonterminals. The roadblock is that the degree bound constraints for nonterminals cannot easily be added into our pair-laminar basis $\mathcal{B}$ family of tight constraints. We are able to add any terminal vertex v as a leaf node $(\{v\}, V - \{v\})$ to $\mathcal{B}$ because for any ELC set pair $(A, A') \in \mathcal{B}$, we have that all terminals are contained in $A \cup A'$. This ensures that the set pair $(\{v\}, V - \{v\})$ does not pair-cross with any other set pair in $\mathcal{B}$. As we do not have the same property for nonterminals, these constraints would have to be dealt with in some other way.

We coded Algorithm 3, allowing degree bounds on nonterminals, in an attempt to either build evidence that this algorithm may work with degree bounds on nonterminals, or else to find a basic solution from which the algorithm could not make progress; that is, every edge has value $< \frac{1}{2}$, and every vertex has at least 5 edges incident to it. Such a basic solution would be a counterexample showing that the iterative rounding method will not work when degree bounds are placed on nonterminals.

Graphs were generated for a specified number of vertices n, and each edge was added with probability $\frac{1}{2}$. We then increased the number of edges around any vertices with degree below some minimum threshold. This was done because any vertex incident to a low number of edges would simply remove the degree bound constraint altogether, and we know that any counterexample must make use of degree constraints on nonterminals. Edges were given weights distributed uniformly at random between 1 and some parameter *maxWeight*, normally set to be 5. Degree bounds B_v for any vertex v were assigned to be an integer between 1 and $|\delta(v)/2| - 2$ uniformly at random, where the interval was

chosen so that it was not trivial to satisfy degree bounds within a $2B_v + 3$ factor. Finally, connectivity requirements for vertices u, v were chosen to be between 0 and $\min\{B_u, B_v\}$ uniformly at random.

We generated approximately 1000 feasible instances. All instances had between $n = 8$ and $n = 15$ vertices, as our program would run too slowly with a large number of vertices. Our minimum vertex degree threshold varied between 7 and $n-1$ edges. No counterexample was found, the algorithm was able to solve all instances to find a $(2, 2B + 3)$-approximate solution. However, this could be the result of only testing on graphs with a small number of vertices, as it seems that if there is a counterexample, it would have to occur on a larger graph where degree constraints will have a greater impact. It could also be the case that there is a counterexample on a small number of vertices, but the algorithm just did not generate this instance. However, these experiments do give hope that this algorithm will work with degree bounds on non-terminals, and hence we present the following conjecture.

4.2.15 Conjecture. *If we allow degree constraints on all vertices, $W \subseteq V$, then Algorithm 3 is a $(2, 2B + 3)$-approximation for* ELC-B.

Chapter 5

Prize Collecting Network Design

We begin this section with a review of a 3-approximation algorithm for the Prize- Collecting Steiner Forest problem by Grandoni et al. [26]. We then take a look at the Prize-Collecting Steiner Tree problem, and attempt to develop an efficient 2-approximation that has a certain Lagrangean multiplier preservation property. We are not able to prove such an algorithm in full generality, but we do provide some results about the structure of a basic feasible solution for the problem. We then take a look at a new problem that arises naturally when we manipulate a basic feasible solution of the Prize-Collecting Steiner Tree problem. We call the new problem the Prize-Collecting Generalized Steiner Tree problem, and provide some insights as to the hardness of this problem.

5.1 Prize-Collecting Steiner Forest

In the Prize-Collecting Steiner Forest problem we are given an undirected graph $G = (V, E)$, with edge costs c and a set of k terminal pairs $T = \{(s_1, t_1), ..., (s_k, t_k)\}$ with a penalty for each pair π_i. We wish to find a minimum cost subgraph H of G in which all terminal pairs are connected.

Here we present a 3-approximation algorithm for the PC-SF problem, due to Grandoni et al. [26]. This algorithm uses iterative rounding, which is justified through an endpoint counting argument. We simplify the proof by using a technique we call *merging columns* to

take care of shared cuts, an idea seen by Hajiaghayi et al. in [30]. We were hoping to find a fractional token argument for this problem, but this is difficult to achieve because of the variables needed to denote whether or not a pair is connected. However, this motivated us to look at the simpler Prize-Collecting Steiner Tree problem, and we take a look at that problem in the next section.

The PC-SF problem can be formulated with the following integer program:

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) + \sum_{i=1}^{k} \pi(i)z(i) && && \text{(IP-SF')} \\ \text{s.t.} \quad & x(A) + z(i) \geq 1 && \forall (s_i, t_i) \in T, A \subset V : |A \cap \{s_i, t_i\}| = 1 \\ & x(e), z(i) \in \{0, 1\} && \forall e \in E, i \in \{1, ..., k\} \end{aligned}$$

We may relax this to a linear program (LP-SF') by relaxing the last constraint to $x(e), z(i) \geq 0$ for all $e \in E, i \in \{1, ..., k\}$. The 3-approximation is based on the following lemma.

5.1.1 Lemma ([26]). *In any basic feasible solution (x, z) to $(LP - SF')$, for at least one edge e, $x(e) \geq \frac{1}{3}$, or for at least one $i = 1, ..., k$, $z(i) \geq \frac{1}{3}$.*

We can now present the 3-approximation for PC-SF, Algorithm 4.

Algorithm 4 Iterative Rounding 3-Approximation for PC-SF

1: $F \leftarrow \emptyset$
2: **while** (There are non-trivial terminal pairs) **do**
3: Solve (LP-SF'), let (x, z) be an optimal basic solution
4: Pay penalties for pairs $i \in 1, ..., k$ where $z(i) \geq \frac{1}{3}$ and remove this terminal pair from future consideration
5: For any edge e where $x(e) \geq \frac{1}{3}$, add e to F and contract the edge in G
6: **end while**

Using the same inductive proof as seen in Jain's approximation algorithm for SNDP (see Section 1.4), we get a 3-approximation for PC-SF, provided that Lemma 5.1.1 holds.

We now work towards showing Lemma 5.1.1.

Let (x, z) be a basic feasible solution. We may assume $x(e) < \frac{1}{3}$, $z(i) < \frac{1}{3}$ for all $e \in E, i = 1, ..., k$, as otherwise the lemma follows. If $x(e) = 0$ for some edge, we can remove this edge, and (x, z) is still a basic feasible solution for the resulting problem. We thus have $0 < x(e) < \frac{1}{3}$ for every edge e, and $0 \leq z(i) < \frac{1}{3}$ for every $i = 1, ..., k$. We may also assume that the graph is connected, as otherwise we could split the problem into subproblems for each connected component, and sove each separately.

Note that in this setting, our constraints are defined based on a set $A \subset V$ and a terminal pair (s_i, t_i). We let $[A, i]$ be the $0, 1$ vector over both edges and i defining terminal pairs, such that $[A, i]$ that represents the left hand side of this constraint. That is $[A, i](e) = 1$ if $e \in \delta(A)$ and 0 for any other edge, $[A, i](j) = 0$ if $j \neq i$ and $[A, i](i) = 1$ if this is a non-pure constraint, 0 otherwise. As (x, z) is a basic feasible solution, there exists a basis of tight constraints that define (x, z). We say that a family of constraints is *laminar* if the corresponding sets defining these constraints form a laminar family.

5.1.2 Lemma ([26])**.** *There is a laminar basis $\mathcal{B}$ of tight constraints defining (x, z), with $|\mathcal{B}| = |E| + k'$, where k' is the number of terminal pairs with nonzero z value.*

Proof. Let $\mathcal{T}$ be the set of tight constraints, and consider a maximal laminar subset $\mathcal{L}$ of $\mathcal{T}$. Suppose $\textsc{Span}(\mathcal{L}) \subsetneq \textsc{Span}(\mathcal{T})$. We define the number of crossings of a set B to be $\textsc{cross}(B) = |\{A \in \mathcal{L} : A \text{ crosses } B\}|$.

Choose a vector $[B, j] \in \mathcal{T} - \textsc{Span}(\mathcal{L})$ minimizing $\textsc{cross}(B)$. By maximality of $\mathcal{L}$, $\textsc{cross}(B) \geq 1$. Let $A \in \mathcal{L}$ one such set that crosses B. Notice that by Lemma 2.3.3, each of $A - B$, $B - A$, $A \cap B$, $A \cup B$ have strictly smaller crossing number than B, and hence are all in $\textsc{Span}(\mathcal{L})$. If we can write B as a linear combination of these four sets, we then have that $B \in \textsc{Span}(\mathcal{L})$, a contradiction.

Note that the following two equations hold:

$$x(A) + x(B) = x(A \cap B) + x(A \cup B) + 2x(A - B, B - A)$$

$$x(A) + x(B) = x(A - B) + x(B - A) + 2x(A \cap B, \overline{A \cup B})$$

We distinguish a few subcases:

Case 1: $A - B$ separates $a \in \{i, j\}$, $B - A$ separates $b \in \{i, j\}$, $a \neq b$. Then

$$\begin{aligned} 2 - z(i) - z(j) &\leq x(A - B) + x(B - A) \\ &= x(A) + x(B) - 2x(A \cap B, \overline{A \cup B}) \\ &\leq x(A) + x(B) \\ &= 2 - z(i) - z(j) \end{aligned}$$

Thus [A-B,a] and [B-A,b] are tight. Moreover, $x(A \cap B, \overline{A \cup B}) = 0$ implying that $\delta(A \cap B, \overline{A \cup B}) = \emptyset$, thus

$$[B, j] = [A - B, a] + [B - A, b] - [A, i].$$

Case 2: $A \cup B$ separates $a \in \{i, j\}$ and $A \cap B$ separates $b \in \{i, j\}, b \neq a$. Then

$$\begin{aligned} 2 - z(i) - z(j) &\leq x(A \cup B) + x(B \cap A) \\ &= x(A) + x(B) - 2x(A - B, B - A) \\ &\leq x(A) + x(B) \\ &= 2 - z(i) - z(j) \end{aligned}$$

By the same argument as in case 1,

$$[B, j] = [A \cup B, a] + [B \cap A, b] - [A, i].$$

Case 3: We may assume without loss of generality that $s_i \in A - B$, $t_i \in B - A$, $s_j \in A \cap B$, and $t_j \in \overline{A \cup B}$.

Note that A and B separate both i and j. Thus $x(A) + z(j) \geq 1 = x(A) + z(i)$ and $x(B) + z(i) \geq x(B) + z(j)$, implying $z_i = z_j$. Call the common value z for simplicity. Then,

$$\begin{aligned} 4 - 4z &\leq x(A - B) + x(B - A) + x(A \cup B) + x(B \cap A) \\ &= 2x(A) + 2x(B) - 2(x(A - B, B - A) + x(A \cap B, \overline{A \cup B})) \\ &\leq 2x(A) + 2x(B) \\ &= 4 - 4z \end{aligned}$$

This implies that $[A-B,i],[B-A,i],[A\cup B,j]$ and $[A\cap B,j]$ are all tight, and there are no edges in $\delta(A-B,B-A)$ or $\delta(A\cap B,\overline{A\cup B})$. Thus

$$2[A,i]+2[B,j]=[A\cup B,j]+[A\cap B,j]+[A-B,i]+[B-A,i],$$

as desired.

Thus we have that $\textsc{Span}(\mathcal{T})=\textsc{Span}(\mathcal{L})$, and we know $\textsc{Span}(\mathcal{T})$ has dimension $|E|+k'$. Let $\mathcal{B}$ be a maximal linearly independent subset of $\mathcal{L}$. Then $\mathcal{B}$ is the basis we desire, and $|\mathcal{B}|=|E|+k'$. □

Let $\mathcal{B}$ be a laminar basis of tight constraints defining (x,z). We say that a set $A\in\mathcal{B}$ if there is a constraint $[A,a]\in\mathcal{B}$ for some a. Note that some sets may appear in $\mathcal{B}$ more than once, of course with different terminal pairs. We call such a set a *shared cut*. Note that if A is a shared cut with terminal pairs a and b, then $z(a)=z(b)$. This basis will be much simpler to deal with if we can find a way to shared cuts. We now define a procedure to deal with these shared cuts. This procedure is used in Hajiaghayi et al.'s iterative rounding algorithm for this problem [30], but they do not justify the lemma that follows in their paper.

Let $S\in\mathcal{B}$ be a shared cut with $a\neq b$ and $[S,a],[S,b]\in\mathcal{B}$. Consider the following procedure, which we call a *column merge* on colums. Suppose we replace every instance of variables $z(a)$ and $z(b)$ in our problem with a new variable $z(ab)$, and set $z(ab)=z(a)=z(b)$. We call this a *column merge* on variables a and b of our problem, because we are adding together these columns in our constraint matrix to form a new column.

5.1.3 Lemma. *Consider a shared cut S with $[S,a],[S,b]\in\mathcal{B}$, $b\neq a$. Apply a column merge to variables a and b. Remove one of the duplicate $[S,ab]$ constraints from $\mathcal{B}$ so that this constraint now only appears once. Then this new x,z is a basic solution in the new problem with basis $\mathcal{B}$.*

Proof. Suppose we have $[S,a],[S,b]$ both in $\mathcal{B}$. We want to show that after performing the column merge, our new constraints are still linearly independent. Let $\mathcal{B}=\{\chi_0=[S,a],\chi_1=[S,b],\chi_2,...\chi_s\}$ be our original basis for x,z. We remove χ_0 from our system, and add columns a and b together, call the new variable ab, to get our new system $\mathcal{B}'=$

$\{\chi'_1 = [S, ab], \chi_2, ...\chi_s\}$. Suppose this new system is linearly dependent, that is, there exist α_i, $i = 1, ..., s$ not all zero such that

$$\sum_{i=1}^{s} \alpha_i \chi'_i = 0$$

Note that this implies $\sum_{i=1}^{s} \alpha_i \chi_i(y) = 0$ for every variable $y \neq ab$.

For variable ab, we have

$$\begin{aligned} 0 &= \sum_{i=1}^{s} \alpha_i \chi'_i(ab) \\ &= \sum_{i=1}^{s} \alpha_i \chi_i(a) + \sum_{i=1}^{s} \alpha_i \chi_i(b) \end{aligned}$$

We cannot have $\sum_{i=1}^{s} \alpha_i \chi_i(a) = 0 = \sum_{i=1}^{s} \alpha_i \chi_i(b)$ as otherwise $\sum_{i=1}^{s} \alpha_i \chi_i = 0$, contradicting that $\mathcal{B}$ is a basis.

Thus $\sum_{i=1}^{s} \alpha_i \chi_i(a) = -\sum_{i=1}^{s} \alpha_i \chi_i(b) \neq 0$. Hence we can scale our α_i such that

$$\begin{aligned} \sum_{i=1}^{s} \alpha_i \chi_i(a) &= 1 \\ \sum_{i=1}^{s} \alpha_i \chi_i(b) &= -1 \\ \sum_{i=1}^{s} \alpha_i \chi_i(y) &= 0 \text{ for all } y \notin \{a, b\} \end{aligned}$$

But this implies that $\chi_0 - \chi_1 = [S, a] - [S, b]$ is in $\textsc{Span}\{\chi_1, \chi_2, ..., \chi_s\}$, violating linear independence of $\mathcal{B}$. □

5.1.4 Corollary. *Through iterated application of Lemma 5.1.3, we may assume there are no shared cuts in $\mathcal{B}$.*

This reduction gives that each set in $\mathcal{B}$ is tight with only one pair. Thus a tight set uniquely determines a tight pair, and we can now use the terms tight set and tight constraint interchangeably. Let $F_{\mathcal{B}}$ be the induced forest on $\mathcal{B}$ (see Lemma 2.3.4). We will need to use the fact that every node in our laminar family induced a connected subgraph, and we show this with the following lemma.

5.1.5 Lemma ([26]). *Let $A \in \mathcal{B}$. Then A induces a connected component.*

Proof. Suppose not. Let C_1 and C_2 be two non-empty sets that partition A and have no edges between them. Note that $x(A) = x(C_1) + x(C_2)$. Suppose A is tight for terminal $s_i \in A$, $t_i \notin A$. Suppose without loss of generality that $s_i \in C_1$. Then $x(A) + z(a) = x(C_1) + x(C_2) + z(a) = 1$ and $x(C_1) + z(a) \geq 1$, thus $x(C_2) = 0$.

Hence C_2 has no edges coming out of it, implying that our graph is not connected, a contradiction. □

We now use an endpoint argument to contradict that $0 < x(e) < \frac{1}{3}$ for every edge $e \in E$ and $z(i) < \frac{1}{3}$ for every terminal pair i.

We assign each endpoint u of edge $e = uv$, to the lowest set in our laminar forest that contains u. These endpoints are then redistributed according to the following rule: starting from the leaves, each set A keeps 2 endpoints and sends any other endpoints up to its parent, of which there are at least one. This will then show that every set gets two endpoints, and some are remaining, implying that —$\mathcal{B}$— ¡ —E—, contradicting Lemma 5.1.2.

Let us show by induction that each set A sends $\deg(A) - 2 > 0$ endpoints to its parent. A leaf set A receives $\deg(A)$ endpoints. Note that $\deg(A) \geq 3$, since otherwise there would be a variable of value at least $\frac{1}{3}$. Hence A keeps 2 endpoints, and sends $\deg(A) - 2 > 0$ to its parent.

Consider an internal node A. Let $A_1, ..., A_h$ be its children, and $A' = A - (\bigcup_{i=1}^{k} A_i)$ be all remaining vertices in A. We let $\text{out}(A_i)$ be the number of edges crossing A_i and A. Similarly, we let $\text{out}(v)$ be the number of edges crossing A that are incident to $v \in A'$. Let m' be the number of edges with both endpoints in A, but not both endpoints in some single child A_i. Then

$$\sum_{i=1}^{h} \deg(A_i) + \sum_{v \in A'} \deg(v) = \sum_{i=1}^{h} \text{out}(A_i) + \sum_{v \in A'} \text{out}(v) + 2m'.$$

Note that $\deg(A) = \sum_{i=1}^{h} \text{out}(A_i) + \sum_{v \in A'} \text{out}(v)$. Hence we get that A receives

$$\sum_{i=1}^{h} (\deg(A_i) - 2) + \sum_{v \in A'} \deg(v) = \deg(A) + 2m' - 2h$$

endpoints. Recall that A requires $\deg(A)$ tokens, so that it can keep 2 and pass on $\deg(A)-2$. As A induces a connected component, $m' \geq |A'| + h - 1 \geq h - 1$. If $m' \geq h$ then we have enough endpoints and we are done.

Suppose $m' = h - 1$. Thus $A' = \emptyset$, and the A_i's partition A. Moreover, A_i's induce a tree T' with one edge between each pair of adjacent children. Note that we cannot have $h = 1$ as otherwise $A_1 = A$, so we know $h \geq 2$. Thus T' contains at least two leaves, say A_1 and A_2. Say A is tight with pair (s_a, t_a) with $s_a \in A, t_a \notin A$. Then at least one of A_1 or A_2 does not contain s_a, say $s_a \notin A_1$. Say A_1 is tight with (s_1, t_1) and $s_1 \in A_1, t_1 \notin A_1$.

We have $x(A) + z(a) = 1 = x(A_1) + z(a_1)$. Let e be the unique edge between A_1 and $A - A_1$. Note that $x(A) = (x(A_1) - x(e)) + (x(A - A_1) - x(e))$. Furthermore, as $A - A_1$ separates (s_a, t_a), $x(A - A_1) + z(a) \geq 1$.

Putting this together, we get

$$\begin{aligned} 2x(e) + z(b) &= [x(A_1) + x(A - A_1) - x(A)] + [1 - x(A_1)] \\ &= x(A - A_1) + [1 - x(A)] \\ &= x(A - A_1) + z(a) \\ &\geq 1 \end{aligned}$$

This implies that either $x(e) \geq \frac{1}{3}$ or $z(b) \geq \frac{1}{3}$, a contradiction.

5.2 Prize-Collecting Steiner Tree

In this section we consider the Prize-Collecting Steiner Tree (PC-ST) problem, which is a special case of the Prize-Collecting Steiner Forest problem in which terminal pair shares some common vertex. We attempt to prove an iterative rounding approximation algorithm using our generalized fractional token argument. We are not able to prove such an algorithm, but we do provide some interesting structural results about basic feasible solutions to the standard LP for this problem.

In the Steiner Tree problem, we have a special vertex r called the *root*, and we have a set of terminals R that we wish to connect to the root. We are allowed to leave some terminal vertices $v \in R$ disconnected from the root, but for every such vertex we pay a price $\pi(v)$.

PC-ST can be naturally formulated with the following integer linear program (IP-ST):

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) + \sum_{v \in R} \pi(v)z(v) && \text{(IP-ST)} \\ \text{s.t.} \quad & x(A) + z(v) \geq 1 && \forall A \subset V - \{r\}, v \in R : v \in A \\ & x(e), z(v) \in \{0,1\} && \forall e \in E, v \in R \end{aligned}$$

The interpretation is that $x(e) = 1 \iff e \in T$, and $z(v) = 0 \iff v$ is connected to r in T. Let (LP-ST) be the linear relaxation of (IP-ST), where we replace the last constraint of (IP-ST) with

$$0 \leq x(e), z(v) \leq 1, \forall e \in E, v \in R.$$

Note that, as long as the optimal fractional solution contains a variable $x(e) \in \{0,1\}$ or $z(v) = 1$, we can reduce the problem and iterate. In particular, setting $x(e)$ to 0 is equivalent to removing edge e, setting $x(e)$ to 1 is equivalent to contracting edge e, and setting $z(v)$ to 1 is equivalent to removing v from the set of terminals R. Unfortunately, the problem resulting from setting some $z(v)$ to 0 is no longer a standard PC-ST problem. To maintain a PC-ST problem, we will not remove variable $z(v)$ when $z(v) = 0$, but we will have to find some other variable that we can round in order to make progress. Our algorithm hinges on the following key conjecture, which we have not yet been able to prove in the general case.

5.2.1 Conjecture. *Any optimal basic solution to (LP-ST) contains either an edge e where $x(e) = 0$, an edge e where $x(e) \geq \frac{1}{2}$, or a terminal v where $z(v) = 1$.*

With such a property, we would be able to design a *Lagrangean multiplier preserving* approximation algorithm for this problem. We say an approximation algorithm for a prize collecting problem is *Lagrangian multiplier preserving* if the output solution (x, z) satisfies $c^T x + \alpha \cdot \pi(z) \leq \alpha \cdot (c^T x^* + \pi(z^*))$ for any optimal solution (x^*, z^*).

Such an algorithm would allow us to construct an algorithm for the k-Steiner Tree problem, which is the problem where one wants to connect at least k terminals to the root (and no penalties are paid for other non-connected terminals) using the methods in Chudak et al. [11]. These authors show how Goemans and Williamson's 2-approximate Lagrangean multiplier preserving algorithm for PC-ST[25] can be translated into a 5-approximation

algorithm for the k-Steiner Tree problem, which is the problem where one wants to connect at least k terminals to the root (and no penalties are paid for other non-connected terminals). The same could be done with this iterative rounding algorithm if we can prove that it is indeed a 2-Lagrangean multiplier preserving approximation algorithm.

Assuming Conjecture 5.2.1, consider the following natural iterative rounding algorithm. Compute an optimal basic solution (x, z) to (LP-ST). If there is an edge e with $x(e) = 0$, remove the edge and iterate on the reduced problem. If there is a terminal v with $z(v) = 1$, we pay that penalty, and remove v from the set of terminals, and then iterate on the reduced problem. Otherwise, take an edge e with $x(e) \geq \frac{1}{2}$, round it up to 1, reduce the problem and iterate. As we round up edge costs on edges that are at most $\frac{1}{2}$, and we only pay penalties when they are paid by the optimal fractional solution, we get the following theorem by standard inductive arguments.

5.2.2 Theorem. *Given Conjecture 5.2.1 algorithm above is a Lagrangean multiplier preserving 2-approximation for* PC-ST.

5.2.1 Preliminaries

For sets of vertices $A, B \subseteq V$, we define $A \times B = \{e \in E : e \in \delta(A) \cap \delta(B)\}$. We say a set $A \subseteq V - \{r\}$ separates a (from the root r) if $a \in A$.

Consider an optimal fractional basic solution (x, z) to (LP-ST). We formulate a new LP in which we do not have any variables $z(v)$ where $z(v) = 0$. Define $R^0 = \{v \in R : z(v) = 0\}$ and $R^+ = R - R^0$. Call our new LP (LP-ST'), and define it as follows:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c(e)x(e) + \sum_{v \in R^+} \pi(v)z(v) && && \text{(LP-ST')} \\
\text{s.t.} \quad & x(A) + z(v) \geq 1 && \forall A \subset V - \{r\}, v \in R^+ : v \in A && (5.1) \\
& x(A) \geq 1 && \forall A \subset V - \{r\}, v \in R^0 : v \in A && (5.2) \\
& 0 \leq x(e), z(v) \leq 1 && \forall e \in E, v \in R^+ &&
\end{aligned}$$

Note that (x, z) is still an optimal basic solution to (LP-ST'). Further, we only have variables $z(v)$ if $z(v) > 0$. We now have two different types of constraints. We call

a constraint of type (5.2) a *pure constraint*, and a constraint of type (5.1) a *non-pure constraint*. We denote the characteristic vector of the constraint for set $A \subseteq V - \{r\}$ and terminal $v \in A$ by $[A, a]$, and this can represent either a pure or a non-pure constraint. As (x, z) is a basic solution, there is a basis of linear independent tight constraints $\mathcal{B}$ that defines (x, z).

If $[A, a] \in \mathcal{B}$, for convenience we say $A \in \mathcal{B}$. For a given set $A \in \mathcal{B}$, there may be multiple constraints of type $[A, a]$ in $\mathcal{B}$. We call the number of occurrences of A in $\mathcal{B}$ the *multiplicity* of A, and denote this by $m(A)$, and if $m(A) \geq 2$ we say A is a shared cut. Note that $m(A) = 1$ if $[A, a] \in \mathcal{B}$ for some $a \in R^0$, as shown by the following lemma.

5.2.3 Lemma. *If $[A, a] \in \mathcal{B}$ for some $a \in R^0$, then $[A, b] \notin \mathcal{B}$ for any $b \in R - \{a\}$.*

Proof. Suppose $[A, b] \in \mathcal{B}$. If $b \in R^0$, then $[A, a] = [A, b]$, contradicting that $\mathcal{B}$ is linearly independent. If $b \in R^+$, then $x(A) + z(a) = 1 = x(A)$ contradicts that $z(b) > 0$. □

5.2.2 Laminar Basis

Let $\mathcal{T}$ be the set of all tight constraints for (x, z) in (LP-ST'). For a family of tight constraints $\mathcal{F} \subseteq \mathcal{T}$, we define $\textsc{Span}(\mathcal{F})$ to be the vector space spanned by vectors $[A, a] \in \mathcal{F}$. We say a family of constraints $\mathcal{F}$ is *laminar* if the family of sets $A \in \mathcal{F}$ is laminar. We will now show that there exists a laminar basis for (x, z) in (LP-ST').

5.2.4 Theorem. *There is a laminar basis for (x, z) in (LP-ST').*

Proof. Consider any maximal laminar subset $\mathcal{L}$ of tight constraints. Assume, for purpose of contradiction, that $\textsc{Span}(\mathcal{L}) \subsetneq \textsc{Span}(\mathcal{T})$. Define the number of crossings of a set $A \in \mathcal{T}$ to be $\textsc{cross}(B) = |\{A \in \mathcal{L} : A \text{ crosses } B\}|$. Choose a vector $[B, b] \in \mathcal{T} - \textsc{Span}(\mathcal{L})$ minimizing $\textsc{cross}(B)$. By maximality of $\mathcal{L}$, we have $\textsc{cross}(B) \geq 1$, otherwise we could add $[B, b]$ to $\mathcal{L}$. Let $[A, a] \in \mathcal{L}$ be a constraint such that A crosses B and A. Note that we could have $a \in R^0$ or $b \in R^0$. If this is the case, say with a, we will want to refer to variable $z(a)$ for convenience; just set $z(a) = 0$. We will show that $[B, b]$ can be expressed as a linear combination of tight vectors, all having a strictly smaller crossing number than B. By assumption, all sets with smaller crossing number are in $\textsc{Span}(\mathcal{L})$, implying that

$[B, b]$ is also in $\textsc{Span}(\mathcal{L})$, a contradiction. We now show how to uncross A and B. Note that following two equations hold by a simple counting argument:

$$x(A) + x(B) = x(A \cap B) + x(A \cup B) + 2x((A - B) \times (B - A))$$
$$x(A) + x(B) = x(A - B) + x(B - A) + 2x((A \cap B) \times \overline{(A \cup B)})$$

We distinguish a few subcases based on the locations of a and b, up to symmetry:

Case 1: $A \cap B \cap \{a, b\} = \emptyset$.

Then $A - B$ separates a and $B - A$ separates b. Thus $x(A - B) \geq 1 - z(a)$ and $x(B - A) \geq 1 - z(b)$, by feasibility of (x, z).

$$\begin{aligned}(1 - z(a)) + (1 - z(b)) &\leq x(A - B) + x(B - A) \\ &= x(A) + x(B) - 2x((A \cap B) \times \overline{(A \cup B)}) \\ &\leq x(A) + x(B) \\ &= (1 - z(a)) + (1 - z(b))\end{aligned}$$

Thus we actually have equality on all lines above, so we have that $[A - B, a]$ and $[B - A, b]$ are tight. Further, we have $x((A \cap B) \times \overline{(A \cup B)}) = 0$ meaning that there are no edges in $(A \cap B) \times \overline{(A \cup B)}$. Note that any set in $\mathcal{L}$ that crosses $A - B$ also crosses B by Lemma 2.3.3, but $A - B$ does not cross A, so $\text{cross}(A - B) < \text{cross}(B)$. Similarly, $B - A$ has strictly smaller crossing number than B, giving us the desired contradiction:

$$[B, b] = [A - B, a] + [B - A, b] - [A, a].$$

Case 2: $A \cap B \cap \{a, b\} \neq \emptyset$. Say, without loss of generalization, $a \in A \cap B$. Then $A \cap B$ separates a and $A \cup B$ separates b. Thus $x(A \cap B) \geq 1 - z(a)$ and $x(B \cup A) \geq 1 - z(b)$.

$$\begin{aligned}(1 - z(a)) + (1 - z(b)) &\leq x(A \cap B) + x(B \cup A) \\ &= x(A) + x(B) - 2x((A - B) \times (B - A)) \\ &\leq x(A) + x(B) \\ &= (1 - z(a)) + (1 - z(b))\end{aligned}$$

Thus we actually have equality on all lines above, so we have that $[A \cap B, a]$ and $[B \cup A, b]$ are tight. Further, we have $x((A - B) \times (B - A)) = 0$ meaning that there

are no edges in $(A-B)\times(B-A)$. As both $A\cap B$ and $A\cup B$ have strictly smaller crossing number than B, by Lemma 2.3.3, we get the desired contradiction:

$$[B,b] = [A\cap B, a] + [B\cup A, b] - [A,a].$$

Thus we have $\textsc{Span}(\mathcal{L}) = \textsc{Span}(\mathcal{T})$. Taking a maximal linearly independent set from $\mathcal{L}$ gives us the basis we desire. □

5.2.3 Maximality

We have thus shown that there is a laminar basis $\mathcal{L}$ for (x,z) in (LP-ST'). Let $F_{\mathcal{L}}$ be the unique forest induced by distinct sets in $\mathcal{L}$ by Lemma 2.3.4.

First notice that as we descend a tree in our forest, the z-values are non-decreasing. Formally, suppose $[A,a],[B,b] \in \mathcal{L}$ and $B \subseteq A$. Then A separates b, so by feasibility $x(A)+z(b)\geq 1$. Since $x(A)+z(a)=1$, we get $z(b)\geq z(a)$. This implies that all pure cuts are going to occur at the top of our forest $F_{\mathcal{L}}$.

We now wish to get further control over the structure of our basis, namely, we would like sets with high multiplicity to appear as high as possible in our forest.

5.2.5 Definition. *The level of a set $A\in\mathcal{L}$ in $F_{\mathcal{L}}$ is the number of edges on the unique path from the root to A in $F_{\mathcal{L}}$.*

5.2.6 Definition. *The* multiplicity vector *of a laminar basis $\mathcal{L}$ is the vector $(k_0,...,k_p)$ where k_i is the sum of the multiplicities of all tight sets in $\mathcal{L}$ at level i in our forest $F_{\mathcal{L}}$. We say that a laminar basis $\mathcal{L}$ is* maximal *if it maximizes the multiplicity vector in the lexicographical sense.*

We will begin with a maximal laminar basis, and will use it to get a basis that is continuous in the sense that terminals will appear in paths in the forest induced by the laminar family.

5.2.4 Continuity

Note that for any terminal a, all tight constraints of the type $[A,a]\in\mathcal{L}$ for some set A form a chain in our laminar family. Otherwise there would be two disjoint sets containing

a. However, in our forest $F_{\mathcal{L}}$, this chain may not necessarily be continuous with respect to the terminal a, as there could be constraints tight with other terminals in between those constraints in our chain. Formally, we define a *discontinuity* for terminal $a \in R^+$ to be a triple $[C, a] \in \mathcal{L}$, $[A, a] \in \mathcal{L}$, and $B \in \mathcal{L}$, where $C \subsetneq B \subsetneq A$ and $[B, a] \notin \mathcal{L}$. We say a laminar basis is *continuous* if it induces no discontinuity. We say that a constraint $[A, a]$ is *valid* if $a \in A$.

We would like to show that we can construct a laminar basis that is continuous. We will process a maximal laminar basis to construct a continuous one. Throughout the following proof, we will be looking at constraints $[A, a], [B, b] \in \mathcal{L}$ and modifying them to get the structure we desire. A *terminal swap* to a on $[B, b]$ means that we remove $[B, b]$ from $\mathcal{L}$ and replace it with $[B, a]$ to get a new laminar basis $\mathcal{L}'$. In order to perform such a terminal swap, we will need to show that the new constraint $[B, a]$ is tight, and the span of all constraints has not reduced. This latter requirement is equivalent to showing that $[B, b]$ is in the span of $\mathcal{L}'$. Note that any such swap is reversible, we could replace $[B, a]$ with $[B, b]$ to get our original laminar basis $\mathcal{L}$. Note that a terminal swap does not affect the multiplicity vector for the laminar basis.

Another type of swap that we will consider is a *set swap*. In this case, a set swap to B on $[A, a]$ would remove $[A, a]$ from $\mathcal{L}$ and replace it with $[B, a]$. For our purposes, we will only consider set swaps when both $A, B \in \mathcal{L}$. As before, for this to maintain a tight basis, we need to show that $[B, a]$ is tight and that $[A, a]$ is in the span of the new family of constraints. If $A \subset B$, we say that this is an *increasing* set swap and if $B \subset A$, and A is a shared cut in $\mathcal{L}$, we say it is a *decreasing* set swap. This terminology is consistent in the sense that an *increasing* set swap increases our multiplicity vector and a *decreasing* set swap decreases our multiplicity vector. Note that we do require for A to be a shared cut in $\mathcal{L}$ for a decreasing set swap to necessarily decrease the multiplicity vector. Importantly, note that any decreasing set swap is reversible, and the reverse swap in an increasing set swap. That is, we can always reverse a decreasing set swap to return to a laminar basis with a higher multiplicity vector.

In the following procedure, we begin with a maximal laminar basis $\mathcal{L}$, and perform terminal swaps and decreasing set swaps to convert this into a continuous basis. If at any step we are able to perform an increasing set swap to B on some constraint $[A, a]$, where $A \subset B$, then we get a contradiction: we can reverse all terminal swaps and decreasing set swaps, and since $B \in \mathcal{L}$, we now have a laminar basis with higher multiplicity vector than

$\mathcal{L}$. Note that indeed $B \in \mathcal{L}$, as set swaps do not introduce any new sets into $\mathcal{L}$. Thus at every stage in this algorithm, we are not able to perform an increasing set swap, and this fact will be used in our proof. We start with a helper lemma that essentially builds our procedure.

5.2.7 Lemma. *Let $\mathcal{L}$ be a maximal laminar basis. Let $\mathcal{L}'$ be a laminar basis obtained from $\mathcal{L}$ through performing terminal swaps and decreasing set swaps. Let C be the root of $F_{\mathcal{L}}$. C may appear with multiple terminals, let a be the one that appears lowest in $F_{\mathcal{L}'}$, and suppose $a \in R^+$. Furthermore, suppose there is a discontinuity for a in $\mathcal{L}'$. Let $[A, a]$ be the highest constraint in which a appears and such that below $[A, a]$, there is no further discontinuity for a. Let $[B, b]$ be a parent of $\mathcal{L}$, $b \neq a$. Then we show two properties:*

1. *We may perform a terminal swap to a on $[B, b]$, maintaining a laminar basis, and*
2. *B is not a shared cut in $\mathcal{L}'$.*

Proof. First note that C separates b and B separates a, thus $z(a) = z(b)$ and $x(A) = x(B) = x(C)$. Thus constraints $[A, b], [B, a]$, and $[C, b]$ are all tight constraints. Note that $a \in B$, and $b \in C$.

However, $[A, b] \notin \mathcal{L}'$, as otherwise we could perform an increasing set swap to B on $[A, a]$, since $[A, a] = [B, a] - [B, b] + [A, b]$, which is a contradiction.

Similarly, $[B, a] \notin \mathcal{L}'$, as otherwise we could perform an increasing set swap to C on $[B, b]$, since $[B, b] = [C, b] - [C, a] + [B, a]$.

List all the constraints in $\mathcal{L}'$ as $\chi_1 = [B, b], \chi_2 = [A, a], \chi_3, ..., \chi_p$. As $\mathcal{L}'$ is a basis, we know that there are coefficients α_i, β_i for $i = 1, ..., p$, each set being not all zero, such that $[A, b] = \sum_{i=1}^{p} \alpha_i \chi_i$ and $[B, a] = \sum_{i=1}^{p} \beta_i \chi_i$.

Observe that $[A, a]+[B, b] = [A, b]+[B, a] = \sum_{i=1}^{p}(\alpha_i+\beta_i)\chi_i$. Note that if $(\alpha_1+\beta_1) \neq 1$ then we could write $\chi_1 = [B, b]$ as a linear combination of χ_i for $i = 2, ..., p$, which violates that the constraints in our basis are linearly independent. Thus $(\alpha_1 + \beta_1) = 1$.

First suppose $\beta_1 = 0$, thus $\alpha_1 = 1$. Then $[B, b] = \frac{1}{\alpha_1}([A, b] - \sum_{i=2}^{p} \alpha_i \chi_i) = [A, b] - \sum_{i=2}^{p} \alpha_i \chi_i$. Thus

$$\begin{aligned}
[C, b] &= [C, a] - [A, a] + [A, b] \\
&= [C, a] - [A, a] + [B, b] + \sum_{i=2}^{p} \alpha_i \chi_i
\end{aligned}$$

Thus we could perform an increasing set swap to C on $[B, b]$, but this is a contradiction. Hence it must be that $\beta_1 \neq 0$.

Thus, we get that $[B, b] = \frac{1}{\beta_1}([B, a] - \sum_{i=2}^{p} \beta_i \chi_i)$, and we may perform a terminal swap to a on $[B, b]$.

We have shown property 1 for our lemma. Let us perform this terminal swap to a on $[B, b]$. If B is a shared cut, then there is some $[B, d] \in \mathcal{L}'$, $d \neq a$. But then $[B, a] = [C, a] - [C, d] + [B, d]$, thus we may perform an increasing swap to C on $[B, a]$, a contradiction. Hence B is not a shared cut in $\mathcal{L}'$. □

We now provide our procedure for creating a continuous laminar basis.

5.2.8 Lemma. *There is a laminar basis of tight valid constraints $\mathcal{L}$ for (x, z) in (LP-ST') that is continuous.*

Proof. Start with a maximal laminar basis of tight constraints, $\mathcal{L}$. We will process $\mathcal{L}$ to construct a continuous laminar basis of tight constraints, $\mathcal{L}'$. We will be removing constraints from $\mathcal{L}$ and adding constraints to $\mathcal{L}'$, and will always maintain that $\mathcal{L} \cup \mathcal{L}'$ is a feasible basis, and $\mathcal{L}'$ is continuous. Start with $\mathcal{L}' = \emptyset$.

If $\mathcal{L} = \emptyset$, then we are done. Also, we may immediately process any node $[C, a]$ that is a pure cut, that is, $a \in R^0$, as these cuts must appear at the top of our forest, and do not induce any discontinuity. By process, we mean remove this constraint from $\mathcal{L}$ and add it to $\mathcal{L}'$. We now assume all such nodes are dealt with.

Start with a root of the forest $F_{\mathcal{L}}$, call it C. C may appear in $\mathcal{L}$ with multiple terminals, let a be the one that appears lowest in our forest $F_{\mathcal{L}}$. Let $[A, a]$ be the lowest constraint in $\mathcal{L}$ in which a appears, and let $\mathcal{C}$ represent every constraint in $\mathcal{L}$ defined for a set B with $A \subsetneq B \subsetneq C$.

By iterative application of Lemma 5.2.7, we may perform a terminal swap to a on every $[B, b] \in \mathcal{C}$, and no such B is a shared cut (see Figure 5.1). We have now ensured there is no discontinuity in a in $\mathcal{L}$.

If C is a shared cut, we need to do a bit more processing. Suppose $[C, d] \in \mathcal{L}$ for some $d \neq a$, and let D be the lowest set in $F_{\mathcal{L}}$ such that $[D, d] \in \mathcal{L}$. Note that $[D, d] \notin \mathcal{C}$, as this would imply $d = a$. Note also that $z(a) = z(d)$. Choose the lowest common ancestor $[B, a] \in \mathcal{C}$ of both $[A, a]$ and $[D, d]$. We perform a decreasing set swap to B on $[C, d]$, which

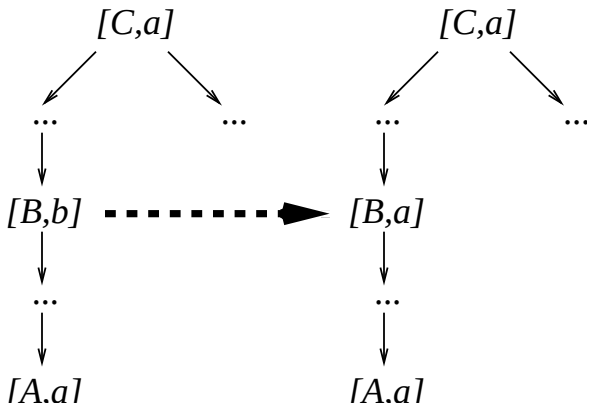

Figure 5.1: A depiction of the basic swap to remove discontinuity from a

maintains a laminar basis as $[B, d]$ is tight, and $[C, d] = [C, a] - [B, a] + [B, d]$ (see Figure 5.2).

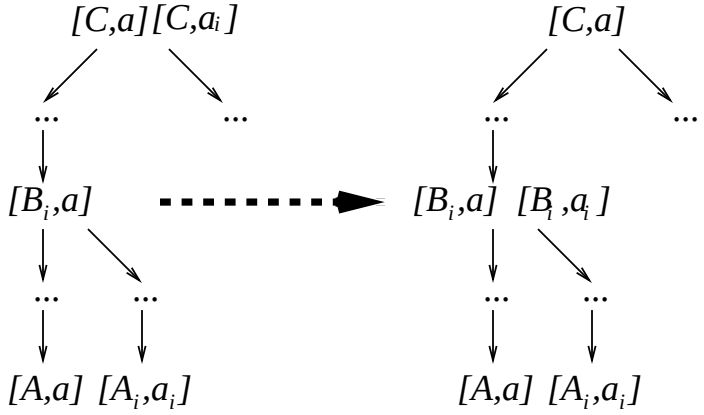

Figure 5.2: A depiction of the second swap made when C is a shared cut

We now remove $\mathcal{C}$, $[A, a]$, and $[C, a]$ from $\mathcal{L}$ and add these to our final solution $\mathcal{L}'$. Note that a is continuous in $\mathcal{L}$, and no set in $\mathcal{L}$ can induce discontinuity in a if added to $\mathcal{L}'$. We now iteratively perform this process on the remaining constraints $\mathcal{L}$. As we always make progress, and with every step we ensure a new terminal is continuous in our final basis, this proves that the final laminar basis is continuous. We now have $[D, a] \in \mathcal{L}$ for every $D \in \mathcal{C}_1$.

□

5.2.5 Application of the Fractional Token Argument

We are unfortunately not able to show that our algorithm makes progress in the general case. However, if our basic feasible solution x has a laminar basis with no shared cuts, then we can prove that we can find either an edge e with $x(e) \geq \frac{1}{2}$ or a vertex $v \in R^+$ with $z(v) \geq 1$. We can show this by means of our fractional token argument.

To apply the generalized fractional token argument, we define a matrix A with rows indexed by constraints from our maximal continuous laminar basis, $\mathcal{L}$. Columns are indexed by edges $e \in E$ and terminals $v \in R$ with $z(v) > 0$. The entry in row $[A, a] \in \mathcal{L}$ is given by $[A, a] - \sum_{[B,b] \text{ child of } [A,a]} [B, b]$. Note that A is non-singular as $\mathcal{L}$ is a basis for (x, z). Define $b_{[A,a]} = 1 - |\{\text{children of } [A, a]\}|$. Note that $[x, z]$ is the unique solution to $Ay = b$. As $\mathcal{B}$ is a basis, our matrix is non-singular and hence has some column whose entries does not sum to zero, and each row has some nonzero term. If we can show properties $[M_2 1]$-$[M_2 3]$ hold for edge columns and $[M_1 1]$-$[M_1 3]$ hold for z columns, then Lemma 3.0.9 applies, and we get that $x(e) \geq \frac{1}{2}$ for some edge $e \in E$ or $z(v) = 1$ for some terminal $v \in R$.

First we justify that $[M_2 1]$-$[M_2 4]$ hold for the edge columns in our system. Consider a node $[A, a]$ and edge $e = uv$. Note that at most two of $[A, a]$'s children can have the same edge in their cut, as otherwise the children would not be disjoint, violating laminarity of $\mathcal{L}$. For an edge $e = uv$, the lowest node containing u and the lowest node containing v are the only nodes that can have a positive entry, and these are the only nodes possible to have e in their cut, and yet not have any children with e in their cut.

If $[A, a]$ has entry -1 for edge e, then there is a unique child, say $[B, b]$ with one endpoint, say u in B. We must have $v \in A$, as otherwise e would be in $\delta(A)$. Further, v cannot be in a separate child, as then e would be in that child's cut, resulting in an entry of -2 for $[A, a]$. The lowest node containing u will not contain v as it is a descendent of $[B, b]$, and hence gets entry of 1 for column e. As A is the lowest node containing v, but e is not in $\delta(A)$, there is no other positive entry for this column.

Finally, if $[A, a]$ has entry -2 for edge e, then there are two children of A, one containing u, the other containing v. The lowest node containing u will thus have e in its cut, and hence an entry of 1, similarly the lowest node containing v will have an entry of 1 as well.

Note that by the fact that $\mathcal{L}$ is a feasible basis, we have that for any non-pure constraint $[A, a] \in \mathcal{L}$, $a \in A$. Combined with the fact that $\mathcal{L}$ is laminar and continuous, we see that any $v \in R$ with $z(v) > 0$ induces a chain in our laminar forest $F_{\mathcal{L}}$ of constraints that are tight

using v. Consider the column of our matrix corresponding to v. The row corresponding to the parent of the top node of this induced chain will have entry -1, the row for the lowest node in the chain will have entry 1, and any other row will have entry 0. This establishes properties $[M_1 1]$-$[M_1 3]$, and thus justifies the use of Lemma 3.0.9, proving that we have either an edge e with value $x(e) \geq \frac{1}{2}$, or a terminal v with $z(v) = 1$.

We show in the next section one possible method of resolving the problem of shared cuts, and how it naturally induces a new network design problem.

5.3 Prize-Collecting Generalized Steiner Tree

One method that we looked at for resolving shared cuts was to perform a column merge on the associated terminals. Suppose we have $[S, a], [S, b] \in \mathcal{B}$, with $a, b \in R^+, a \neq b$, and consider the new problem after performing a column merge on variables a and b, for which x, z is still a basic feasible solution (see Lemma 5.1.3). This new system defines a new problem, which we call the Prize-Collecting Generalized Steiner Tree problem (PC-GST) (not to be confused with the Prize-Collecting Steiner Forest problem). It is defined by a graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}_+$, groups of vertices $R_1, R_2, ..., R_k$ with a common root r in each, and a penalty function $\pi : 1, ..., k \to \mathbb{Q}_+$. The goal is to find a minimum cost subgraph H of G, where the cost of H is defined to be $c(H) = \sum_{ein E(H)} c(e) + \sum_{i=1}^{k} \pi(i) z(i)$, where $z(i) = 0 \iff$ the vertices in R_i are in the same component in H.

We first present an LP for this problem in which integer solutions to the LP correspond to feasible solutions for the problem.

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e)x(e) + \sum_{v \in R^+} \pi(v)z(v) && && \text{(LP-PCGST)} \\ \text{s.t.} \quad & x(A) + z(i) \geq 1 && \forall A \subseteq V - \{r\} : A \cap (R_i - \{r\}) \neq \emptyset && (5.3) \\ & 0 \leq x(e), z(i) \leq 1 && \forall e \in E, i \in \{1, ..., k\} \end{aligned}$$

By Lemma 5.1.3, the merging of columns a and b results in a PC-GST problem in which x, z is still a basic solution. In this new problem, we have a variable $ab \in R^+$ where

$z(ab) = 0 \iff$ both a and b are connected to the root r, so it is easy to see that we are now in a PC-GST problem.

5.3.1 Counterexample for an Iterative Rounding Approximation Algorithm

We had hoped that the PC-GST problem could be approached via iterative rounding, which would allow us to then solve our original PC-ST problem. In order to achieve the 2 Lagrange multiplier preserving approximation algorithm that we were hoping for in the PCST, we need to be able to show that in any basic solution to PCRSF, there is either a zero edge variable, an edge with value at least $\frac{1}{2}$, or a z variable with value 1. However, the following example that shows that there are basic feasible solutions for the PCSRF problem in which every edge variable has value $\frac{1}{k}$, and a maximal z value of $\frac{k-2}{k}$.

We have some $k \geq 3$, $R_1 = \{r, 1, 2, ..., k\}$, $\pi(1) = 1$, $R_2 = \{r, v\}$, $\pi(2) = m$ for some large m, $c(e) = 1$ for every edge $e \in E$.

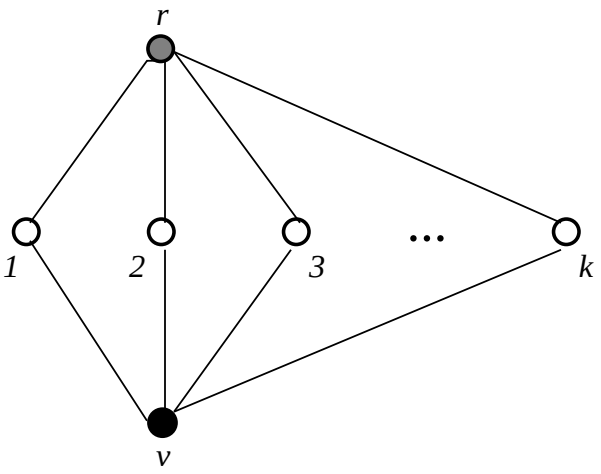

Figure 5.3: Example with a bad basic solution

Note that the optimal solution is $x(e) = \frac{1}{k}$ for every edge e and $z(1) = \frac{k-2}{k}$, $z(2) = 0$. It has optimal value $2 + \frac{k-2}{k} < 3$. If we set $k = 3$, we see that $z(1) = \frac{1}{3} = x(e)$ for every edge e, meaning that an iterative rounding approach to this problem will not be able to do any better than a 3-approximation.

Note that to show that x, z is a basic solution, we require $2k+2$ tight, linearly independent constraints that hold for the solution. In this case, one such basis for x, z are given by the constraint $z(2) = 0$ in addition to the constraints of type 5.3 for the following set group pairs:

1. $(\{i\}, 1)$ for each $i \in \{1, ..., k\}$
2. $(\{v\}, 2)$
3. $(\{v, i\}, 2)$ for each $i \in \{1, ..., k\}$

There is, however, a nice way to deal with this extreme point. Note that if we multiply our basic feasible solution by k, we now get that every edge value is integer. We then put $k \cdot x(e)$ as the capacity on each edge, and then double each edge to get an Eulerian graph. Let $\lambda(r, x)$ denote the connectivity between r and x.

We would like a result similar to this following theorem by Bang-Jensen et. al. [4], but with an additional property we will explain later. In this result, a pre-flow digraph is one in which every vertex other than a special root has more incoming edges than outgoing edges, and thus we may think of our Eulerian graph as a pre-flow digraph.

5.3.1 Theorem ([4])**.** *In a pre-flow digraph $D = (U, A)$ with root z, for any integer $k \geq 1$ there is a family $\mathcal{F}$ of k disjoint arborescences of root z so that every node x belongs to* $\min(k, \lambda(z, x))$ *members of $\mathcal{F}$.*

In our case we use $z = r$ and apply for k trees. For our basic feasible solution, we can pick trees as follows:

1. Define T_i to be the tree $ri + iv$ for each $i \in \{1, ..., k-2\}$.
2. Define T_{k-1} to be $\bigcup_{i=1}^{k} ri$
3. Define T_k to be $rk + ki + \bigcup_{i=1}^{k-1} iv$

Note that we additionally have the property that all of R_1 is contained within the 2 trees T_{k-1}, T_k, and $R_2 = \{v\}$ is contained within all k trees. That is, R_1 is contained within a $1 - z(1) = \frac{2}{k}$ fraction of all trees, and R_2 is contained within a $1 - z(2) = 1$ fraction of trees (i.e. all trees).

In general, this special property that we desire is that R_i is contained within at least $1 - z(i)$ fraction of all trees.

We let $c(T_i) + \pi(T_i)$ represent the cost for feasible solution T_i in the PC-GST problem. Choose i^* to be a tree of minimum cost. Note that by construction,

$$\begin{aligned} c(T_{i^*}) + \pi(T_{i^*}) &\leq \frac{1}{k}(\sum_{i=1}^{k} c(T_i) + \sum_{i=1}^{k} \pi(T_i)) \\ &= \frac{1}{k} \cdot \sum_{i=1}^{k} c(T_i) + \frac{1}{k} \cdot \sum_{i=1}^{k} \pi(T_i) \\ &\leq 2c^T x + \pi(z) \end{aligned}$$

Thus if we can always obtain such a T_{i^*} we can get a 2-approximation algorithm for PC-GST that is Lagrangean multiplier preserving. In order to get such an algorithm, we need the following conjecture, as explained above.

5.3.2 Conjecture. *Let $D = (V, A)$ be an Eulerian multigraph with a special vertex z, and terminal groups of vertices $R_1, R_2, ..., R_s$ with z in each group. There is a family $\mathcal{F}$ of k edge-disjoint trees rooted at z so that every group $R_i \subseteq V$ belongs to at least $r(x) = \min(k, \frac{1}{2} \cdot \lambda(z, R_i))$ members of $\mathcal{F}$, where $\lambda(z, R_i)$ is the minimum connectivity between z and any vertex $v \in R_i$.*

5.3.2 Hardness of Approximation

We provide some insights that may show that it is unlikely that we will be able to find a Lagrangean multiplier preserving approximation for PC-GST. We first need to introduce a problem that we call the k-Generalized Steiner Tree (k-GST), in which our goal is to find a minimum cost tree that connects at least k groups R_i.

As explained earlier, Chudak et al. [11] have shown how a constant factor Lagrangean multiplier preserving algorithm for PC-ST can be translated into a constant factor approximation algorithm for the k-Steiner Tree problem through a *Lagrangean relaxation* technique. A parallel result is not yet known for k-GST, but a similar approach could work. Thus if we show that it is hard to create a constant factor k-GST approximation algorithm, it is likely also hard to find a constant factor Lagrangean multiplier preserving approximation algorithm for PC-GST.

First, we introduce a few other problems for which we have known hardness results.

Given a graph $G = (V, E)$ and a parameter k, the DENSEST k-SUBGRAPH problem is to find a set of k vertices with maximum number of induced edges. This problem has been well studied [36, 15], and the best known approximation algorithm is a $O(n^{\frac{1}{3}} - \epsilon)$-approximation algorithm for some small $\epsilon > 0$. [16]. On the hardness side, it is known that there is no *polynomial time approximation scheme* for this problem [36].

The MINIMUM k-EDGE COVERAGE (MkEC) problem is the problem of finding the minimum number of vertices in a graph G whose induced subgraph has at least k edges. Hajiaghayi and Jain [29] show the following theorem.

5.3.3 Theorem ([29])**.** *If there is a polynomial time α-approximation algorithm $\mathcal{A}$ for* MINIMUM k-EDGE COVER*, then there is a polynomial time $2 \cdot \alpha^2$-approximation algorithm for the* DENSEST k-SUBGRAPH *problem.*

We now show a reduction from MkEC to k-GST.

5.3.4 Theorem. *If there is a polynomial time α-approximation for k-*GST*, then there is a polynomial time α-approximation for* MkEC*.*

Proof. For a graph instance $G = (V, E)$ of MkEC, we construct an instance of the k-GST problem as follows. Construct a star graph S with a center c and a unit-cost edge cv for every $v \in V(G)$. For each edge $uv \in E(G)$, construct a group $R_{uv} = \{c, u, v\}$ for our k-GST instance.

Now, note that any subgraph of S that connects at least k of the groups R_{uv} corresponds to a set of vertices whose induced subgraph in G has at least k edges, and vice versa. Thus we apply our α-approximation for PC-GST to S to get an α-approximation to MkEC. □

The reduction shown immediately gives us the following result.

5.3.5 Corollary. *If there is a polynomial time α-approximation for k-*GST*, then there is a polynomial time $O(\alpha^2)$ approximation for the* DENSEST k-SUBGRAPH *problem.*

Corollary 5.3.5 gives us that any algorithm for k-GST of approximation factor better than $O(n^{\frac{1}{6}-\epsilon})$ improves the best current approximation factor for the DENSEST k-SUBGRAPH problem. A further observation is that the k-GST problem is hard even on star graphs.

Following the *Lagrangean relaxation techniques* explained in [11], one may be able to show that a constant factor Lagrangean multiplier preserving algorithm for PC-GST would result in a constant factor approximation algorithm for k-GST. Of course, this must be further investigated. If this parallel result holds, however, this would imply that it is hard to find a constant factor Lagrangean multiplier preserving algorithm for PC-GST.

Chapter 6

Future Work

After looking at many different network design approximation algorithms, there are several remaining open questions. One question is whether the generalized fractional token argument present in Chapter 3 can be used to produce a Lagrangian multiplier preserving algorithm for any network design problem, but in particular, for the Prize-Collecting Steiner Tree problem. Are there other scenarios in which we can use this generalized version of the argument. We briefly looked at a hyperedge covering problem, in which case we would begin to see the submatrix A_3 come into play, but the constraints did not take on the exact form described. When subtracting characteristic vectors of cuts in hypergraphs, we can end up with multiple negative entries in one column, which is not allowed by the current version of the token argument. Can the argument be modified to deal with this scenario?

Another open question is the existence of a bicriteria approximation algorithm for the Degree Bounded Element Connectivity problem, with degree bounds on all vertices. One could attempt to determine whether or not the $(2, 2B + 3)$-approximation presented in this paper will work with degree bounds on nonterminals (see Conjecture 4.2.15). As of yet, there is no non-trivial approximation algorithm that can deal with degree bounds on nonterminals. Another direction would be to improve the error on the degree bounds. Perhaps a better bicriteria algorithm can be proven using Lau and Singh's $(2, 6r_{max} + 3)$-approximation for Degree Bounded Survivable Network Design as a model.

One could formalize whether the *Lagrangean relaxation techniques* explained in [11]

will translate to the creation of a constant factor approximation algorithm for the k-Generalized Steiner Tree problem given a constant factor Lagrangean multiplier preserving algorithm for the Prize Collecting Generalized Steiner Tree problem. If this is the case, as mentioned in Section 5.3, it would be hard to find such a constant factor Lagrangean multiplier preserving algorithm for PC-GST.

One last open question in this area would be to consider whether the current best of 2.54-approximation algorithms for Prize-Collecting Steiner Forest, Survivable Network Design, and Element Connectivity can be improved to 2-approximations with some new insight.

References

[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. In Baruch Awerbuch, editor, *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 134–144, New Orleans, LS, May 1991. ACM Press. 5, 6

[2] Aaron Archer, MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for PRIZE-COLLECTING STEINER TREE and TSP. In *FOCS*, pages 427–436. IEEE Computer Society, 2009. 7

[3] Balas. The prize collecting traveling salesman problem. *NETWORKS: Networks: An International Journal*, 19, 1989. 7

[4] Jørgen Bang-Jensen, András Frank, and Bill Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM J. Discrete Math.*, 8(2):155–178, 1995. 86

[5] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree bounded directed network design. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 769–778, New York, NY, USA, 2008. ACM. 2

[6] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Program*, 59:413–420, 1993. 7

[7] Boyd and Pulleyblank. Optimizing over the subtour polytope of the travelling salesman problem. *MATHPROG: Mathematical Programming*, 49, 1991. 2

[8] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanitá. An improved lp-based approximation for steiner tree. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2010. 5

[9] Cheriyan, Vempala, and Vetta. Network design via iterative rounding of setpair relaxations. *COMBINAT: Combinatorica*, 26, 2006. 1, 6

[10] Edwin Kah Pin Chong and Stanislaw H. Çzak. *An introduction to optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, pub-WILEY:adr, second edition, 2001. 14

[11] F. A. Chudak, T. Roughgarden, and D. P. Williamson. Approximate k-MSTs and K-Steiner trees via the primal-dual method and Lagrangean relaxation. *Mathematical Programming*, 100(2):411–421, 2004. 74, 87, 89, 90

[12] Julia Chuzhoy and Sanjeev Khanna. An O(k^3log n)-approximation algorithm for vertex-connectivity survivable network design. *CoRR*, abs/0812.4442, 2008. informal publication. 6

[13] Edmonds. Submodular functions, matroids, and certain polyhedra. In *International Workshop on Combinatorial Optimization, LNCS*, volume 5, 2001. 2

[14] Jack Edmonds and Rick Giles. A min-max relation for submodular functions on graphs. In *Studies in integer programming (Proc. Workshop, Bonn, 1975)*, pages 185–204. Ann. of Discrete Math., Vol. 1. North-Holland, Amsterdam, 1977. 2

[15] Feige, Peleg, and Kortsarz. The dense k-subgraph problem. *ALGRTHMICA: Algorithmica*, 29, 2001. 88

[16] Uriel Feige. Relations between average case complexity and approximation complexity. In *STOC*, pages 534–543, 2002. 88

[17] Fleischer, Jain, and Williamson. An iterative rounding 2-approximation algorithm for the element connectivity problem. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001. 2, 4, 5, 6, 21, 23, 24, 26, 27, 28, 29, 35, 36, 39, 40, 42

[18] Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. System Sci.*, 72(5):838–867, 2006. 1

[19] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian J. of Mathematics*, 8:399–404, 1956. 12

[20] András Frank. Kernel systems of directed graphs. *Acta Sci. Math. (Szeged)*, 41(1-2):63–76, 1979. 2

[21] Furer and Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1992. 7

[22] Furer and Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *ALGORITHMS: Journal of Algorithms*, 17, 1994. 7

[23] Harold N. Gabow. On the $L_{infinity}$-norm of extreme points for crossing supermodular directed network LPs. In Michael Jünger and Volker Kaibel, editors, *Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, Proceedings*, volume 3509 of *Lecture Notes in Computer Science*, pages 392–406. Springer, 2005. 1

[24] M. X. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994. 18

[25] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 307–316, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics. 6, 7, 74

[26] Fabrizio Grandoni, Jochen Könemann, Thomas Rothvoß, and Chaitanya Swamy. An iterated rounding 3-approximation algorithm for prize-collecting steiner forest. Manuscript, 2009. 5, 7, 66, 67, 68, 72

[27] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988. 14, 43

[28] M. Hajiaghayi, R. Khandekar, G. Kortsars, and Z. Nutov. Prize-collecting steiner network problems. In *Integer Programming and Combinatorial Optimization, 16th International IPCO Conference, 2010, Proceedings*, 2010. 7

[29] M. T. Hajiaghayi and K. Jain. The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema. In *SODA*, pages 631–640, 2006. 3, 7, 88

[30] MohammadTaghi Hajiaghayi and Arefeh Nasri. Prize-collecting steiner networks via iterative rounding. In *Proceedings of the 9th Latin American Symposium on Theoretical Informatics*, Oaxaca, Mexico, 2010. 7, 67, 70

[31] Hwang, Richards, and Winter. The steiner tree problem. *ANNALSDM: Annals of Discrete Mathematics*, 53, 1992. 3, 5

[32] Jain, Mandoiu, Vazirani, and Williamson. A primal-dual schema based approximation algorithm for the element connectivity problem. *ALGORITHMS: Journal of Algorithms*, 45, 2002. 6, 35

[33] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS-98)*, pages 448–457, Los Alamitos, CA, November8–11 1998. IEEE Computer Society. iii, 1, 2, 6, 8, 9, 15, 17, 18, 19, 20, 21, 36, 55

[34] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations, Miller,Tatcher, Plenum*. 1974. 3

[35] Marek Karpinski and Alexander Zelikovsky. New approximation algorithms for the steiner tree problems. Technical Report TR-95-036, International Computer Science Institute, Berkeley, CA, August 1995. 5

[36] Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SICOMP: SIAM Journal on Computing*, 36, 2006. 88

[37] Lau, Naor, Salavatipour, and Singh. Survivable network design with degree or order constraints. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2007. 35, 47

[38] Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009. iii, 2, 5, 7

[39] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 759–768, New York, NY, USA, 2008. ACM. 2, 7

[40] Vardges Melkonian and Éva Tardos. Algorithms for a network design problem with crossing supermodular demands. *Networks*, 43(4):256–265, 2004. 1

[41] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927. 12

[42] Viswanath Nagarajan, R. Ravi, and Mohit Singh. Simpler analysis of lp extreme points for traveling salesman and survivable network design problems. *Operations Research Letters*, 38(3):156 – 160, 2010. iii, 2, 5, 10, 30

[43] Hans Juergen Proemel and Angelika Steger. RNC-approximation algorithms for the steiner problem. Institutsbericht, Technische Universitaet Muenchen, Institut fuer Informatik, October 1996. 5

[44] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001. 7

[45] Robins and Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIJDM: SIAM Journal on Discrete Mathematics*, 19, 2005. 5

[46] Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *SODA*, pages 770–779, 2000. 5

[47] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons Ltd., Chichester, 1986. A Wiley-Interscience Publication. 13, 14

[48] Yogeshwer Sharma, Chaitanya Swamy, and David P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete*

algorithms, pages 1275–1284, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 7

[49] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 661–670, New York, NY, USA, 2007. ACM. 7

[50] Chaitanya Swamy. personal communication, 2010. 5, 30

[51] A. Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993. 5